CLOUD COMPUTING APPLICATIONS

Cloud Databases – Microsoft Azure CosmosDB

Prof. Reza Farivar

Azure CosmosDB

- Globally distributed, multi-model database
  - Wire compatible with Cassandra
  - Table API
    - 5 types of consistency levels
  - MongoDB API
  - Etcd API
    - etcd is a consistent distributed key value storage
    - Compare with Zookeeper
    - Backend of Kubernetes
  - Gremlin API
- Replicated State Machine (RSM) for concurrency
  - Specific algorithm not published
  - *RAFT is also a state machine consensus algorithm*

Azure CosmosDB

- Write-optimized, resource-governed and schema-agnostic database engine
  - It automatically indexes everything it ingests
    - Internally based on a document store model
    - All JSON documents are a tree
    - Make an index for each path of the tree
  - **Synchronously** makes the index durable and highly available before acknowledging the client's updates while maintaining low latency guarantees
- BW-Tree indexing
  - Log Structure Record
  - Latch free updates
  - Atom-record-sequence (ARS) system
- Uses Lamport's TLA+ specification language to describe SLAs at each consistency level

CosmosDB Consistency Models

- **Strong:** With strong consistency, you are guaranteed to always read the latest version of an item similar to read committed isolation in SQL Server. You can only ever see data which is durably committed. Strong consistency is scoped to a single region.
- **Bounded-staleness:** In bounded-staleness consistency read will lag behind writes and guarantees global order and not scoped to a single region. When configuring bounded-staleness consistency you need to specify the maximum lag by:
  - Operations: For a single region the maximum operations lag must be between 10 and 1,000,000, and for the multi region, it will be between 100,000 and 1,000,000.
  - Time: The maximum lag must be between 5 seconds and 1 day for either single or multi-regions.
- **Session:** This is the most popular consistency level, since it provides consistency guarantees but also has better throughput.
- **Consistent Prefix:** A global order is preserved, and prefix order is guaranteed. A user will never see writes in a different order than that is which it was written.
- **Eventual:** Basically, this is like asynchronous synchronization. It guarantees that all changes will be replicated eventually, and as such, it also has the lowest latency because it does not need to wait on any commits.

Figure 4. Multiple well-defined consistency choices along the spectrum.

Consistency Level	Guarantees
Strong	Linearizability
Bounded Staleness	Consistent Prefix. Reads lag behind writes by *k* prefixes or *t* interval
Session	Consistent Prefix. Monotonic reads, monotonic writes, read-your-writes, write-follows-reads
Consistent Prefix	Updates returned are some prefix of all the updates, with no gaps
Eventual	Out of order reads

Azure CosmosDB Implementation

- Cosmos DB service is deployed on several replicated shared-nothing nodes across geographical regions for high-availability, low-latency, and high throughput.
- Some or all of these distributed nodes form a replica set for serving requests on a data shard that contains documents.
- Among the replicas, one of them is elected as a master to perform totally-ordered writes on the data shard.
- Writes are done on the write-quorum (W), a subset of the replica nodes, to ensure that the data is durable.
- Reads are performed on read-quorum (R), a subset of replica nodes, to get the desired consistency levels (Strong, Bounded-staleness, Session, Consistent Prefix, Eventual) as configured by users.
- Data is partitioned at logic level and is replicated at storage layer in terms of physical partitions to achieve desired availability and throughput.
- storage
  - transactional storage engine
  - analytical storage engine
  - storage engines are log-structured and write-optimized