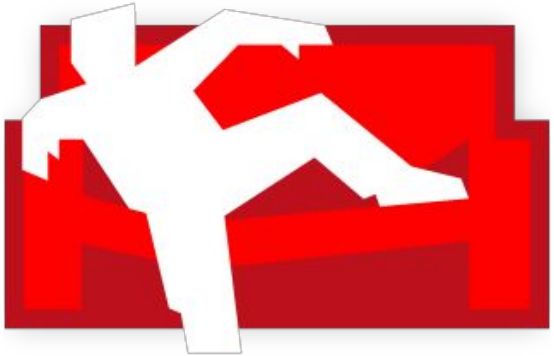
Get started with Couchbase



For Java Developer

By Akshathkumar Shetty

Big Data ?

Large volume of data
(mb,gb,tb,petabytes)

Needs quick processing
(real time)

Unstructured data
(text, images, video, tweets, gifs)

What to do with BigData

Analytical Use:

Batch processing large volume (Hadoop)

Operational Use:

NoSQL DB for real time use (Couchbase, MongoDB)

Why not RDBMS ?

Does not scale for cloud scale

Inflexible structure

Scales Up only i.e., vertically

Is disk first (so slow)

Is table oriented (1970s design when HDD was expensive)

Workaround: Cache at app/db, sharding, blobs, KV stores

Is NoSQL an option ?

Scale out for cloud scale

Flexible structure.. store JSON directly or any object

Is memory first (so fast)

Cache is built in

Replication and failover fully managed

Supports native Key Value storage

Whats NoSQL DB?

- Key Value storage
- Extended to document storage
- Column Family
- Graph Based

Which SQL ?

CAP ?

CP Based: Couchbase, MogoDB

AP Based: HBase, Cassandra

CA Based: MySQL, Oracle, MSSQL

Why Couchbase ?

- High Availability Cache
- Key Value Store
- Document Store with search, index and map reduce
- Simple and easy to manage
- SQL (N1QL) queryable JSON data
- High Performance access
- Online upgrades including indexes and s/w upgrades
- True cluster support, memory 2 memory replication, out of box cross datacenter replication systems

Couchbase.. A Bit more

- Started as Memcached
- Became Membase (persisted Memcached n KV store)
- Then Couchbase (operational NoSQL DB)
- Data stored as Key Value pair or Key with Document Value
- Key: up to 250 byte string
- Value: any value max 20mb
- Key must be unique to bucket
- Metadata are stored for documents: CAS, TTL, (sdk specific flags like type)
- Cluster -> Bucket -> Node -> Document

Who uses Couchbase ?

Trusted by the World's Biggest Brands

ebay LinkedIn nielsen PayPal SGN

zynga

How fast ?

Benchmark: MongoDB 3.2 vs. Couchbase Server 4.5 for Query and Read/Write Performance

How do the latest releases of two leading NoSQL databases compare on both read/write and query performance? The emerging technologies thought leader Avalon Consulting, LLC benchmarked MongoDB 3.2 and Couchbase Server 4.5 to find out. These big data experts ran industry standard (YCSB) workloads for both read/write and query.

The bottom line: Couchbase outperformed MongoDB by 7x on reads, 5x on writes, and 3x on queries.

The benchmark tested how well MongoDB and Couchbase Server performed with:

- A mixed read/write workload and a query workload
- 150 million documents (300GB of data)
- Insufficient memory to cache all the data (only 54% in memory)
- Up to 280 concurrent clients

* First Name

* Last Name

* Email Address

* Company

* Phone

How to access data?

- Via key (very fast)
- MapReduce View (Dist. Secondary Index)
- N1QL `Nickel` - A SQL for documents (via GSI or DSI)

Arch ?

