Couchbase DB

Aktuelle DB Architekturen und Technologien

Vorlesung des Studienganges Informatik

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

Robert Orbach, Oliver Mohr, Oliver Braun

28.3.2025

Bearbeitungszeitraum	13.02.2025 bis 28.3.2025
Matrikelnummern; Kurs	8489365, xxxxxxx, xxxxxxx; INF22A
Betreuer des Dualen Partners	Prof. Dr. Carmen Winter

Inhaltsverzeichnis

1 Type of Database & History

Couchbase is a distributed NoSQL database that combines document database and key-value store capabilities. Unlike traditional single-model NoSQL solutions, Couchbase offers a hybrid approach that makes it versatile for different use cases.

1.1 Key Characteristics and Features

Couchbase distinguishes itself through several core capabilities that combine to create a versatile database platform:

1.1.1 Data Model and Storage

- **JSON Document Storage**: Schema-flexible documents supporting nested attributes, arrays, and objects (up to 20MB per document)
- **Key-Value Operations**: Sub-millisecond CRUD operations with direct key access, optimistic concurrency (CAS), and TTL expiration
- **Memory-First Architecture**: Built-in caching layer with configurable memory quotas and background disk persistence

1.1.2 Query and Indexing

- **SQL++/N1QL Query Language**: SQL-like syntax for JSON with support for JOINs, nested attributes, arrays, aggregations, and subqueries
- **Comprehensive Indexing**: Primary, secondary, composite, partial, and array indexes to optimize query performance

1.1.3 Distribution and Scalability

- **Distributed Architecture**: Automatic sharding, configurable replication, auto-failover, and online rebalancing
- **Multi-Service Design**: Independent scaling of data, query, index, search, analytics, and eventing services
- **Cross-Datacenter Replication**: Built-in XDCR for geographic distribution

1.1.4 Security and Enterprise Features

- **Access Control**: Role-Based Access Control (RBAC), LDAP integration, and operation auditing
- **Encryption**: TLS for data in transit and field-level encryption for sensitive data

1.2 Brief History

Couchbase was formed in 2010 through the merger of Membase (a key-value store with Memcached compatibility) and CouchDB technology (document database capabilities). Key milestones include:

- 2011: Release of Couchbase Server 1.0
- 2015: Introduction of N1QL, a SQL-like query language for JSON
- 2017: Launch of Couchbase Mobile for edge computing
- 2021: Couchbase goes public with IPO

The introduction of N1QL in 2015 was particularly significant, as it bridged the gap between NoSQL flexibility and SQL familiarity.

1.3 Use Case Examples

Couchbase excels in several common application scenarios:

User Profile Management: Document model for dynamic attributes, fast key-value access, and session expiration.

Product Catalogs: Flexible schema for product attributes, N1QL for advanced queries, and full-text search.

Gaming Applications: Low-latency key-value operations, document model for player/game data, and horizontal scaling.

Internet of Things: Mobile sync for edge devices, time-series support for sensor data, and schema flexibility for device diversity.

2 Which aspects of relational DBs are improved?

Couchbase addresses three fundamental limitations of relational databases: schema rigidity, scaling challenges, and query limitations for complex data structures.

2.1 Schema Flexibility

Relational limitation: Relational databases require predefined schemas where all data must conform to a fixed structure. Schema changes typically involve ALTER TABLE operations that often cause downtime, require data migrations, and necessitate application code updates.

Other NoSQL limitations:

- **MongoDB**: Requires field-level indexes specified in advance, with indexing limitations on deeply nested structures
- **Cassandra**: Uses table-like structures requiring column family definitions upfront
- **DynamoDB**: Limits secondary indexes and requires careful attribute planning

Couchbase advantage: Couchbase's document model delivers superior flexibility through:

- Schema-free JSON documents with unlimited nesting depth
- No predefined structure requirements at collection level
- Full indexing capabilities on any field or nested path
- Support for heterogeneous documents within the same collection
- Built-in schema validation when governance is needed

2.2 Scalability Improvements

Relational limitation: Traditional relational databases were designed for vertical scaling (bigger servers) rather than horizontal scaling. As data grows, this approach eventually hits hardware limits. Manual sharding requires complex application logic, and joins become problematic across distributed data.

