Instituto Tecnologico de Costa Rica

Sede Central de Cartago

Escuela de Computación

Resumen 4 (R4)

IC-4302 Bases De Datos II GR 1

Estudiantes: Deyan Sanabria Fallas #2021046131

Profesor: Gerardo Nereo Campos Araya

Fecha de Entrega: 21/09/2022

Segundo Semestre 2022

Indice

What is Azure DocumentDB?

Microsoft's multi-tenant distributed database service for managing JSON documents at Internet scale.

Capabilities and architecture overview

Capabilities:

> - Query language supports rich relational and hierarchical queries.
> - Optimized for consistent queries in the face of sustained high volume document writes.
> - Application logic provided via stored procedures and triggers authored entirely in JavaScript.
> - Well-defined and tunable consistency levels.
> - All machine and resource management is abstracted from users.
> - Developers can interact with resources via HTTP verbs for CRUD, queries and stored procedures.

Architecture:

> - In a cluster, DocumentDB service has a dedicated local SSDs in each machine
> - DocumentDB persists data on local SSDs and replicates it among instances within the replica set.

Indexing subsystem goals:

> - Automatic Indexing
> - Configurable Storage/Performance Tradeoffs
> - Efficient, rich hierarchical and relational queries
> - Consistent queries in face of sustained volume of document writes
> - Multi-tenancy

Schema Agnostic Indexing

Lack of a schema specification, makes no assumption about the document and allows collections to vary in schemas. The database engine operates directly at the leve of JSON Grammar

Documents as Trees:

> - DocumentDB represents documents as trees which normalizes the structure and the instance values. The interior nodes represent the schema and the leaf nodes represent the values

Index as a Document:

> - The index can be serialized to a valid JSON Document. The cost to index and query a document is the same as a flat JSON Document.

DocumentDB Queries:

> - DocumentDB Supports queries written in SQL and JavaScript, Both get translated to Query IL which supports projections, filters, aggregates, sorts, flatten operators, expressions, system

> provided instrinsics and user defined functions.

Logical Index Organization

Directed Path as terms:

> - *Foward Path* starts from each node in the tree to a leaf, *Reverse Path* from leaf to the root.
> - Storage cost is measured by the number of paths generated for the index structure.
> - indexing maintenance cost is measured by resources consumed for index maintenance corresponding to a batch of document writes.
> - DocumentDB uses a combination of partial foward path for paths where range support is needed, and partial reverse path for paths needing equality support.

Bitmaps as Postings Lists:

> - A postings list captures the document ids of all the documents which contain the given term. The size of the postings list is the number of documents in the collection that contains a given term as well as the pattern of occurrence of document ids in the postings list.
> - For a dynamic posting list representation, compact and capable of computing fast set operations, DocumentDB uses two techniques: *Partitioning a Postings List* and *Dynamic Encoding of Posting Entries*.

Customizing the Index:

> Developers can customice the following aspects of the indexing policy:
>
> - Including/Excluding documents and paths to/from index.
> - Configuring Various Index Types.
> - Configuring Index Update Modes.

Physical Index Organization

The "Write" Data Structure:

> - Index maintenance must be performed against the following constraints:
>   - Index update performance must be a function of the arrival rate of the index-able paths.
>   - Index update cannot assume any path locality among the incoming documents.
>   - Index update for documents in a collection must be done within the resources allocated per DocumentDB collection.
>   - Each index update should have the least possible write amplification (ideally <= 1).
>   - Each index update should incur minimal read amplification (ideally <= 1).

The Bw-Tree for DocumentDB:

> - Uses latch-free in-memory updates and log structured storage for persistence.
> - It exploits two trends in modern hardware: multi-core processors with multi-level memory/cache hierarchy, and flash memory based SSDs with fast random reads.

Index Updates:

> - The first step is *document analysis* performed by the document analyzer.
> - Two types of Index Updates: Consistent And Lazy.

Index Replication and Recovery:

> - DocumentDB uses a single master for writes, clients issue writes against the distinguished primary replica of the replica set, which in-turn propagates the client's request guaranteeing a total order to the secondary replicas in the set.

Index Resource Governance:

> - Done with an abstract rate based currency called a Request Unit (RU/second), which encapsulates a chunk of CPU, memory and IOPS. And provides the normalized unit for accounting, provisioning, allocating and consuming throughput guarantees.

Insights from the production workloads

Document Frequency Distribution:

> - Document frequency distribution for the unique terms universally follow Zipf's Law, and validates many of the decisions made for the DocumentDB's logical index organization in the context of JSON documents.

Schema Variance:

> - regardless of the workload, document or collection size, the number of unique leaf nodes (instance values) completely dwarf the number of interior nodes (schema).

Query Performance:

> - The highest value of query precision is 1, this is when only those documents which contain the results of the query are loaded. For the same number of bytes allocated to a term, the query precision decreases as the size of the DocumentDB collection grows.

Blind Incremental Updates:

> - Doing highly performant index updates within an extremely frugal memory and IOPS budget is the key reason behind the blind incremental update access method.