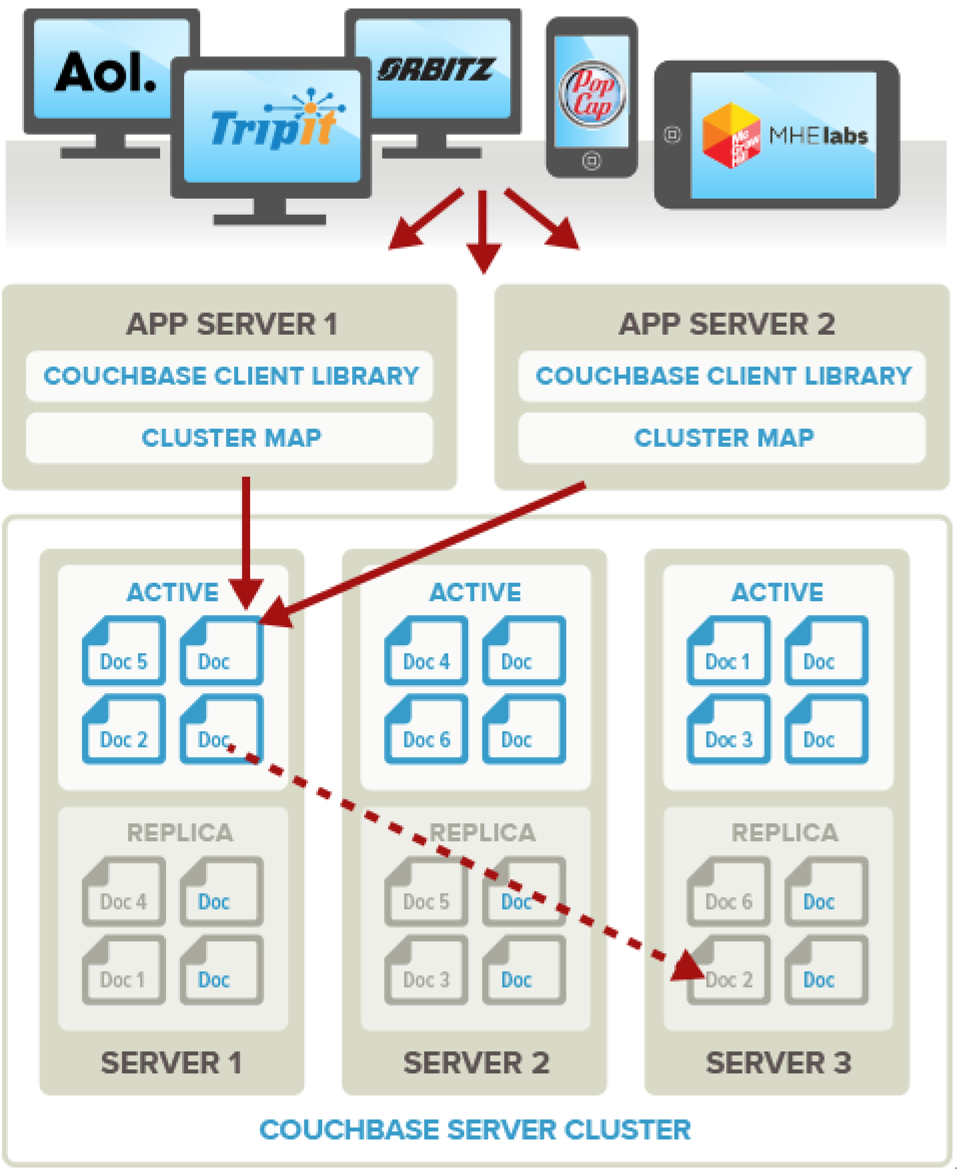
CouchBase – High-Level Architecture

At the highest level, each node in a Couchbase cluster is identical and has two main components: the data manager and the cluster manager. Other important architectural elements are a powerful map reduce engine to incrementally index and query documents, and cross datacenter replication technology to replicate documents across geographical data centers.



Applications use Couchbase smart client SDKs to interact with the Couchbase Server cluster. These SDKs are available for most popular languages including Java, .NET, PHP, Ruby, C and C++.

Our SDKs are aware of the cluster topology and automatically route requests to the right server.

Data is consistently hashed by the client to shards, which are evenly spread across all server nodes. The cluster map keeps track of which nodes store what data, and knows when nodes are added or go down.

