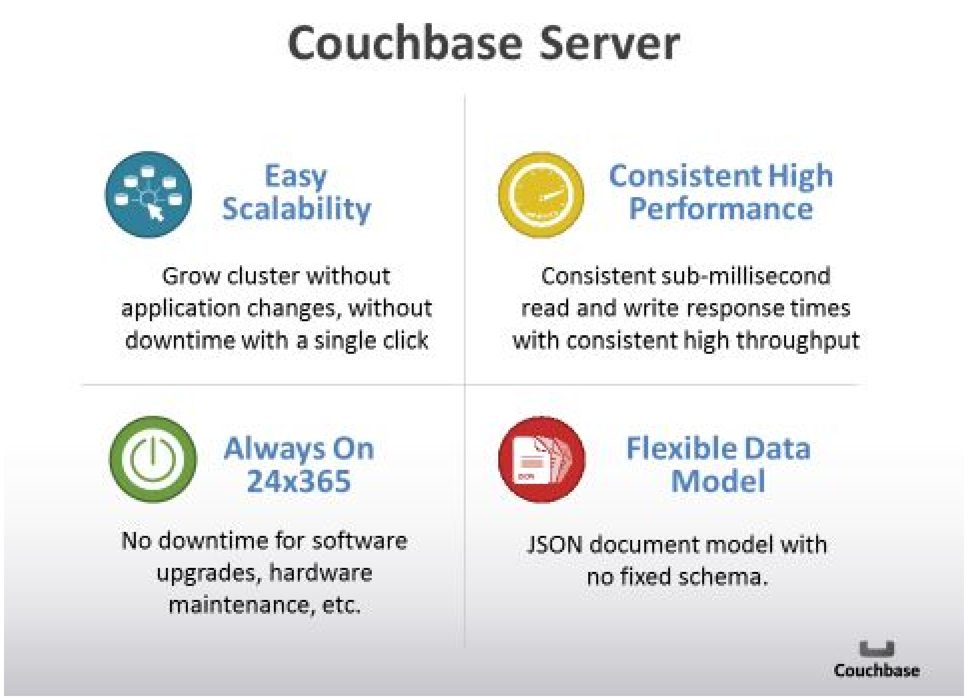
COUCHBASE

Shruti Arora
Pulkit Dhir
Shreya Goel
Varun Maheshwari

What is Couchbase?

- Distributed NoSQL document-oriented database.
- Specialized to provide low-latency data management for large-scale applications.
- Supports both key-value and document-oriented use cases
- Obtained as packaged software in both enterprise and community editions.



Why NoSQL?

- Industry after industry is shifting to the Digital Economy.

- At the heart of every Digital Economy business are its web, mobile, and Internet of Things (IoT) applications.

- Today’s web, mobile, and IoT applications share the following characteristics -
  - Support large numbers of concurrent
  - Deliver highly responsive experiences
  - Be always available
  - Handle semi- and unstructured data
  - Rapidly adapt to changing requirements

- The new enterprise technology architecture needs to be far more agile, and requires an approach to real time data management.

Why relational databases fall short?

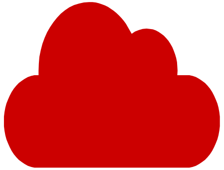
- Relational databases were born in the era of mainframes and business applications
- These databases were engineered to run on a single server
- Traditional databases don’t address the need to develop with agility and to operate at any scale
- NoSQL databases emerged as a result of the exponential growth of the Internet and the rise of web applications

Five Trends Create New Technical Challenges that NoSQL Addresses

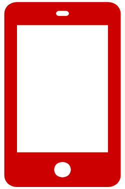
More customers are going online

The Internet is connecting everything

Big Data is getting bigger



Applications are moving to the cloud



The world has gone mobile

Power of SQL

Flexibility of JSON

Scalability of NoSQL

Features

Develop with Agility

Easier, Faster Development
Flexible Data Modeling
Powerful Querying & Indexing
Big Data Integration

Operate at Any Scale

Elastic Scalability
Always-on Availability
Consistent High Performance

Document Databases

- Each record in the database is a self-describing document
- Each document has an independent structure
- Documents can be complex
- All databases require a unique key
- Documents are stored using JSON or XML or their derivatives
- Content can be indexed and queried
- Offer auto-sharding for scaling and replication for high-availability

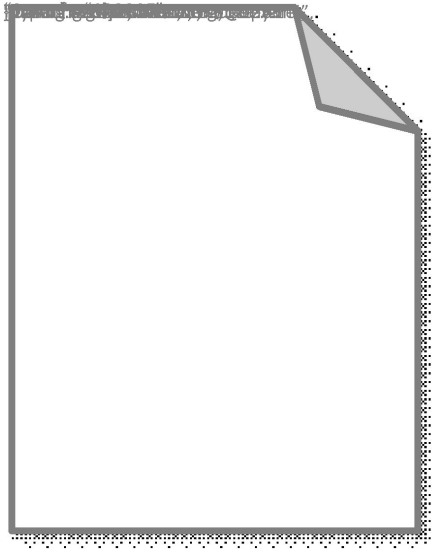


Figure 1. Relational model for flight schedules

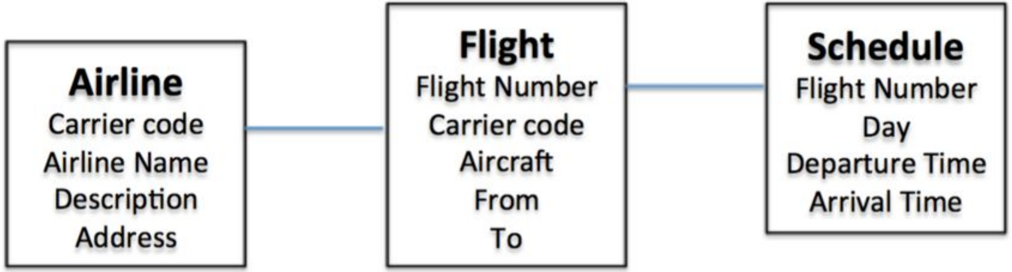
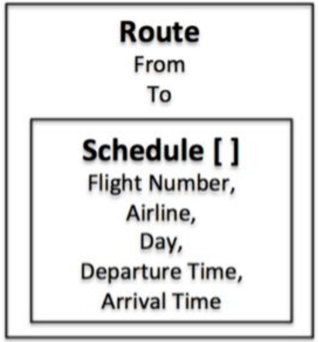


Figure 2. Document data model for flight bookings



Relational Data Model vs Document Data Model

Relational data model

Highly-structured table organization with rigidly-defined data formats and record structure.

