Couchbase Under the Hood

Introduction

Couchbase is a distributed, multimodel NoSQL database known for its fast in-memory performance, scalability, and advanced security features. It offers flexible JSON data modeling and provides both relational and multimodel data access services for operational and analytic applications. Couchbase can be accessed as a fully-managed database-as-a-service called Couchbase Capella or through Kubernetes-managed containerized cluster deployments. It also supports local installations with Community and Enterprise edition binary packages. With a "polyglot persistent" design architecture, Couchbase combines the performance of key-value caching, the flexibility of JSON document-based storage, and the reliability of a relational database system of record, eliminating the need for managing multiple systems and technologies.

Couchbase originated from the merger of CouchOne and Membase, bringing together the expertise of CouchDB developers and memcached architects. This union resulted in a database that supports dual-model access, allowing developers to seamlessly switch between key-value and JSON document-based approaches. Couchbase has evolved as a pioneer in multimodel NoSQL databases, expanding its capabilities beyond key-value access to include additional services like SQL++, Full-Text Search, Eventing, Analytics, and Backup. Despite its origins, Couchbase has charted its own path and should not be confused with CouchDB.

The core design principles of Couchbase revolve around memory and network-centric architecture, workload isolation, and an asynchronous approach to operations. Couchbase leverages memory caching for fast reads and allows writes to be performed in memory, offering synchronous or asynchronous replication and persistence. The database also supports multimodel data access, enabling a variety of access methods to its underlying JSON and key-value storage structures. With workload isolation, different tasks like persistence, indexing, and querying are separated, allowing independent scaling and optimization. Couchbase adopts an asynchronous approach to operations, reducing latency and blocking by processing replication, persistence, and index management in the background. These principles ensure low latency, high performance, and efficient resource utilization within the Couchbase database system.

JSON Data Model and Access Methods

Couchbase is built on the foundational JSON data model, supporting various data types and providing efficient serialization and deserialization. It stores data as individual documents with key-value pairs, allowing extensive access capabilities for JSON-formatted values and limited access for non-JSON documents. This document model offers flexibility in structuring data with varied schemas and nested structures, enabling developers to express many-to-many relationships without reference or junction tables. Subcomponents within documents can be directly accessed and updated, and multiple document schemas can be aggregated for querying. The use of JSON simplifies schema design and deployment compared to traditional RDBMS systems.

Couchbase's flexible approach allows document structures to vary, even across multiple documents, enabling efficient representation of object differences. It supports progressive schema evolution, allowing the addition of properties and structures to documents without requiring updates to other documents. This adaptability empowers applications to change behavior without extensive modifications or downtime. Couchbase offers multiple document access methods, facilitating interactions with stored JSON data. These methods provide flexibility and will be discussed further in the paper, demonstrating different ways to access and manipulate the data.

In Couchbase, keys and values are crucial in JSON documents, with certain limitations. Keys serve as unique identifiers assigned during item creation and remain immutable. They must be UTF-8 strings without spaces, with a maximum length of 250 bytes and uniqueness within the bucket. Values can be basic or complex types, such as numbers, strings, booleans, embedded documents, or arrays. The maximum value size is 20 MB. Sub-documents within JSON documents allow partial modifications without transferring the entire document over the network. Couchbase's data containment model consists of Buckets, Scopes, Collections, and Documents, providing a hierarchical structure for organizing and managing data. Buckets serve as the highest-level containers, Scopes enable data organization and access isolation, Collections group similar documents, and Documents represent individual data instances. This model aligns with relational database concepts, offering flexibility and efficiency in data management.

Couchbase Services

Couchbase employs dedicated services to facilitate data access methods, with the Data Service as its core. These services operate on top of the Data Service, which handles caching, persistence, and data serving within the cluster. Couchbase enables flexible scaling of services, independent of node size or number, allowing for workload isolation and scalability. The architecture separates services, granting developers control over workload allocation and the ability to assign dedicated nodes for specific tasks. Applications communicate with services through a common SDK, which automatically detects the cluster's topology and configuration, simplifying communication.

The Key/Value Engine supports the Data Service and manages key-value data storage and retrieval. It comprises a multi-threaded, disk-based storage layer and a managed object cache for efficient read and write operations. Each node running the Data Service has its own Key/Value Engine responsible for persisting and caching a portion of the dataset. Couchbase introduces Magma, a high data-density storage engine that enhances performance, reduces memory usage, and expands storage capacity compared to the original Couchstore.

Couchbase incorporates memory management features, including the managed object cache and item pager, to optimize memory usage. Compression options are available to reduce data size on disk and during network transfers. Compaction processes help reclaim disk space and reduce fragmentation. Document mutations are handled at the document level, triggering replication, disk persistence, and indexing processes. The key-value store, at the core of Couchbase, offers low-latency access to data through simple CRUD APIs. Key-value operations are atomic, and Compare-And-Swap functionality handles concurrent updates to the same document.

