Team 6

Poornima Dixith

Rahul Deshpande

Sowmya Tanneri

Introduction

- Azure Cosmos DB is Microsoft's globally distributed, multi-model database service "for managing data at planet-scale“, launched in May 2017
- It builds upon and extends the earlier Azure DocumentDB, which was released in 2014.
- It is schema-less and generally classified as a NoSQL database.
- It offers throughput, latency, availability, and consistency guarantees with comprehensive service level agreements (SLAs), something no other database service can offer

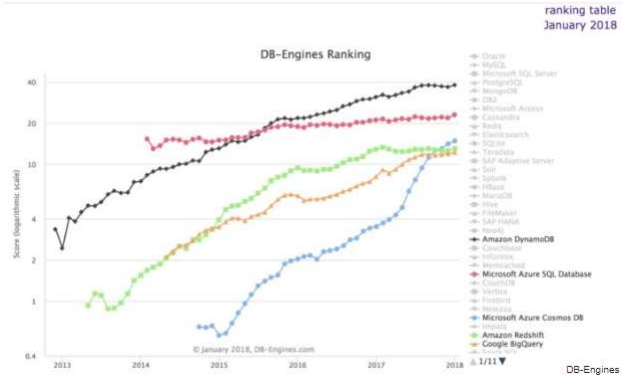
History – Who?

- Microsoft Corporation invented Cosmos DB.
- **Competing/Similar alternatives**
  - Google Cloud Spanner.
  -Amazon RDS + Amazon DynamoDB + Amazon Redshift.

History – Why ?

- **Enable customers to elastically scale throughput and storage based on demand, globally.**
  -System should deliver the configured throughput within 5 seconds at the 99th percentile, from the time of the request to scale.
- **Enable customers to build highly responsive, mission-critical applications**.
  -System must deliver predictable and guaranteed end-to-end low read and write latencies at the 99th percentile.
- **Ensure that the system is "always on".**
  -System must provide 99.99% availability regardless of the number of regions associated with their database.
- **-Enable developers to write correct globally distributed applications**.
  -System must offer an intuitive and predictable programming model around data consistency.
- **Relieve the developers from the burden of database schema/index management and versioning**.
  -Keeping database schema and indexes in-sync with an application's schema is painful for globally distributed applications.
- **Natively support multiple data models and popular APIs for accessing data.**
  -The translation between the externally exposed APIs and internal data representation needed to be efficient.
- **Operate at a very low cost**
  -To pass on the savings to customers.

History – How ?



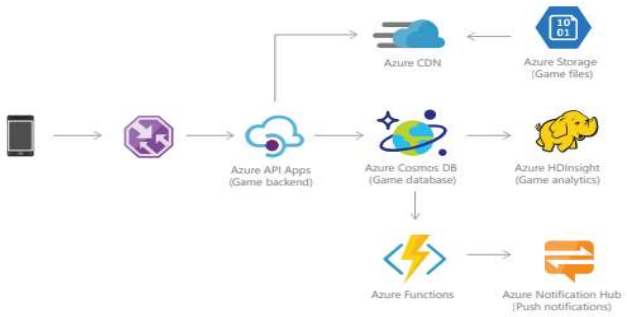
-Amazon Redshift for data warehousing.
-Amazon Aurora/RDS for traditional relational workloads.
-AWS DynamoDB for NoSQL.
-Cosmos DB seems to be heading in the opposite direction, with a one-size-fits-all approach to data.

Features

- **Global distribution**
  - Cosmos DB automatically replicates all of your data to any number of regions of your choice, for fast, responsive access
- **Multi-model APIs**
  - Cosmos DB allows you to use key-value, graph, and document data in one service, at global scale and without worrying about schema or index management. . Cosmos DB automatically indexes all data, and allows you to use your favorite NoSQL API including SQL, JavaScript, Gremlin, MongoDB, and Azure Table storage to query your data
- **Provisioned throughput**
  - Cosmos DB allows you independently and elastically scale storage and throughput across one or multiple global regions.
- **Choice of consistency**
  - Cosmos DB offers five well-defined consistency levels—strong, bounded staleness, session, consistent-prefix and eventual—for an intuitive programming model with low latency and high availability for applications spanning the world.
- **Comprehensive SLAs**
  - . Cosmos DB is the first and only service to offer industry-leading comprehensive 99.99% SLAs for latency at the 99th percentile, guaranteed throughput, consistency and high availability