You must define a schema before adding records to a database.

Within a table, you need to define constraints in terms of rows and named columns as well as the type of data that can be stored in each column.

Document data model

Collection of complex documents with arbitrary, nested data formats and varying “record” format.

A document-oriented database contains documents, which are records that describe the "schema" of the data in the document, as well as the actual data.

You can also use one or more documents to represent a real-world object.

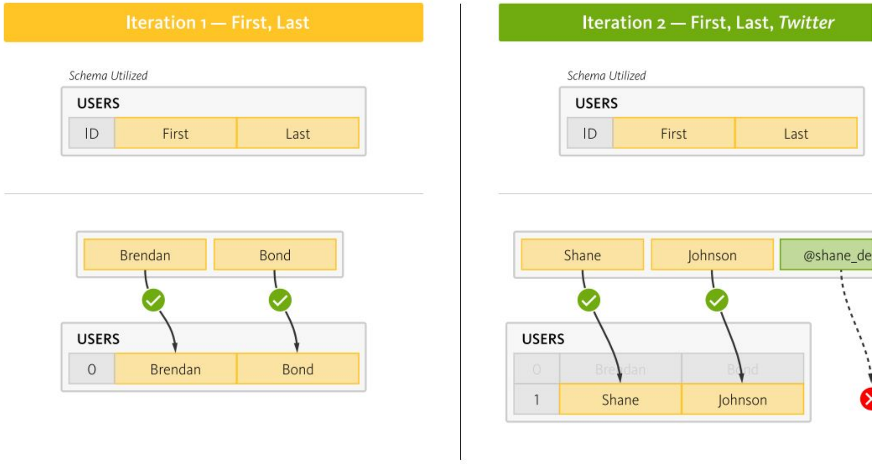


Figure 1: *RDBMS – An explicit schema prevents the addition of new attributes on demand*

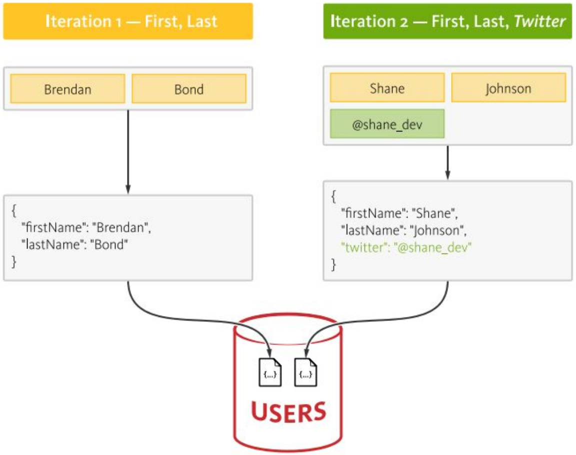


Figure 2: *JSON – The data model evolves as new attributes are added on demand.*

Shane	Johnson	Big Data	Product Marketing	Couchbase
Shane	Johnson	Big Data	Technical Marketing	Red Hat
Shane	Johnson	Java	Product Marketing	Couchbase
Shane	Johnson	Java	Technical Marketing	Red Hat
Shane	Johnson	NoSQL	Product Marketing	Couchbase
Shane	Johnson	NoSQL	Technical Marketing	Red Hat

Figure 4: *RMDBS – Queries return duplicate data, applications have to filter it out.*

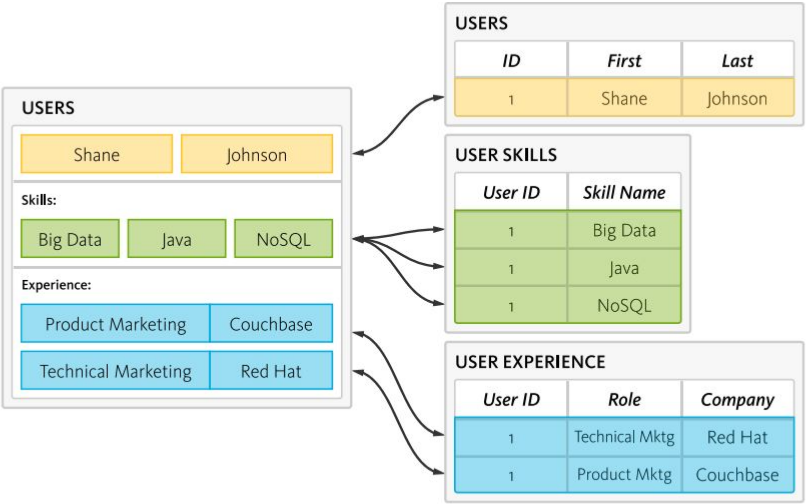


Figure 3: *RDBMS – Applications "shred" objects into rows of data stored in multiple tables.*

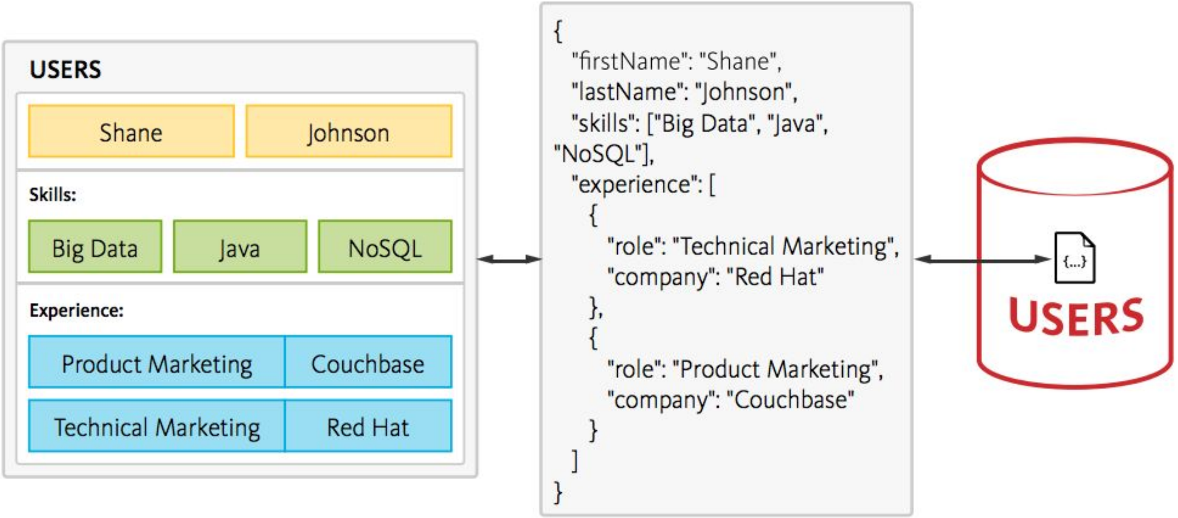


Figure 5: *JSON – Applications can store objects with nested data as single documents.*

