From 0 to syncing in 30 min

Matthew Revell @matthewrevell
Philipp Fehre @ischi

Couchbase
Introduction Couchbase
Couchbase 101

- Distributed key-value store
- Create secondary indexed views using Map Reduce
- Automatic key-based partitioning
- Integrated memcached caching layer
- Strongly consistent within a cluster
- Cross-data centre replication
- Intelligent SDKs
Couchbase Organization

- Couchbase operates like a Key-Value Document Store
- Key is a UTF-8 string up to 256 Bytes
- Values can be:
  - **Simple Datatypes**: strings, numbers, datetime, boolean, and binary data can be stored -- they are stored as Base64 encoded strings
  - **Complex Datatypes**: dictionaries/hashes, arrays/lists, can be stored in JSON format (simple lists can be string based with delimiter)
  - JSON is a *special class of string* with a specific format for encoding simple and complex data structures
- Schema is unenforced and implicit, schema changes are programmatic, done online, and can vary from Document to Document
Couchbase Server 2.0 Architecture

- **Query API**: 8092
- **Memcapable 1.0**: 11211
- **Memcapable 2.0**: 11210

**RAM Cache, Indexing & Persistence Management (C)**
- Couchbase EP Engine
- Disk Persistence

**Server/Cluster Management & Communication (Erlang)**
- REST management API/Web UI
- HTTP 8091
- Erlang/OTP
- Erlang port mapper 4369
- Distributed Erlang 21100 - 21199

**Components**
- Couchbase EP Engine
- Object-level Cache
- Moxi
- Storage interface
- Query Engine
- HTTP 8091
- Erlang port mapper 4369
- Distributed Erlang 21100 - 21199
- RAM Cache, Indexing & Persistence Management (C)
- Disk Persistence
- Object-level Cache
- Moxi
- Storage interface
- Query Engine
- HTTP 8091
- Erlang port mapper 4369
- Distributed Erlang 21100 - 21199
Supported SDK's

www.couchbase.com/communities

- Each supported SDK page has instructions for setup
- PHP, Ruby, and Python clients are wrappers around libcouchbase C library, so libcouchbase must be installed *first*
- For other community clients, click on "All Clients" on left nav, scroll down the page and you can see clients for Go, Erlang, Clojure, TCL, Perl and others.
# Easy Scalability

## FEATURES

<table>
<thead>
<tr>
<th>Auto Sharding</th>
<th>XDCR</th>
<th>Single Node Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Manual Sharding</td>
<td>Database handles propagation of updates to scale across clusters and geos. Provides disaster recover / data locality</td>
<td>Hugely simplifies management of clusters. Easy to scale clusters by adding any # of nodes</td>
</tr>
</tbody>
</table>
### Consistent, High Performance

#### FEATURES

<table>
<thead>
<tr>
<th>Feature</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Massive Concurrent Connections</td>
<td>Support a large number of users needed for interactive apps</td>
</tr>
<tr>
<td>Fine Grained Locking</td>
<td>Allows high concurrency and in turn high throughput via highly granular latches</td>
</tr>
<tr>
<td>Built-in Cache</td>
<td>No need of separate cache layer</td>
</tr>
<tr>
<td>Hash Partitioning</td>
<td>Uniform data distribution</td>
</tr>
<tr>
<td></td>
<td>Uniform load distribution - NO hotspots</td>
</tr>
</tbody>
</table>

**Performance:**

- **PEERFRRANCE**
- **P** - Support a large number of users needed for interactive apps
- **E** - Fine Grained Locking
- **R** - Built-in Cache
- **F** - Hash Partitioning
- **R** - Massive Concurrent Connections
- **M** - Performance

**Support a large number of users needed for interactive apps**

**Fine Grained Locking**

- Allows high concurrency and in turn high throughput via highly granular latches

**Built-in Cache**

- No need of separate cache layer
- Database manages actively used data

**Hash Partitioning**

- Uniform data distribution
- Uniform load distribution - NO hotspots
Always on 24x7 Capability

FEATURES

Online DB upgrades and maintenance
- Online DB upgrades and HW maintenance
- Optimized swap operation to replace nodes

Online administrative operations
- All admin operations online
- Compaction
- Indexing
- Rebalance
- Backup & Restore

HA via Replication
- High availability using in-memory replication
- Auto or manual failover
- XDCR for disaster recovery

DR via XDCR
Flexible Data Model

FEATURES

Schema-less for structured / un/semi-structured data

Data with mixed structure better managed via JSON in a document DB than an RDBMS

Maintains Native object representation

Represent data as objects instead of shredding into rows and columns
Create indexes on any attribute of the document

Handles constantly changing data

Each document can have a different structure
Easy to change data without database changes and downtime
Basic Operation

- Docs distributed evenly across servers
- Each server stores both active and replica docs
  Only one server active at a time
- Client library provides app with simple interface to database
- Cluster map provides map to which server doc is on
  App never needs to know
- App reads, writes, updates docs
- Multiple app servers can access same document at same time
Add Nodes to Cluster

- Two servers added with one-click operation
- Docs automatically rebalance across cluster
  - Even distribution of docs
  - Minimum doc movement
- Cluster map updated
- App database calls now distributed over larger number of servers

