Isaac Araya Solano | Carnet: 2018151703

Resumen #6

INTRODUCTION

Couchbase is a distributed, multimodel NoSQL database that offers fast in-memory performance, scalability, and advanced security. It supports various data access methods and consolidates multiple data access layers into a single platform. It provides a fully-managed database-as-a-service called Couchbase Capella, as well as Kubernetes-managed containerized cluster deployments. It supports local installations of its Community and Enterprise edition binary packages.

Couchbase has focused on delivering exceptional performance, multimodel flexibility, and SQL familiarity. It offers self-managed and fully-managed cloud-based product lines, making it a modern and versatile NoSQL database. Further details can be found in the following table and throughout the text of this paper.

It is the original multi-model NoSQL database resulting from the merger of CouchOne and Membase. It offers a reliable, scalable, and fast in-memory key-value database with document-based access. Couchbase supports SQL++ as its primary query language and embraces the open source development model. It provides low latency, reliable replication, and is available as both a fully managed service (Couchbase Capella) and self-managed deployments (Cloud Native Database Automation).

Couchbase's core performance design principles revolve around memory and network-centric architecture, multimodel data access, workload isolation, and an asynchronous approach.

JSON DATA MODEL AND ACCESS METHODS

The JSON data model supports basic and complex data types, making it convenient for web application programming. Couchbase stores data as individual documents with a key-value structure. When the value is JSON-formatted, Couchbase provides rich access capabilities. The document model allows for flexibility, enabling the storage of JSON documents with varied schemas and nested structures, eliminating the need for rigid schemas found in relational databases. Developers can easily express many-to-many relationships and access subcomponents of documents directly. The flexibility of the JSON document structure, combined with the Bucket-Scope-Collection-Document organizational hierarchy, provides developers with maximum flexibility in defining application data meta models. Couchbase's approach eliminates the lengthy schema design and deployment cycles associated with traditional RDBMS-based systems.

Document access methods

Managing JSON data is at the core of Couchbase’s document database capabilities, but there are several ways for applications to access the data. Each of these methods is described in further detail later in this paper, but the following provides a basic explanation and coding example of using it.

Key-value

An application provides a document ID (the key), Couchbase returns the associated JSON or binary object. The inverse occurs with a write or update request.

Query and Analytics

SQL-based query syntax to interact with JSON data, similiar to relational databased, returns matching JSON results. Comprehensive DML, DQL and DDL syntax supports nested data and non-uniform schema.

Full-text search

Using text analyzers with tokenization and language awareness, a search is made for a variety of field and boolean matching functions. Search returns document IDs, relevance scoring and optional context data.

Eventing

Custom Javascript functions are executed within the database as data changes or based on timers. Support for accessing and updating data, writing out to a log or calling out to an external system.

Couchbase Mobile

Couchbase Lite SDK provides common create, update, and delete tasks as well as query, full-text search, and triggers on the device. Sync Gateway keeps data updated with a Couchbase Server database and other devices.

Key, value, and sub-documents

In Couchbase, keys, values, and sub-documents play important roles in the structure of JSON documents. Keys are unique identifiers, values can be of basic or complex types, and sub-documents allow for modifying specific parts without transferring the entire document.

Key Organizing Concepts for Documents

Couchbase offers a flexible data containment model for optimized cluster performance and scalability. It consists of Buckets, Scopes, Collections, and Documents. Buckets are containers for data, with virtual (vBuckets) and ephemeral options. Scopes organize collections of documents, similar to schemas in relational databases. Collections group documents logically, analogous to tables. This multi-level structure enables efficient data organization and retrieval. Couchbase utilizes this model for organizing and accessing data, with security roles applied at different levels. Ephemeral and Memcached buckets offer in-memory options. vBuckets distribute data partitions, while Scopes and Collections provide data isolation and efficient operations. Documents adhere to JSON standards and form the foundation of Couchbase.

Cluster Design Concepts

Couchbase consists of nodes hosting single instances of Couchbase Server, forming clusters. Nodes can be added or removed from clusters, and data replication occurs within nodes and across geographically distributed clusters. Couchbase provides multiple services for data access and processing, such as Query, Indexing, Backup, Full Text Search, Analytics, and Eventing, with each service managed separately on one or more nodes.

COUCHBASE SERVICES

Couchbase implements various data access methods through dedicated services, with the Data Service at its core. Each service has its own resource quotas and can be scaled independently. Couchbase offers flexible scaling options, allowing specific services to be scaled individually. This differs from other platforms where a monolithic set of services is installed on every node. Couchbase uses a shared-nothing architecture, providing workload isolation and scalability. Applications communicate with each service through a common SDK, which is aware of the cluster's topology and configuration. The core data service handles document mutations and streams data to other services.