Example: User Profile

User Info

KEY	First	Last	ZIP_id
1	Dipti	Borkar	2
2	Joe	Smith	2
3	Ali	Dodson	2
4	John	Doe	3

Address Info

ZIP_id	CITY	STATE	ZIP
1	DEN	CO	30303
2	MV	CA	94040
3	CHI	IL	60609
4	NY	NY	10010

To get information about specific user, you perform a join across two tables

Document Example: User Profile

```
{
   "ID": 1,
   "FIRST": "Dipti",
   "LAST": "Borkar",
   "ZIP": "94040",
   "CITY": "MV",
   "STATE": "CA"
}
```

JSON

=

User Info

KEY	First	Last	ZIP_id
1	Dipti	Borkar	2
2	Joe	Smith	2
3	Baxter	Dodson	2
4	Lari	Gorin	3

\+

Geo Info

ZIP_id	CITY	STATE	ZIP
1	DEN	CO	30303
2	MV	CA	94040
3	CHI	IL	60609
4	NY	NY	10010

All data in a single document

Making a Change Using RDBMS

User Table

User ID	First	Last	Zip	Country ID
1	Dipti	Borkar	94040	001
2	Joe	Smith	94040	001
3	Ali	Dodson	94040	001
4	Sarah	Gorin	NW1	002
5	Bob	Young	30303	001
6	Nancy	Baker	10010	001
7	Ray	Jones	31311	001
8	Lee	Chen	V5V3M	008
. . .				. . .
50000	Doug	Moore	04252	001
50001	Mary	White	SW195	002
50002	Lisa	Clark	12425	001

Photo Table

User ID	Photo ID	Comment	Country ID
2	d043	NYC	001
2	b054	Bday	007
5	c036	Miami	001
7	d072	Sunset	133
5002	e086	Spain	133

Status Table

User ID	Status ID	Text	Country ID
1	a42	At conf	134
4	b26	excited	007
5	c32	hockey	008
12	d83	Go A's	001
5000	e34	sailing	005

Affiliations Table

User ID	Affl ID	Affl Name	Country ID
2	a42	Cal	001
4	b96	USC	001
7	c14	UW	001
8	e22	Oxford	002

Country Table

Country ID	Country name
001	USA
002	UK
003	Argentina
004	Australia
005	Aruba
006	Austria
007	Brazil
008	Canada
009	Chile
. . .	
130	Portugal
131	Romania
132	Russia
133	Spain
134	Sweden

Making the Same Change with a Document Database

Just add information to a document

Logical Data Modeling

The logical data modeling phase focuses on describing your entities and relationships. It is done independently of the requirements and facilities of the underlying database platform.

The key definitions you need from your logical data modeling exercise:

- **Entity keys**
- **Entity attributes**
- **Entity relationships**

Physical Data Modeling

The physical data model takes the logical data model and maps the entities and relationships to physical containers.

Table 1. Data representation and containment in Couchbase Server versus relational databases

Couchbase Server	Relational databases
Buckets	Databases
Buckets or Items (with type designator attribute)	Tables
Items (key-value or document)	Rows
Index	Index

Items

Items consist of a key and a value. A key is a unique identifier within the bucket. Value can be a binary or a JSON document. You can mix binary and JSON values inside a bucket.

Keys

Each value (binary or JSON) is identified by a unique key. Keys are immutable. Thus, if you use composite or compound keys, ensure that you use attributes that don't change over time.

Values

- **Binary values**: Binary values can be used for high performance access to compact data through keys.
- **JSON values**: JSON provides rich representation for entities. Couchbase Server can parse, index and query JSON values. JSON provide a name and a value for each attribute.

Buckets

Couchbase Server also provides a container called a bucket to group items. Buckets are primarily used to control resource allocation and to define security and storage properties.

JavaScript Object Notation (JSON)

It is a lightweight data-interchange format which is easy to read and change. JSON is language-independent although it uses similar constructs to JavaScript.

JSON supports the following basic data types:

- Numbers, including integer and floating point
- Strings, including all unicode characters and backslash escape characters
- Boolean: true or false
- Arrays, enclosed in square brackets: ["one", "two", "three"]
- Objects, consisting of key-value pairs, and also known as an associative array or hash. The key must be a string and the value can be any supported JSON data type.

Data access

The ways to access the data:

1. Key value access pattern
2. Querying data
   - MapReduce
   - N1QL
   - Full text search (FTS)

N1QL

- A declarative query language that extends SQL for JSON.
- N1QL enables you to query JSON documents without any limitations - sort, filter, transform, group, and combine data with a single query.
- No longer limited to “single table” and “table per query” data models.

N1QL vs SQL

SQL STATEMENT

```sql
SELECT name, author
FROM books
```

N1QL STATEMENT

```sql
SELECT name, author
FROM books
```

SQL RESULTS (ROWS)

name	author
Ender's Game	Orson Scott Card
Foundation	Isaac Asimov
Neuromancer	William Gibson
Consider Phlebas	Iain M. Banks
Revelation Space	Alastair Reynolds
...	...

N1QL RESULTS (DOCUMENT)


```
{
  "results": [
    {"name": "Ender's Game", "author": "Orson Scott Card
    {"name": "Foundation", "author": "Isaac Asimov"},
    {"name": "Neuromancer", "author": "William Gibson"},
    {"name": "Consider Phlebas", "author": "Iain M. Bank
    {"name": "Revelation Space", "author": "Alastair Rey
    ...
  ]
}
```

Indexing

- An index is a data-structure that provides quick and efficient means to query and access data, that would otherwise require scanning a lot more documents.
- Couchbase Server speeds up data access with indexes.
- Couchbase provides both local and global indexes.