Couchbase advantage: Couchbase provides a truly distributed architecture that surpasses both relational and other NoSQL solutions:

- **Linear Performance**: Near-linear throughput increases with additional nodes (95% efficiency vs. theoretical maximum)
- **In-Memory Optimization**: Memory-first architecture with configurable RAM-to-disk ratios
- **Intelligent Rebalancing**: Auto-weighted data redistribution during cluster changes
- **Multi-Region**: Advanced cross-datacenter replication (XDCR) with filtering and compression

2.3 Advanced Query Capabilities

Relational limitation: SQL struggles with hierarchical data structures common in modern applications. Complex nested structures must be split across multiple tables, requiring joins that become performance bottlenecks at scale. This creates a mismatch with object-oriented application code.

Couchbase solution: SQL++/N1QL (SQL for JSON) combines SQL familiarity with direct operations on nested JSON structures:

```
SELECT product.name, product.specs.cpu
FROM products AS product
WHERE "black" IN product.colors;
```

This query directly accesses nested fields and array elements without complex joins or subqueries.

2.4 Relationship to Codd's Rules

Couchbase strategically diverges from Codd's relational model while preserving key benefits and adding NoSQL advantages:

Principle	Relational Approach	MongoDB Approach	Couchbase Approach
Data Structure	Fixed tables and columns	BSON documents	JSON documents with optional schemas
Query Language	SQL	Proprietary	SQL++ (N1QL)
Transactions	ACID by default	ACID with limitations	Configurable ACID with versatile consistency
Relationships	Foreign keys and constraints	Manual reference or embedding	Flexible references with JOIN support
Schema Management	Strict schema enforcement	Schema-optional	Schema-optional with validation
Distribution Model	Complex federation	Replica sets with sharding	Unified cluster with multi-dimensional scaling

Couchbase fundamentally reimagines database architecture by combining the query power and familiarity of relational systems, the flexibility of document databases, the performance of key-value stores, and the scalability of distributed systems—creating a truly versatile data platform that eliminates the traditional tradeoffs between different database paradigms.

3 Deployment and Installation

Couchbase Server can be deployed in two different ways, depending on the user's needs and infrastructure preferences.

3.1 Cloud Deployment

The simplest way to use Couchbase Server is through Couchbase Capella, a fully managed cloud-based service. This eliminates the need for a on-premise installation and ongoing maintenance, making it an excellent choice for users who prefer a hassle-free, scalable solution.

3.1.1 Using Couchbase Capella

To get started with Couchbase Capella, follow these steps:

1. Sign up for a Couchbase Capella account by visiting the official Couchbase Capella website and following the registration instructions. [1]
2. Once the account is created, set up a new Couchbase Server cluster by following the step-by-step guide provided within the Capella interface.
3. After the cluster is successfully deployed, a sample-bucket is already added, and a connection string and credentials can be created to access the cluster.

[2]

3.2 On-premise Installation

For users who prefer greater control over their deployment, installing Couchbase Server (on-prem) is a viable option. The recommended approach for this is using Docker, which simplifies the setup process while offering flexibility. However, this method requires Docker to be installed on the machine and some level of system configuration.

3.2.1 Docker Installation

To install Couchbase Server with Docker, follow these steps:

1. Install Docker on your machine by following the official installation guide available on the Docker website.
2. Pull and run the Couchbase Server container with the following command:
   ```
   docker run -d --name db -p 8091-8097:8091-8097 -p 9123:9123 -p 11207:11207 -p 11210:11210 -p 11280:11280 -p 18091-18097:18091-18097 couchbase
   ```
3. Once the container is running, access the Couchbase Web Console by opening a web browser and navigating to `http://localhost:8091`
4. Follow the on-screen instructions to set up the Couchbase Server cluster and configure the necessary settings.