Query service

Couchbase's query service allows for scalable processing of SQL++ queries, enabling independent scaling of query workloads. SQL++ combines the flexibility of JSON with the power of SQL, providing a familiar language for data retrieval, manipulation, and creation. It supports features like nested JSON arrays, adapting queries to schema changes, and performing various data operations. ACID transactions can be defined within SQL++ to maintain data consistency across documents, collections, and cluster nodes. The query service utilizes a cost-based query optimizer to optimize query execution by leveraging available indexes and minimizing data transfer.

In summary, Couchbase's query service offers scalable processing of SQL++ queries, combining JSON flexibility with SQL power. It supports advanced features for data retrieval and manipulation, and ACID transactions provide data consistency. The service's cost-based query optimizer optimizes query execution by utilizing indexes and minimizing data transfer.

Index service

Couchbase's index service plays a crucial role in efficient query execution by managing Global Secondary Indexes (GSIs). It monitors document mutations using the database change protocol stream (DCP) and ensures that indexes are kept up to date. The index service supports various index types, including primary, secondary, composite, functional, array, adaptive, and flex indexes, catering to different data structures and query requirements. Index Advisor, a built-in query command, helps determine the appropriate index to use based on the query statement's object and predicate selections. Couchbase provides control over query consistency, allowing users to choose between faster queries with eventual consistency or stronger consistency with block-level synchronization.

Indexes in Couchbase are updated asynchronously, allowing for high write throughput but potentially introducing some data inconsistency. To address this, Couchbase offers different levels of query consistency, such as not_bounded, at_plus, and request_plus, enabling applications to balance latency and consistency based on their specific needs. Memory-optimized indexes (MOI) use a skip list structure, optimizing memory consumption and concurrent processing of index updates and scans. MOI provides highly efficient indexing for high-velocity mutations and frequent scans, operating primarily in memory and requiring sufficient RAM to store all indexes.

Search service

The Search service in Couchbase is designed for performing Full-Text Searches (FTS) on JSON data within a bucket or collection. It allows users to create, manage, and query inverted indexes for searching text within documents. The service incorporates both an indexer and query processor on each search node, and data nodes use the DCP stream to send mutations to the FTS indexer for updating indexes. Index creation is highly customizable through a JSON index definition file, the Couchbase SDK, or a graphical web interface. Different analyzers can be applied to indexes based on document type attributes, document IDs, or specific attributes. Indexes can be combined using virtual index aliases, allowing seamless transitioning between different indexes without downtime. The search process involves applying analyzers to indexed data and search requests, and matches are returned with document IDs and relevance scoring. Couchbase utilizes Bleve, an open-source search project, for the FTS capabilities of the Search service.

Eventing service

The Eventing service in Couchbase enables the creation of server-side functions triggered by data mutations. These functions, written in JavaScript, can process data from the DCP stream and perform actions such as creating or updating documents. The service scales independently and combines features of change data capture and multi-channel data streaming.

Couchbase's Analytics service provides ad hoc querying capabilities without the need for indexes. It uses the SQL++ language and supports complex queries over a large number of documents. The service runs on separate nodes, allowing for efficient parallel query processing and bulk data handling. It offers advantages such as a common data model, workload isolation, high data freshness through the DCP stream, high availability, and integration with popular tools like Tableau. Additionally, it supports consolidation from multiple Couchbase clusters and enables data ingestion from external storage systems through Remote Links.

Mobile and the edge App Services

Couchbase Mobile extends the capabilities of NoSQL databases to the edge, enabling the development of responsive and always-available mobile applications. It consists of Couchbase Lite, an embedded NoSQL JSON document store with SQL-based query APIs and support for multiple programming languages, and Sync Gateway, a synchronization gateway service responsible for data synchronization, access control, and authentication. With Couchbase Mobile, applications can operate offline and sync data when connectivity is available, enabling peer-to-peer synchronization and delta syncing to optimize network bandwidth. It provides data recovery, on-device encryption, and multi-platform support for iOS, Android, and .NET, allowing developers to build secure and scalable mobile apps.

Couchbase Mobile's offline-first approach empowers applications to function continuously even in the absence of network connectivity, ensuring uninterrupted user experiences. Peer-to-peer synchronization enables data exchange between devices with unreliable network connections, while delta sync reduces data transfer by syncing only changed data. The availability of client SDKs in multiple programming languages facilitates development for various platforms, including iOS, Android, and .NET. On-device replicas ensure data availability and disaster recovery, and on-device encryption enhances data security. Additionally, Couchbase Mobile supports deployment on premises or in the cloud, offering flexibility for enterprise users and supporting hybrid cloud deployments for edge devices operating autonomously.