Types of indexes

1. Composite Indexes
2. Covering Indexes
3. Filtered Indexes
4. Function-based Indexes
5. Sub-document Indexes
6. Incremental mapreduce views
7. Spatial Views
8. Full-text Indexes

Data management

- Atomicity properties
- Strong consistency and durability
- Consistency of indexes and Replicas
- Tunable durability Requirements
- Concurrency
- Document Expiration

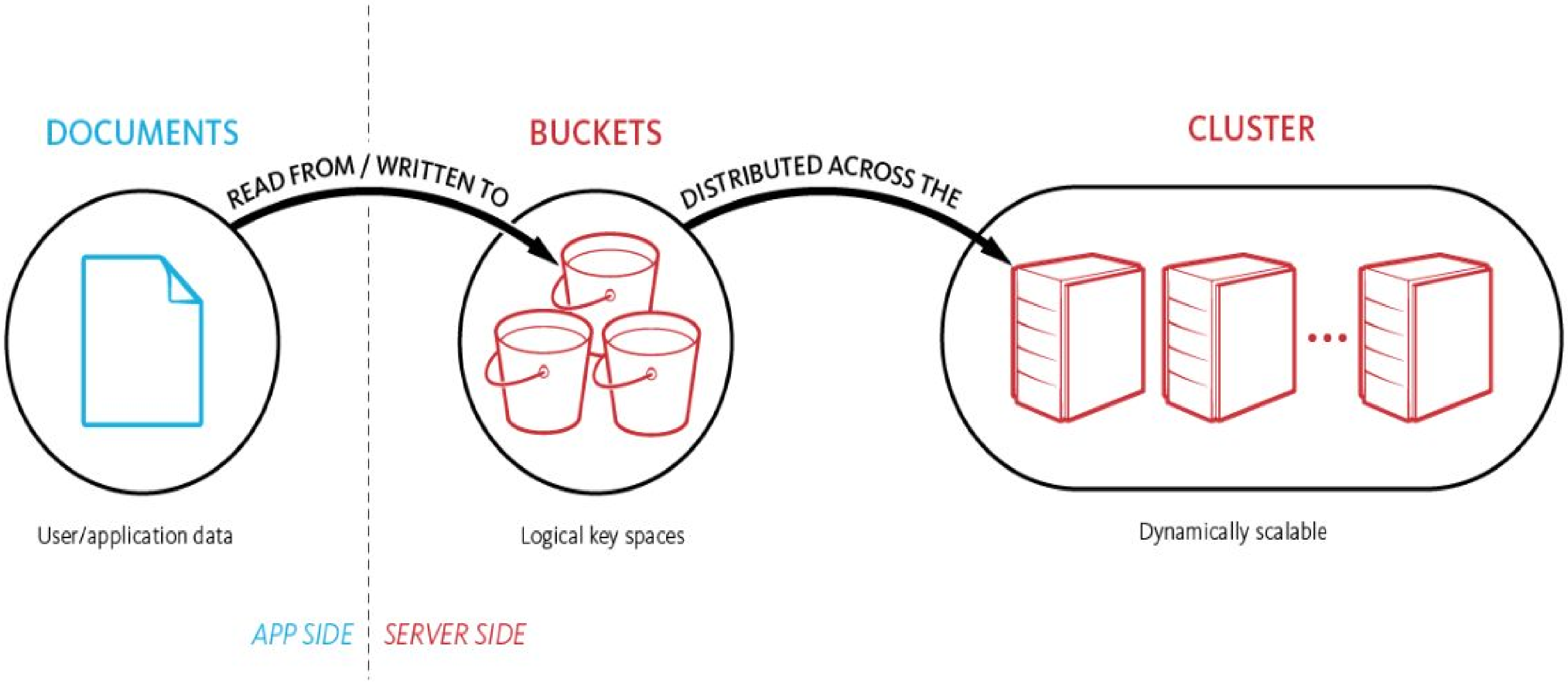
Distributed data management

- Couchbase Server is a distributed system that is built from the ground up for easy scale out and management.
- Couchbase Server has a peer-to-peer topology and all the nodes are equal and communicate to each other on demand.

Multidimensional scaling

- MDS enables users to turn on or off specific services on each Couchbase Server node so that the node in effect becomes specialized to handle a specific workload
- Advantages:
  - Independent Services
  - Quick and efficient
  - Customize machines
  - Workload isolation

BUCKETS vs vbuckets



- Data Change protocol
  - Data Change Protocol (DCP) is a high-performance streaming protocol that communicates the state of the data using an ordered change log with sequence numbers

- Replication
  - creates copies of active data, distributes those replicas across the nodes in the cluster, ensuring that every copy is located on a separate node, and then continues to maintain the replicas over time.

Availability

Couchbase Server delivers key high availability features such as zero downtime administration and maintenance, built-in data redundancy, and automatic failover.

Factors that increase system uptime and availability include:

- Number of replicas
- Number of racks or availability zones
- Number of clusters

Cross Datacenter Replication (XDCR)

It is used to replicate data between clusters in different data centers and geographic regions, and can also be used to sync a second Couchbase Server cluster within the same data center.

XDCR serves an important role in high availability / disaster recovery, performance, and load distribution.