Data Service and Key/Value Engine

The Data Service in Couchbase is responsible for caching, persisting, and serving data to applications. It uses the KV Engine, with options for Couchstore and Magma as storage engines. The managed object cache efficiently stores and retrieves documents in memory. Documents can have TTL settings and be compressed. Memory management and compaction ensure efficient usage of resources. Mutations happen at the document level, with replication and persistence to disk occurring asynchronously within the cluster and across clusters.

Key-value data access

Couchbase is a distributed key-value (KV) store that forms the core of its document database. The KV store stores unique IDs (keys) with corresponding information (values) and provides fast, low-latency operations. It offers simple CRUD APIs for creating, reading, updating, and deleting documents by their IDs. The KV store holds the most up-to-date data, while other services provide eventually consistent indexes. KV operations are atomic, and applications can use Compare-And-Swap (CAS) to avoid conflicts during concurrent updates.

Query Service

The Couchbase query service allows users to process SQL++ queries, combining JSON flexibility with SQL power. It follows a scalable paradigm and supports independent scaling. SQL++ implements the SQL++ standard, providing a familiar language for data access. It supports NoSQL features and offers a rich set of features, including extended SELECT statements and important sub-clauses. Couchbase supports ACID transactions in SQL++, spanning multiple documents, collections, and cluster nodes. Transactional queries may be slower but provide flexibility. The service utilizes a cost-based query optimizer and has a patent for optimizing queries accessing JSON structures.

Index Service

The index service in Couchbase handles the maintenance and management of Global Secondary Indexes (GSI). It keeps indexes up to date by monitoring document mutations through the database change protocol stream (DCP) from the data service. The index service is separate from the query service, allowing for workload isolation. Various types of indexes are supported, including primary, secondary, composite/covered, functional, array, adaptive, and flex indexes. These indexes enable efficient querying of different data structures and offer flexibility in indexing and searching.

Index Advisor

Is a built-in query command, ADVICE, that interrogates the database for which index or GSI to use or build given the object and predicate selections contained in the query statement.

Couchbase provides control over query consistency, allowing applications to choose between faster queries with potential data inconsistencies or stronger consistency. The consistency levels include "not_bounded" for low latency, "at_plus" for "read-your-own-write" semantics, and "request_plus" for strong consistency. Indexes are updated asynchronously, and memory-optimized indexes (MOI) use a skip list structure to optimize memory consumption and concurrent processing. MOI is ideal for high-velocity mutations and frequent scans, requiring sufficient RAM to store all indexes.

Search Service

Couchbase has a Search service for full-text searches on JSON data with configurable indexes and analyzers. The Eventing service supports custom server-side functions triggered by data mutations. Both services offer scalable and optimized execution environments.

Analytics

Couchbase's Analytics service enables ad hoc querying without indexes, supporting real-time and operational analytics on active JSON data. It efficiently handles complex queries, including joins, aggregations, and grouping operations. The service runs on separate nodes, providing workload isolation and allowing for efficient parallel query processing. Data is pushed from the operational data stream into shadow buckets, ensuring high data freshness and preserving operational performance. The Analytics service offers advantages such as a common data model, workload isolation, high data freshness, high availability, integration with Tableau, and compatibility with data processing tools. It also allows consolidation from multiple Couchbase clusters and data ingestion from external sources like AWS S3, Azure Blob storage, and Google Cloud storage.

Mobile and the edge App Services

Couchbase Mobile enables offline-first mobile apps with NoSQL database capabilities. It includes Couchbase Lite for embedded storage and a SQL-based query API, and Sync Gateway for data synchronization, access control, and scalability with enterprise-grade security.

Couchbase Mobile has several important features:

- Offline and offline-first capabilities, ensuring apps can run and store data on the device until network connectivity is restored.
- Peer-to-peer synchronization for exchanging data between devices even with unreliable network connections.
- Delta sync, reducing network bandwidth by synchronizing only changed data.
- Client SDKs for various programming languages and platforms, including C, Objective-C, Java, Swift, Kotlin, Linux, Windows, iOS, and Android.
- Data recovery on the edge with on-device replicas for high availability and disaster recovery.
- On-device encryption for end-to-end security by encrypting the local embedded database.
- Deployment flexibility in on-premises or cloud environments, including hybrid cloud deployments for edge devices operating autonomously.
- Multi-platform support on iOS, Android, and .NET, including desktop and Windows mobile apps, with a powerful SQL-based fluent query API.