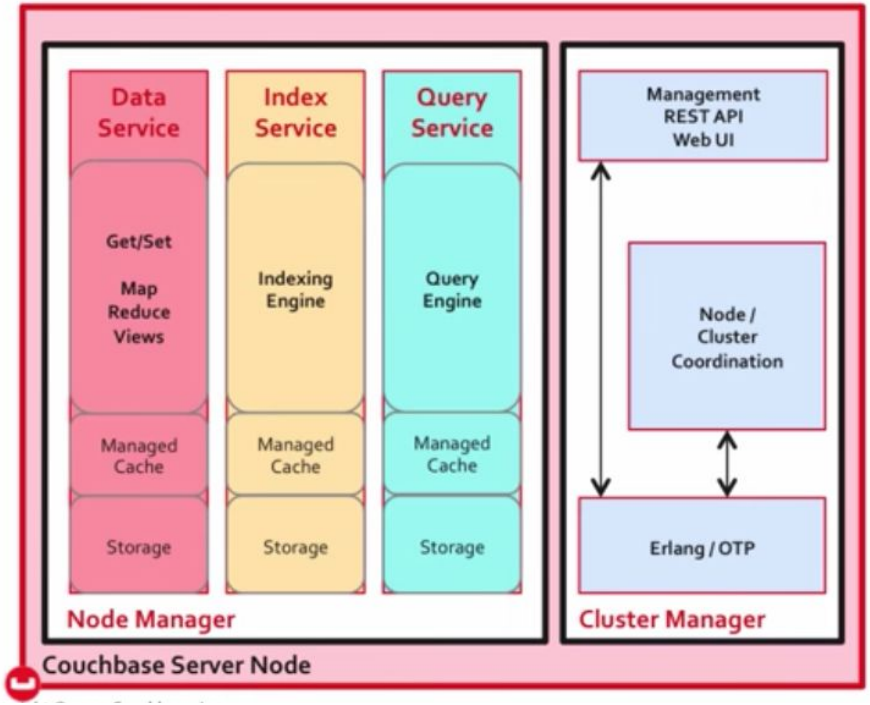
Couchbase Server nodes are identical

Two core components

- ✓ Cluster Manager
- ✓ Node Manager

Three independent services

- ✓ Data Service
- ✓ Index Service
- ✓ Query Service



How does Store & Read Document work

---

https://youtu.be/tTQHDvG4mU4

JSON in Couchbase SDK

String

`java.lang.String`

Raw JSON received via HTTP or other API

JsonObject

`com.couchbase.client.java.document.json.JsonObject`

Behaves like a serializable JSON-specific Map

JsonDocument (Document)

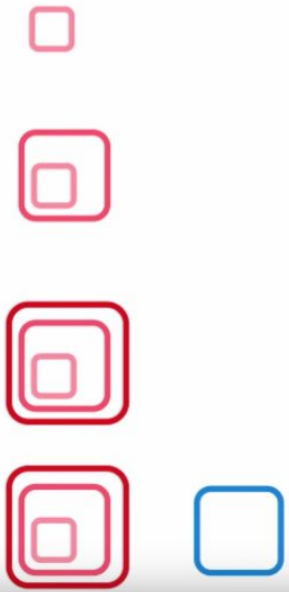
`com.couchbase.client.java.document.JsonDocument`

Canonical Couchbase JSON representation,
with metadata, interchangeable across SDKs

Corresponding local POJO

`com.couchbase.customer360.domain.Entity`

Data as represented in your own application

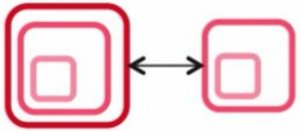


SDK Methods for JSON

- ✓ Encode JSON *String* as *JsonObject*
- ✓ Decode *JsonObject* to JSON *String*

```
com.couchbase.client.java.transcoder;
JsonObject jsonObject = transcoder.stringToJsonObject(jsonString);
String jsonString = transcoder.jsonObjectToString(jsonObject);
// or ...
String jsonString = jsonObject.toString();
```

- ✓ Create *JsonDocument* from *JsonObject*
- ✓ Extract *JsonObject* from *JsonDocument*



```
com.couchbase.client.java.document.JsonDocument;
JsonDocument jsonDocument = JsonDocument.create(id, jsonObject);
JsonObject jsonObject = jsonDocument.content();
```

Reading using Java

get (id, [timeout])

- ✓ retrieves a document by key
- ✓ accepts *String* or *Document*
- ✓ returns *JsonDocument* or *null*
- ✓ throws *CouchbaseException*

Alternatives

getAndLock() – get, then change CAS

getAndTouch() – get, then update TTL

getFromReplica() – get replica, stale?

Inserting using Java

public class **JsonDocument**
extends AbstractDocument<JsonObject>
implements Serializable

static **create** (*ID*, [expiry], content, [CAS])
- ✓ factory for *JsonDocument* creation
- ✓ ID assigned to document as metadata
- ✓ CAS may also be assigned (persisted)

ID structure is arbitrary, up to 256 bytes
Counter ID pattern discussed ahead
CAS for optimistic locking discussed ahead

```
protected Entity JsonDocument toJsonDoc(Entity source) {
  String id = source.getId();
  try {
    String s = converter.toJson(source);
    JsonObject content = transcoder.stringToJsonObject(s);
    JsonDocument doc =
      JsonDocument.create(id, content, source.getCas());
    return doc;
    } catch (Exception e) {
      throw new RepositoryException(e);
    }
}
```

Inserting options

How do you "create" a document in the bucket?

public class **CouchbaseBucket**
extends Object
implements **Bucket**

insert (*Document*, [persistence], [replicas], [timeout])

- ✓ inserts a *Document*
- ✓ may set disk *ReplicateTo* contraints (1, 2, or 3 additional replicas)
- ✓ may set *PersistTo* constraints, else acknowledged from cache
- ✓ may set *TimeUnit* timeout, else default
- ✓ returns *Document* with updated CAS

```
public <T extends Entity> T create(T entity,
  Class<? extends T> type) {
    JsonDocument docIn = toJsonDoc(entity);
    JsonDocument docOut;
    try {
      docOut = bucket.insert(docIn);
    } catch (CouchbaseException e) {
      throw new RepositoryException(e);
    }
    return fromJsonDoc(docOut, type);
}
```

HelloWorld

```
>gradle init --type java-library

>gradle

#add compile 'com.couchbase.client:java-client:2.3.5'
```

Installation

---

www.couchbase.com

Download Couchbase Community Server 4.5.0 or higher

Install DB

```
wget http://packages.couchbase.com/releases/4.5.0/couchbase-server-community_4.5.0-ubuntu14.04_amd64.deb

sudo apt-get install python-minimal

or

sudo apt-get install python3


sudo dpkg -i couchbase-server-community_4.5.0-ubuntu14.04_amd64.deb


URL: http://IP:8091
```

Ref

- http://learn.couchbase.com/
  - CB020
  - CB030
  - CB130j
-