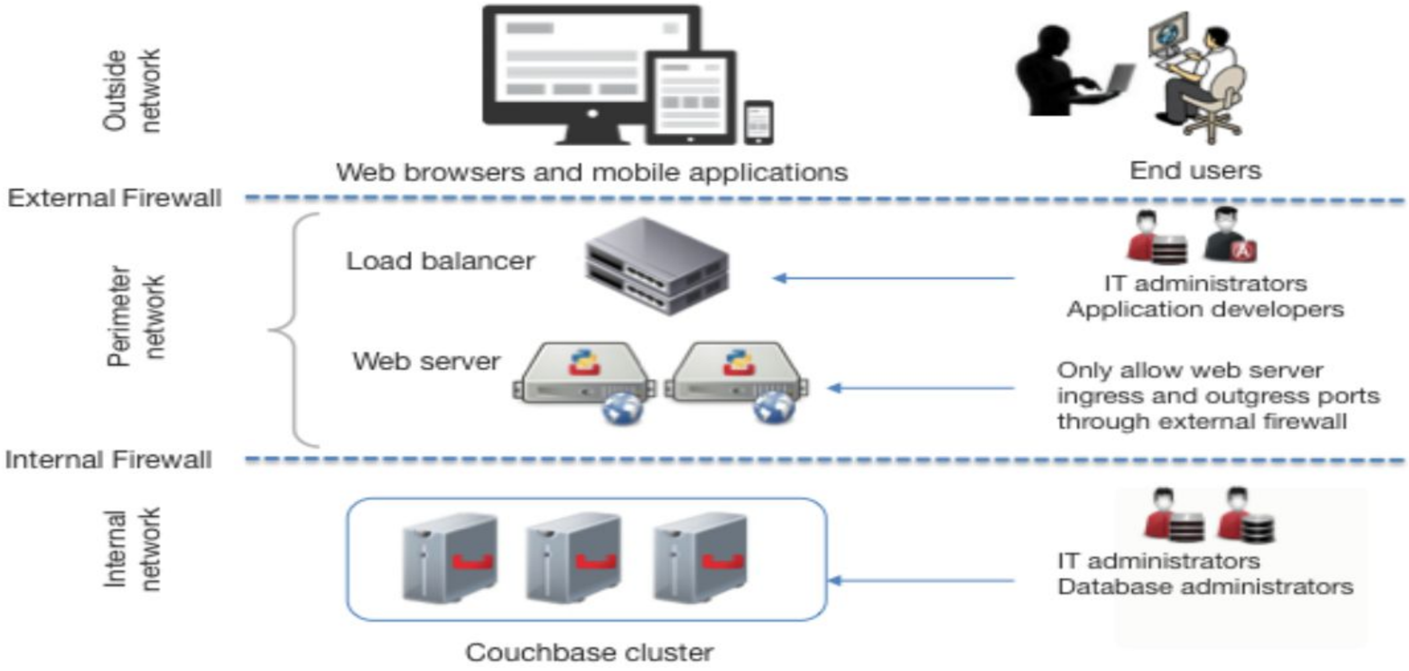
- For disaster recovery, one or more clusters can act as hot standbys, enabling cluster-level failover by taking over load as soon as a cluster stops responding.
- In case of serious failures, XDCR can also be used to recover data from a remote cluster. The result is similar to recovery using a backup but often faster.
- In geographically distributed data centers, XDCR can improve performance by placing data close to end users.

Security

Couchbase offers security mechanisms that help protect against threats and breaches.

1. Authentication and Authorization
2. Encryption
3. Auditing

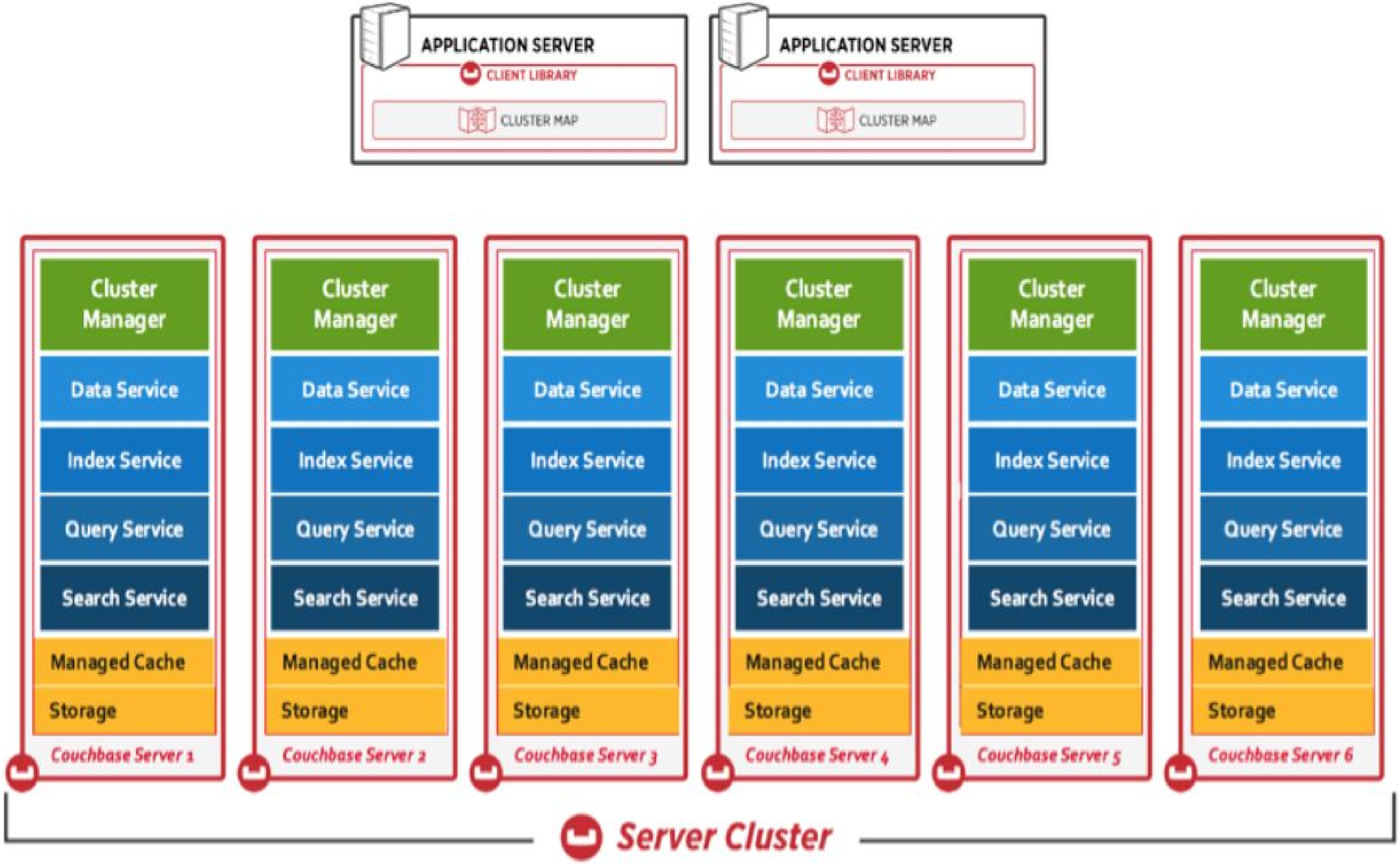
Security


Outside network
Web browsers and mobile applications
End users
External Firewall
Perimeter network
Load balancer
IT administrators
Application developers
Web server
Only allow web server ingress and outgress ports through external firewall
Internal Firewall
Internal network
IT administrators
Database administrators
Couchbase cluster

Architecture overview

❖The core architecture is designed to simplify building modern applications with a flexible data model and simpler high availability, high scalability, high performance, and advanced security.