Couchbase Server Cluster

App Server 1
- COUCHBASE Client Library
  - CLUSTER MAP

App Server 2
- COUCHBASE Client Library
  - CLUSTER MAP

SERVER 1
- Active
  - Doc 5
  - Doc 2
  - Doc 9
  - Replica
    - Doc 4
    - Doc 1
    - Doc 8

SERVER 2
- Active
  - Doc 4
  - Doc 7
  - Doc 8
  - Replica
    - Doc 6
    - Doc 3
    - Doc 2

SERVER 3
- Active
  - Doc 1
  - Doc 3
  - Doc 6
  - Replica
    - Doc 7
    - Doc 9
    - Doc 5

SERVER 4
- Active
  - Replica
  - Doc
  - Doc

SERVER 5
- Active
  - Replica
  - Doc
  - Doc

Read/write/update

User Configured Replica Count = 1
Fail Over Node

- App servers accessing docs
- Requests to Server 3 fail
- Cluster detects server failed
  - Promotes replicas of docs to active
  - Updates cluster map
- Requests for docs now go to appropriate server
- Typically rebalance would follow
What is Couchbase Lite
JSON Anywhere

Couchbase [Lite] + Couchbase [Sync Gateway] + Couchbase [Server]

On + In Data + In Data

Apple, Android, HTML5
Traditional Mobile

- Construction/Deconstruction of JSON
- RDBMS Style Local Storage
- Complex Sync Code While Offline or Cannot Use App While Offline
- Application Server REST Services
Mobile DBaaS

- RDBMS Style
- Local Storage

- Complex Sync Code

- Scaling
- Simple

- JSON

- REST Services
Synchronization
Automated when Connectivity Present
Couchbase Lite
The only NoSQL Database for Mobile Devices

Features
Ultra-lightweight, secure JSON database
Native support for iOS, Android and REST/HTML5
Powerful conflict resolution
Automated Sync
Couchbase Lite
The only NoSQL Database for Mobile Devices

Features
Ultra-lightweight, secure JSON database
Native support for iOS, Android and REST/HTML5
Powerful conflict resolution
Automated Sync
Couchbase Lite Features

- Use JSON in your Mobile App Data Tier
- Always Work with Local Data on Device
- Modify schema dynamically, super flexible
- **When Connected:** Create New Data and Change Documents, they Sync Automatically
- **When Not Connected:** Work with solid local NoSQL database
- Powerful conflict resolution features
The Couch syncing history
Remember this?

Lotus Notes

yeah, it can do that too
CouchDB replication

- Uses the CouchDB replication protocol
- Replication via HTTP
- Proven to work nicely
Couchbase Sync Gateway

CouchDB replication with Couchbase
So what can we do

- Handle user authentication
- Handle document channels
- Synchronise multiple devices / applications
Channels organize what JSON Documents are synchronized down to which Users

User A
cannels: food

User B
cannels: sports

User C
cannels: food, travel

User D creates Content with Channel association

These Users Subscribed to Content Channels

---

doc.channels : “food”
doc.channels : [ “food”, “sports” ]
doc.channels : [ “sports”, “travel” ]
User Authentication?

- Facebook
- Mozilla Persona
- OAuth
- Username / Password
Setting everything up

- Install Sync Gateway
- Setup a Sync Gateway config
- Setup the Couchbase Bucket to use
- Install CouchbaseLite (Cocoa Pods anybody?)
"message": "Hello World!"
Ok that was easy, right?
Let’s build a TodoApp
What do we need?

- Manage lists and tasks
- A datastore wired to a view
- User login
- Offline functionality
- Cross device usability
Demo
@interface Titled : CBLModel

+ (NSString*) docType;

- (instancetype) initInDatabase: (CBLDatabase*)database
    withObject: (NSString*)title;

@property (copy) NSString* title;
@property (readonly) NSDate* created_at;

@end
// Returns a query for all the lists in a database.
+CBLQuery* queryListsInDatabase: (CBLDatabase*)db {
  CBLView* view = [db viewNamed: @"lists"];
  if (!view.mapBlock) {
    [view setMapBlock: MAPBLOCK({
      if ([doc[@"type"] isEqualToString:kListDocType])
        emit(doc[@"title"], nil);
    }) reduceBlock: nil version: @"1"];
  }
  return [view createQuery];
}
Setting up the model

[[self.database modelFactory] registerClass:[List class] forDocumentType:@"list"];
Wire up the Datastore

- Implements a datasource for TableView
- Monitors any changes
Wire up the Datastore

#import <UIKit/UIKit.h>
#import <Couchbaselite/CBLUITableSource.h>

@class DetailViewController;

@interface MasterViewController : UIViewController
    <CBLUITableDelegate>

@property (strong, nonatomic) DetailViewController *detailViewController;
@property (nonatomic) IBOutlet CBLUITableSource* dataSource;
@property (strong, nonatomic) IBOutlet UITableView* tableView;
@end
User Login

- Handled via Facebook
- Native Social Login handling in iOS
- (void)defineSync {
    pull = [_database createPullReplication:_remoteURL];
    pull.continuous = YES;

    push = [_database createPushReplication:_remoteURL];
    push.continuous = YES;

    [self listenForReplicationEvents: push];
    [self listenForReplicationEvents: pull];

    [_authenticator registerCredentialsWithReplications: @[pull, push]];
}
Questions?
Links

- CouchbaseMobile
- Code for TodoLite
- Code for Sync Hello World
- Introduction to Sync Gateway Video
- Sync Gateway Homebrew Formula