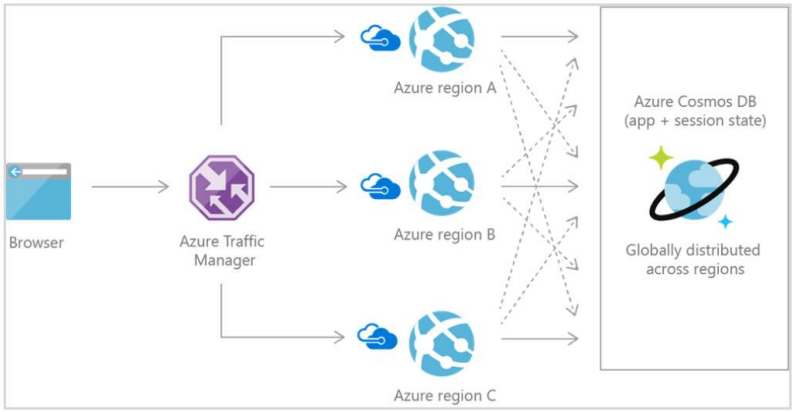
Example Use Case

- **Gaming**
  - To deliver social and personalized content like in-game stats and high-score.
  - Single-millisecond latencies for reads and writes to deliver lag-free experiences.



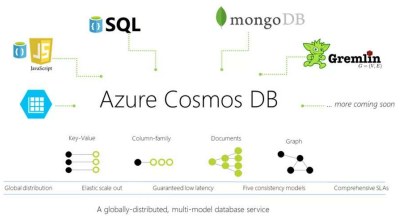
Web and mobile applications

- To store and query chat sessions, tweets, blog posts, ratings, and comments for web, mobile, and social media applications.
- Automatic indexing of data by cosmos DB helps the third-party social networks integrated with these applications to have flexibility in retrieving data.



Advantages

- Multiple data models and multiple API support, so we are no longer limited to just one platform



- Five consistency models used to achieve high availability and low latency.

Disadvantages

- Costly for small business
- Few query limitations when using document DB API

Cost Discussion

- In Azure cosmos DB, we are billed for the provisioned throughput and consumed SSD storage.
- Throughput is expressed in request units per second(RU/s) which can be provisioned at granular levels(Cosmos DB container/Cosmos DB database).

Pricing details

UNIT	PRICE
SSD Storage (per GB)	$0.25 GB/month
Provisioned Throughput (single region writes) per 100 RU/s	$0.008/hour
Provisioned Throughput (multi region writes) per 100 RU/s	$0.016/hour

Reserved capacity based pricing of Cosmos DB offers even more cost savings (up to 65% discount and eases the burden of capacity planning away from the user. For a one-time upfront fee, user can reserve provisioned throughput for one or three years at a significant discount.

	1 YEAR RESERVATION		3 YEAR RESERVATION	
THROUGHPUT	SINGLE REGION WRITE	MULTIPLE REGION WRITE	SINGLE REGION WRITE	MULTIPLE REGION WRITE
PRICE/SAVINGS	PRICE PER 100 RU/S (SAVINGS OVER PAYG)	PRICE PER 100 RU/S (SAVINGS OVER PAYG)	PRICE PER 100 RU/S (SAVINGS OVER PAYG)	PRICE PER 100 RU/S (SAVINGS OVER PAYG)
First 50K RU/s	$0.0068 (~15%)	$0.0128 (~20%)	$0.006 (~25%)	$0.0112 (~30%)
Next 450k RU/s	$0.006 (~25%)	$0.0112 (~30%)	$0.0052 (~35%)	$0.0096 (~40%)
Next 2.5M RU/s	$0.0056 (~30%)	$0.0104 (~35%)	$0.0044 (~45%)	$0.008 (~50%)
Over 3M RU/s	$0.0044 (~45%)	$0.008 (~50%)	$0.0032 (~60%)	$0.0056 (~65%)