❖The applications connect to a Couchbase Server cluster to perform read and write operations, and run queries with low latencies (sub millisecond) and high throughput (millions of operations per second).



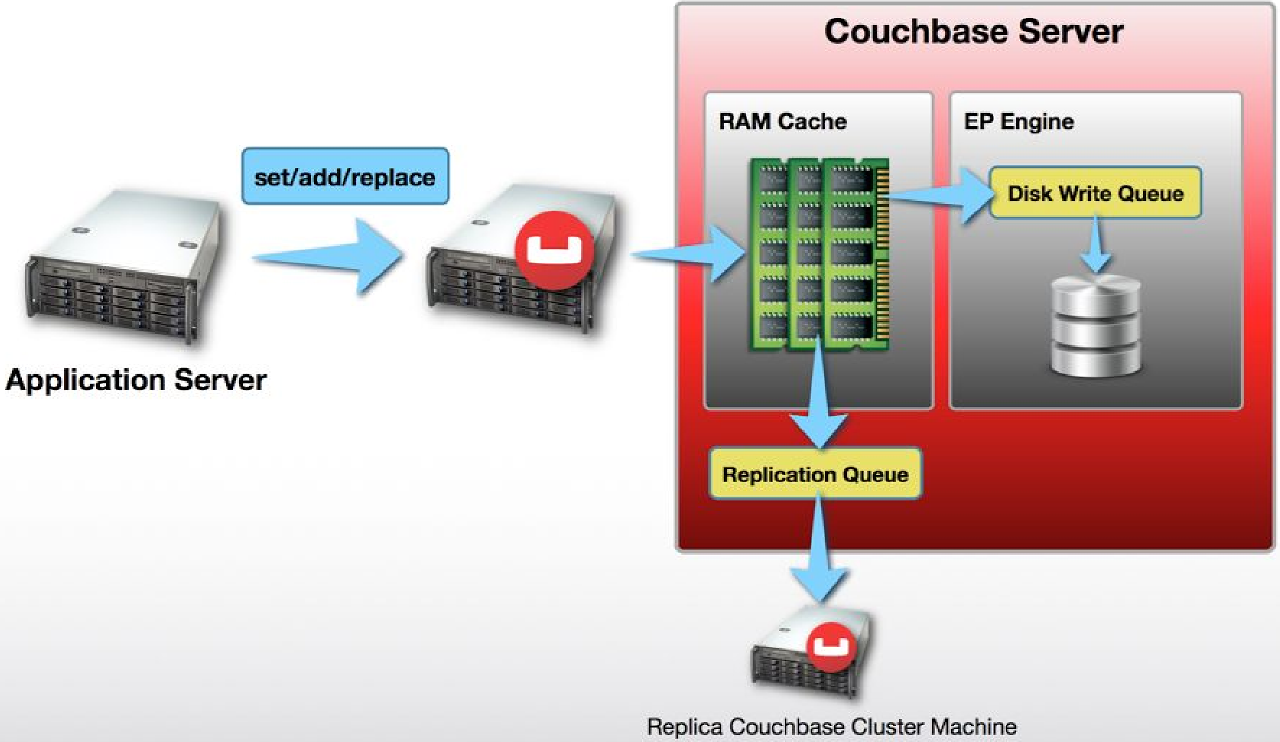
APPLICATION SERVER
CLIENT LIBRARY
CLUSTER MAP
APPLICATION SERVER
CLIENT LIBRARY
CLUSTER MAP
Cluster Manager
Data Service
Index Service
Query Service
Search Service
Managed Cache
Storage
Couchbase Server 1
Couchbase Server 2
Couchbase Server 3
Couchbase Server 4
Couchbase Server 5
Couchbase Server 6
Server Cluster

Terminology

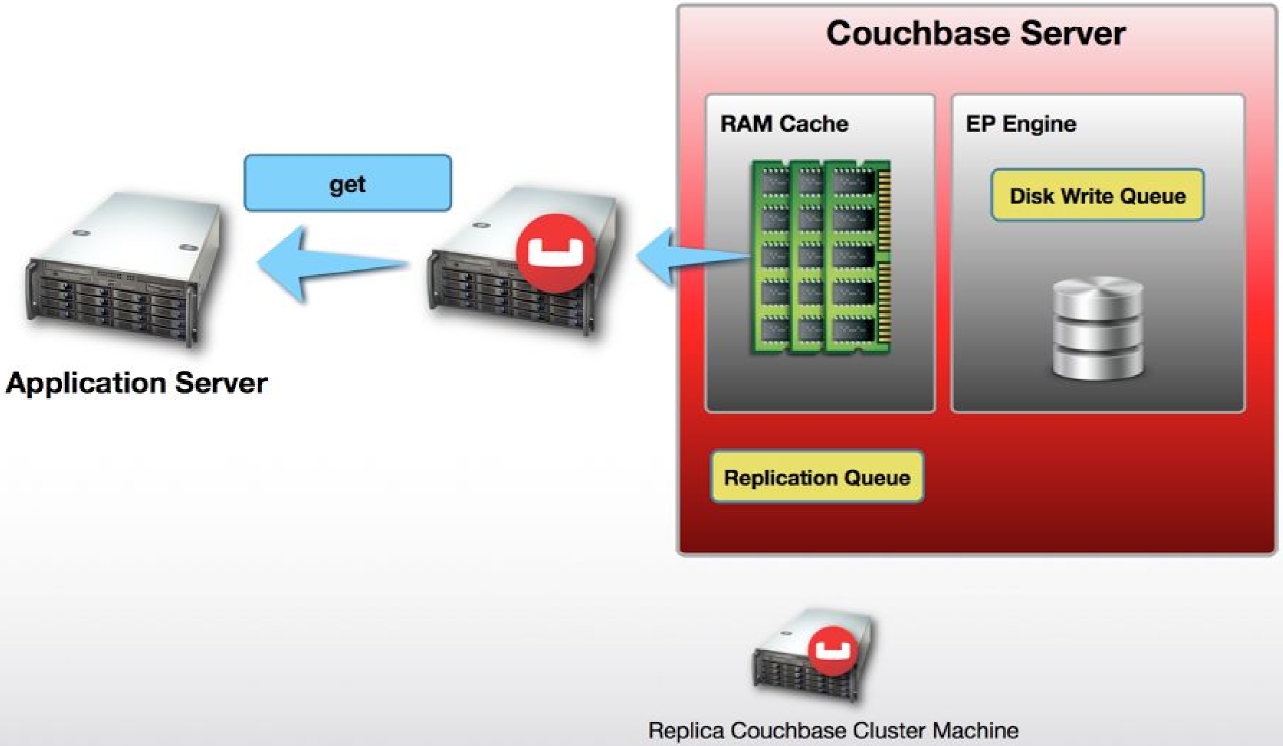
The key terms and concepts used in the Couchbase Server architecture :