[3]

3.3 Choosing the Right Deployment Method

Selecting between Couchbase Capella and a on-premise installation depends on various factors. Thus it is essential to consider the advantages and disadvantages of each deployment method before making a decision.

Couchbase Capella, as a cloud deployment, offers a fully managed service with automated updates and backups, ensuring high availability and scalability without the need for infrastructure maintenance. However, it comes with ongoing cloud usage costs and offers less flexibility in configuration and customization.

In contrast, an on-premise installation using Docker provides full control over configuration and security settings while avoiding cloud-related costs. Yet, it requires manual maintenance, updates, and backups, is limited by local machine resources, and involves a more complex initial setup compared to Capella.

Ultimately, Capella is ideal for businesses seeking a hassle-free, scalable solution, while an on-prem installation is better suited for developers who need full control over their environment.

4 APIs and SDKs

Couchbase provides a variety of APIs and SDKs to interact with the database, making it easy to integrate Couchbase into your applications. These APIs and SDKs are available in multiple programming languages, allowing developers to work with Couchbase using their preferred language.

4.1 Couchbase SDKs

Couchbase offers official SDKs for popular programming languages, including Java, .NET, Node.js, Python, Go, and others. These SDKs provide a high-level interface to interact with Couchbase, simplifying the development process and enabling developers to focus on building their applications. [4]

4.1.1 Features of Couchbase SDKs

- Document-oriented API for storing and retrieving JSON documents.
- Key-Value operations for fast data access.
- Querying capabilities using N1QL (SQL for JSON).
- Full-text search integration.
- Eventing and Analytics support.

4.2 Couchbase REST API

In addition to the SDKs, Couchbase also provides a REST API that allows developers to interact with the database over HTTP. This API is useful for scenarios where a native SDK is not available or when integrating with other systems that support RESTful communication.

4.3 Example Code - Python

The Python SDK provides a high-level interface to interact with Couchbase, making it easy to perform CRUD operations, query data, and manage the cluster. To install the couchbase module run: 'pip install couchbase' [5]

Let's look at a code snippet that demonstrates basic CRUD operations:

Skript 4.1: Python example

```
# Sample airline document
sample_airline = {
  "type": "airline",
  "id": 8091,
  "callsign": "CBS",
  "iata": None,
  "icao": None,
  "name": "Couchbase Airways",
}
```

```
key = "airline_8091"

# Create a document with specific ID
result = collection.insert(key, sample_airline)

# Retrieve a document by ID
result = collection.get(key)

# Update a document by ID
sample_airline["name"] = "Couchbase Airways!!"
result = collection.replace(key, sample_airline)

# Delete a document by ID
result = collection.remove(key)
```

We can see that interacting with Couchbase using the couchbase-python-module is straightforward and requires minimal code to perform common database operations.

5 Advantages and Disadvantages

5.1 Technical Advantages

Performance Metrics:

- Sub-millisecond key-value operations (<1ms @ 99th percentile)
- Memory-first architecture with configurable ejection policies
- B-tree based global secondary indexes for query optimization
- Write-optimized storage engine with append-only commits
- 30-40

Technical Capabilities:

- Multi-model support: K-V, document, spatial, full-text within single platform
- ANSI JOIN and NEST operations in N1QL with pushdown optimization
- Cross Datacenter Replication (XDCR) with filtering and compression
- SSLv3/TLS 1.2+ encryption with FIPS 140-2 compliance
- SDK support for Java, .NET, Node.js, Python, Go with reactive extensions

5.2 Technical Limitations

Performance Constraints:

- Document size limit: 20MB (default)
- Memory overhead: 56 bytes metadata per document
- Transaction latency: increased by 15-30% for multi-document ACID
- Query performance degrades with >10 JOINs in single statement
- Minimum 4GB RAM recommended per node for production

Architectural Considerations:

- Default consistency: eventual for queries, strong for K-V operations
- CAP theorem positioning: CP for document operations, AP for cross-cluster
- Minimum 3 nodes recommended for high availability
- Scaling requires rebalance operations (minimal but measurable impact)