Cost Example

Scenario 1

There is one container in West US with provisioned throughput of 20K RU/s and three containers in East US, East Asia, North Europe with 20K RU/s. The storage for each of the container is 1TB. Cost with single region write throughput and billing options(pay as you go, 1 year reserved, 3 year reserved)

Item	Usage(Month)	Rate	Pay as you go	1 year reserved	3 year reserved
Throughput bill for container in West US	20K RU/s x 24 x 31	$0.008 per 100 RU/s per hour	$1190.40	$992.80	$876.00
Throughput bill for container in 3 regions(East US, East Asia, North Europe)	3 x 20K RU/s x 24 x 31	$0.008 per 100 RU/s per hour	$3,571.20	$2,978.40	$2628.00
Storage bill for container in West US	1 TB(1 x 1024 GB)	$0.25/GB	$256.00	$256.00	$256.00
Storage bill for container in East US, North Europe and East Asia	3 x 1 TB (3 x 1024 GB)	$0.25/GB	$768.00	$768.00	$768.00
Total			$5785.60	$4995.20	$4528.00

Scenario 2

There is one container in West US with provisioned throughput of 20K RU/s and three containers in East US, East Asia, North Europe with 20K RU/s. The storage for each of the container is 1TB. Cost with multiple region write throughput and billing option(pay as you go, 1 year reserved, 3 year reserved)

Item	Usage(Month)	Rate	Pay as you go	1 year reserved	3 year reserved
Throughput bill for container in West US	20K RU/s x 24 x 31	$0.016 per 100 RU/s per hour	$2380.80	$1868.80	$1635.20
Throughput bill for container in 3 regions(East US, East Asia, North Europe)	3 x 20K RU/s x 24 x 31	$0.016 per 100 RU/s per hour	$7142.40	$5606.40	$4905.60
Storage bill for container in West US	1 TB(1 x 1024 GB)	$0.25/GB	$256.00	$256.00	$256.00
Storage bill for container in East US, North Europe and East Asia	3 x 1 TB ()	$0.25/GB	$768.00	$768.00	$768.00
Total			$10,547.20	$8499.20	$7564.80

Conclusion

- Flexible and reliable non relational database.
- High availability and low latency.
- Cost effective in some use cases.

DEMO

WHICH API TYPE?

- Options
  - SQL
  - MongoDB
  - Gremlin
  - Cassandra
  - Table

SQL API TYPE – DATA STRUCTURE

- Data Stored As Json Documents (Schemaless)
- A Document Db Server hosts Databases
- Each Database has multiple collections
- Each collection has multiple documents

DEMO STEPS

- CREATE THE DOCUMENT DB SERVER
- CREATE THE DATABASE -> COLLECTION -> DOCUMENT
- DEMONSTRATE CLIENT SDK USAGE
- PRINT RESULTS
- DISPLAY THE POWER OF QUERIES

STEP 1: CREATE THE DOCUMENT DB SERVER

- OPTIONS:
  - PowerShell
  - Azure CLI
  - Azure Portal
  - C# Client SDK

STEP 2: CREATE DATABASE, COLLECTION AND DOCUMENT

- OPTIONS
  - Azure Portal
  - PowerShell
  - C# Client Application (Azure Cosmos DB SDK)
  - Azure CLI

STEP 3: DEMONSTRATE CLIENT SDK USAGE

- Options:
  - .NET
  - Java
  - Node.js
  - Python
  - PowerShell
  - Azure CLI

STEP 4: EXECUTE SQL ON THE PORTAL

- SAMPLE QUERY TYPES:
  - ALIASING
  - AGGREGATE
  - USER DEFINED FUNCTIONS

THANK YOU PROFESSOR LLOYD!

BIBLIOGRAPHY

- Tutorial References:
  - https://docs.microsoft.com/en-us/azure/cosmos-db/
- Images
  - https://makeameme.org/meme/lets-get-started-5ab513
  - http://fr.memegenerator.net/instance/72004484/any-questions-thanks-for-listening-any-questions
  - https://makeameme.org/meme/any-questions-wed


THANK YOU!!!
imgflip.com