1. Node
2. Cluster
3. Bucket
4. Item
5. vBucket
6. Cluster map
7. vBucket map
	8. Replication
	9. Rebalance
	10. Failover
	    a. Graceful Failover
	    b. Hard Failover
	    c. Automatic Failover
	11. Node Lifecycle

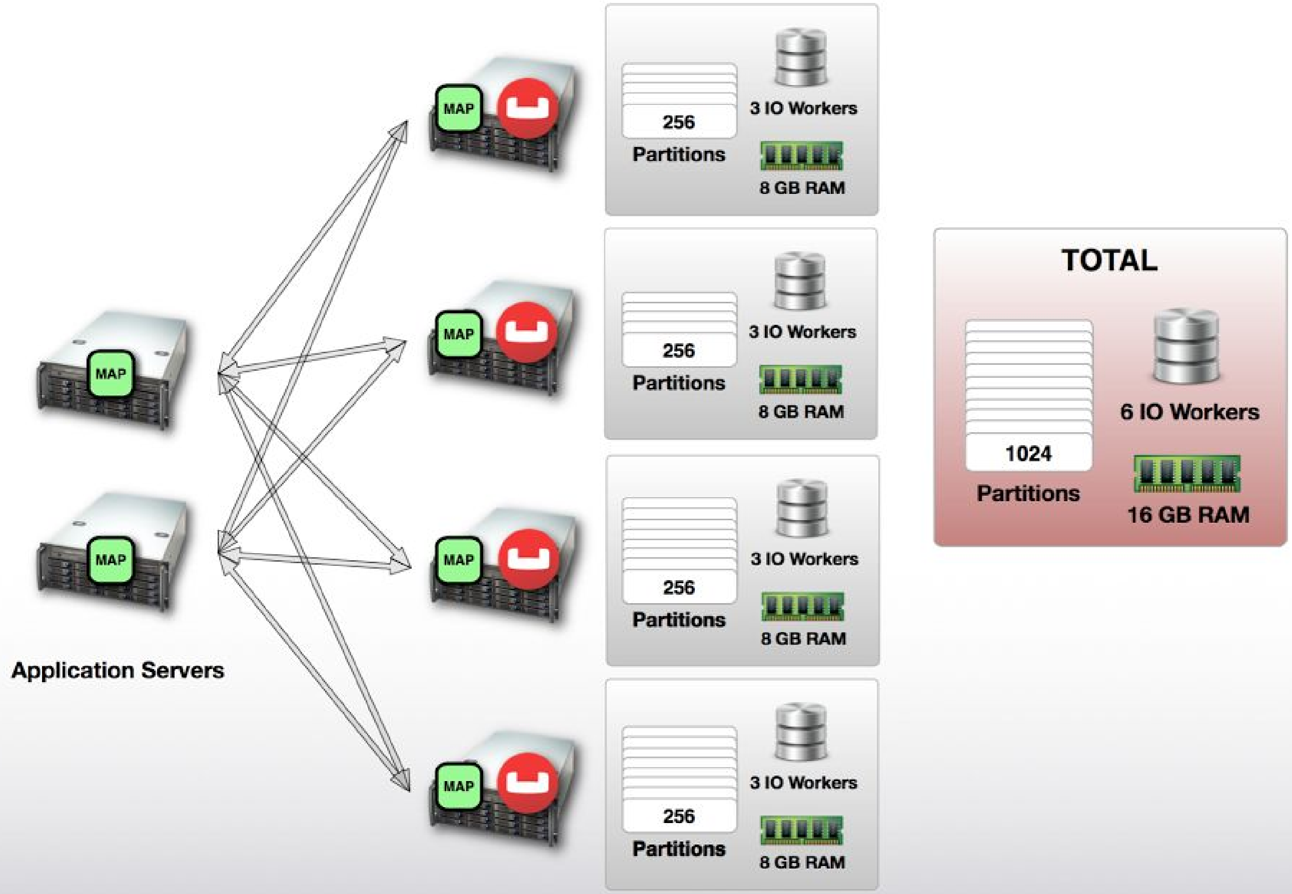
Storage Operations



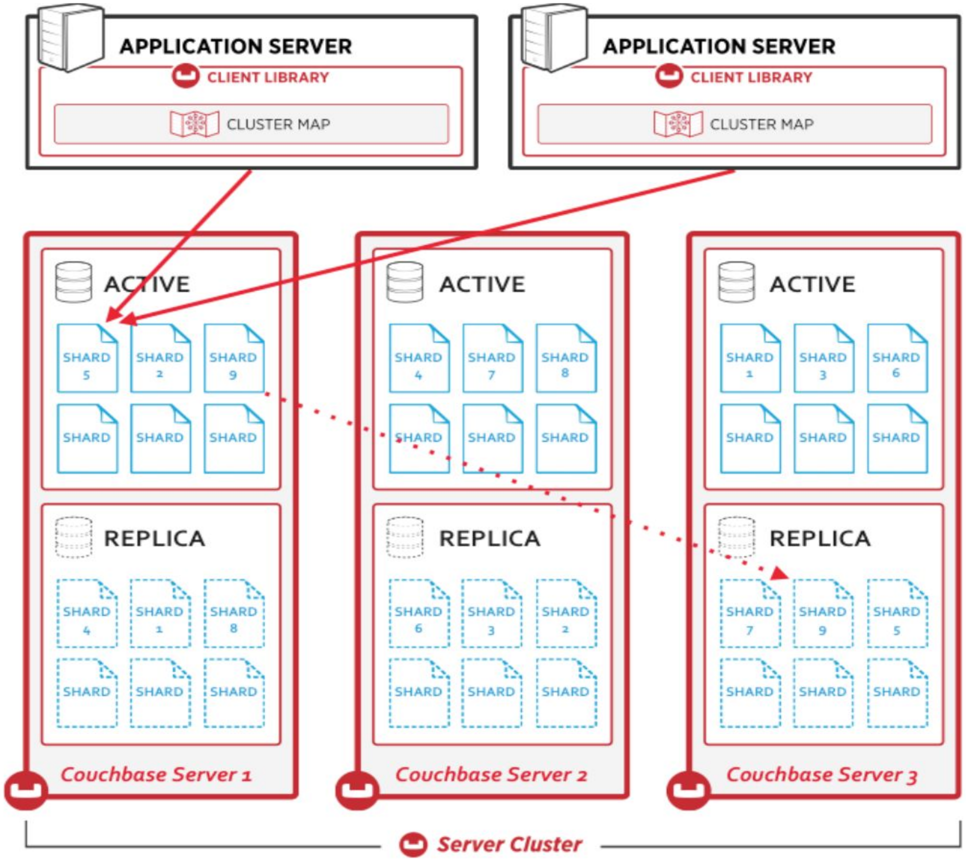
Retrieval Operations



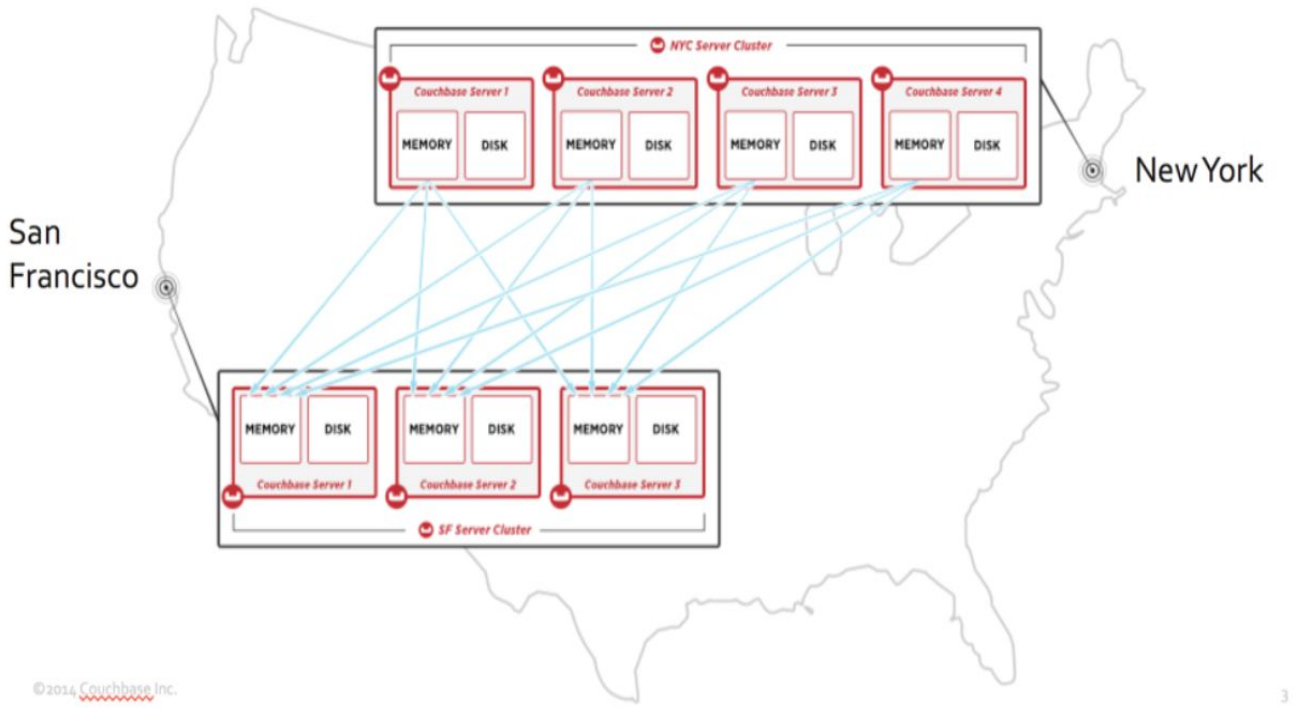
Horizontal Scale-Rebalance



Intra-cluster replication



Cross data center replication



Storage architecture

- Couchbase uses multiple storage engines:
  - Data Service,
  - MapReduce Views,
  - Spatial Views,
  - Couchstore
  - Index Service,
  - Search Service, and
  - ForestDB

Caching layer

- Data service uses a managed cache
- Index and Search services manage the cache
- Query service manages memory to calculate query responses

Ram quotas

- RAM quota allocation is governed through individual services. Each service in Couchbase Server tunes its caching based on its needs.

- Services that use RAM quotas:
  - Data service
  - Index and search service
  - Query service

Querying data & query data service

- Retrieving data with document key
- Querying data using View queries
- Querying data using Spatial queries
- Querying data using N1QL queries

Use Cases

Real-Time Big Data

Leverage streaming integration with Hadoop and Storm to support and enable real-time analytics.

Mobile Applications

Build mobile apps with offline support via an embedded database and automatic synchronization.

Digital Communication

Support real-time interaction and communication with low latency read/write access to messages.

