Introduction

Couchbase is a modern distributed, multimodel NoSQL database. Couchbase's core architecture supports a flexible JSON data model at its foundation and uses familiar relational and multimodel data access services to supply data to operational and analytic applications.

Essential NoSQL requirements and features

NoSQL databases evolved beyond enterprise relational databases to address performance and flexibility deficiencies made evident as applications became more sophisticated and "Big Data" became an industry-standard buzzword.

Pragmatic business needs for more advanced technical requirements have pushed multimodel NoSQL databasesnto the forefront.

These modern requirements have driven Couchbase's development from inception:

- Ensure high-performance.
- Provide data model and data access flexibility.
- Support distributed cluster networks and mobility.
- Provide incredible value and low TCO.

Couchbase has been focused on setting a high standard in each of these areas. The result is a robust and accessible database platform with exceptional performance at scale, multimodel flexibility, and SQL familiarity delivered in both self-managed and fully-managed cloud-based product lines.

Core performance design principles

Memory and network-centric architecture for speed and low latency

- The most used data and indexes are transparently cached in memory for fast reads.
- Writesareperformedinmemoryandreplicatedor persisted synchronously or asynchronously.
  Using transaction guarantees ensures consistency, but may introduce lags in performance.
- InternalDatabaseChangeProtocol(DCP)streams data mutations from memory to memory at
  network speeds to support replication, indexing, and mobile synchronization.

Multimodel data access blending JSON flexibility with Key/Value speed

JSON DATA MODEL AND ACCESS METHODS

JSON Document data model

The JSON data model supports basic and complex data types: numbers, strings, nested objects, and arrays. JSON provides rapid serialization and deserialization, is native to JavaScript, and is the most common REST API data format.

Couchbase stores data as individual documents, comprised of a key and a value. When the value is JSON- formatted, Couchbase provides rich access capabilities; when not, the document is stored as a binary BLOB and has more limited access characteristics.

Key, values, and sub-documents

Keys

Each value is identified by a unique key, or ID, defined by the user or application when the item is originally created. The key is immutable: once the item is saved, the key cannot be changed.

Values

The maximum size of a value is 20 MB. Each document consists of one or more attributes, each of which has its own value. An attribute's value can be a basic type, such as a number, string, or boolean; or a complex type, such as an embedded document or an array.

Sub-documents

A sub-document is an inner component of a JSON document. The sub-document API uses a path syntax to specify attributes and array positions to read/write. This makes it unnecessary to transfer entire documents over the network when only partial modifications are required.

Cluster Design Concepts

Nodes

Couchbase nodes are physical or virtual machines that host single instances of Couchbase Server.

Clusters

A cluster consists of one or more nodes running Couchbase Server.

Services

The core of Couchbase is the Data Service that feeds and supports all the other systems and data access methods.

COUCHBASE SERVICES

Each service has its own resource quotas and, where applicable, related indexing and inter-node communication capabilities. This provides several very flexible methods to scale services when needed – not just scaling up to larger machines or scaling out to more nodes. Couchbase provides both options, as well as the ability to scale specific services independently from one another.
Multi-dimensional scaling is covered in a later section but is the foundation for Couchbase workload isolation and scaling capabilities.

Data Service and Key/Value Engine

The Data Service is the foundation for storing data in Couchbase Server must run on at least one node of every cluster – it is responsible for caching, persisting, and serving data to applications and other services within the cluster