Customer 360° View

Aggregate customer information from multiple sources with different data models.

Customers

LinkedIn Monitors Massive Data with Couchbase. Couchbase Server provides the scalability and performance the site engineering team needs to power its metric visualization engine

General Electric set out to bring together device connectivity, data integration and management, data analytics, cloud, and mobility all in a way that works seamlessly together and intuitively for all the members of its business.

Marriott decided it was time to replace its legacy infrastructure to better compete in the Digital Economy
The company evaluated several NoSQL solutions before deciding to switch to Couchbase.

Couchbase VS MongoDB

- Concurrency - Couchbase Server was able to handle over 3x as many concurrent clients as MongoDB.
- Throughput - Couchbase Server was able to provide 2.5x the throughput of MongoDB.
- Latency - Couchbase Server was able to provide 4-5x lower latency than MongoDB.
- Price / Performance Ratio - The cost per operation for Couchbase Server would be 22-40% of that for MongoDB.

Couchbase VS Cassandra

Couchbase Server Outperforms Cassandra by 6X on Google Cloud Platform

Results of the Google Cloud benchmark are summarized in the following table:

	COUCHBASE	CASSANDRA	COUCHBASE ADVANTAGE
Throughput	1.1 million writes/sec	1 million writes/sec	-
Latency	27ms	23ms	-
Nodes	50 @ 16 cores	300 @ 8 cores	6x
Total Cores	800	2400	3x
Price/Hour	$56	$330	6x
Price per 10K Transactions	$0.51 / 10K ops	$3.30 / 10K ops	6x

Thank you