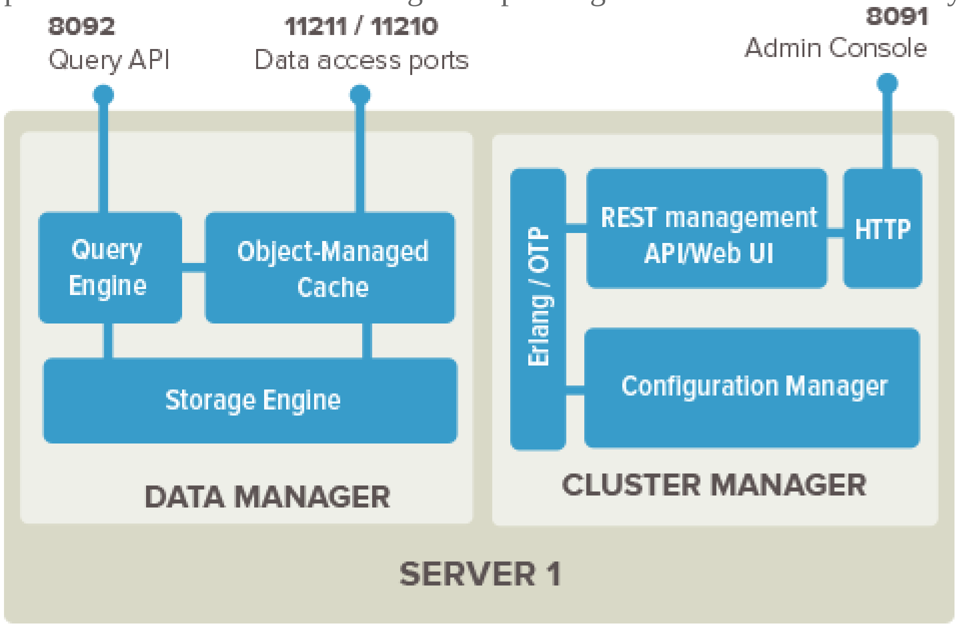
If a server crashes, Couchbase Server detects the failure, promotes replica documents on another server to active status, and updates the cluster map to reflect the new cluster topology. The smart clients automatically fetch the new cluster map from the server and route requests to the other active servers without any application changes or downtime.

Server Architecture

Every server in the Couchbase server cluster is identical, with two components: the cluster manager and the data manager. This architecture scales out linearly without any single point of failure. With a single click of a button, you can grow your cluster without any application downtime. Couchbase Server is a high-performance database that provides submillisecond application response time and high throughput – with just a few servers, you can provide an awesome application experience to many concurrent users.

Cluster Manager

The cluster manager supervises the configuration and behavior of all the servers in a Couchbase cluster. It configures and supervises internode behavior like managing replication streams and rebalancing operations. It also provides metric aggregation and consensus functions for the cluster, and a RESTful cluster management API. The cluster manager is built atop Erlang/OTP, a proven environment for building and operating fault-tolerant distributed systems.



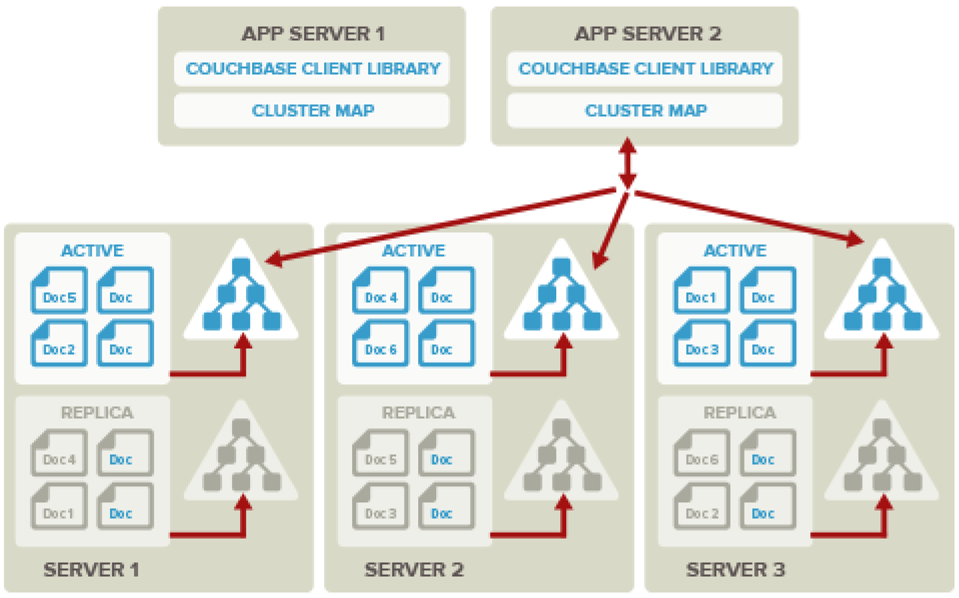
In the event of a server failure, the cluster manager detects that a node is down, promotes replica documents to active status, and recomputes the cluster map so that the client application can continue operating without any downtime. When new servers are added to the cluster or existing servers removed, the cluster manager rebalances data so that data and I/O are uniformly distributed across all cluster nodes.

Data Manager

The role of the data manager is data storage and access. On each node, the data manager takes care of active and replica data for a subset of the shards it manages. The 1024 shards are spread across the nodes in the cluster. Documents in JSON format are stored in logical entities called buckets. Clients access documents stored in a bucket by communicating directly with the node responsible for the corresponding shard.

There are multiple replica copies of each document in the cluster (configurable to up to three replicas). The data manager has a built-in object-managed cache for consistent sub-millisecond latency for random read and write operations, independent of the load or size of the data accessed. Read and write operations first go to the in-memory object-managed cache – if a document being read is not in the cache, it is fetched from disk. Updates to documents are first made in the in-memory cache and later eventually persisted to disk. Fine-grained document-level locking boosts request throughput to support millions of concurrent users with fewer servers.

The data manager exposes two “memcapable” ports, one to support Couchbase smart clients and one to support memcached client libraries via a proxy. With these ports, Couchbase Server can be a drop-in replacement for memcached. Schema-less data is stored in an append-only storage engine that is indexed and queried across the cluster. Most of the code in the data manager is written in C and C++.

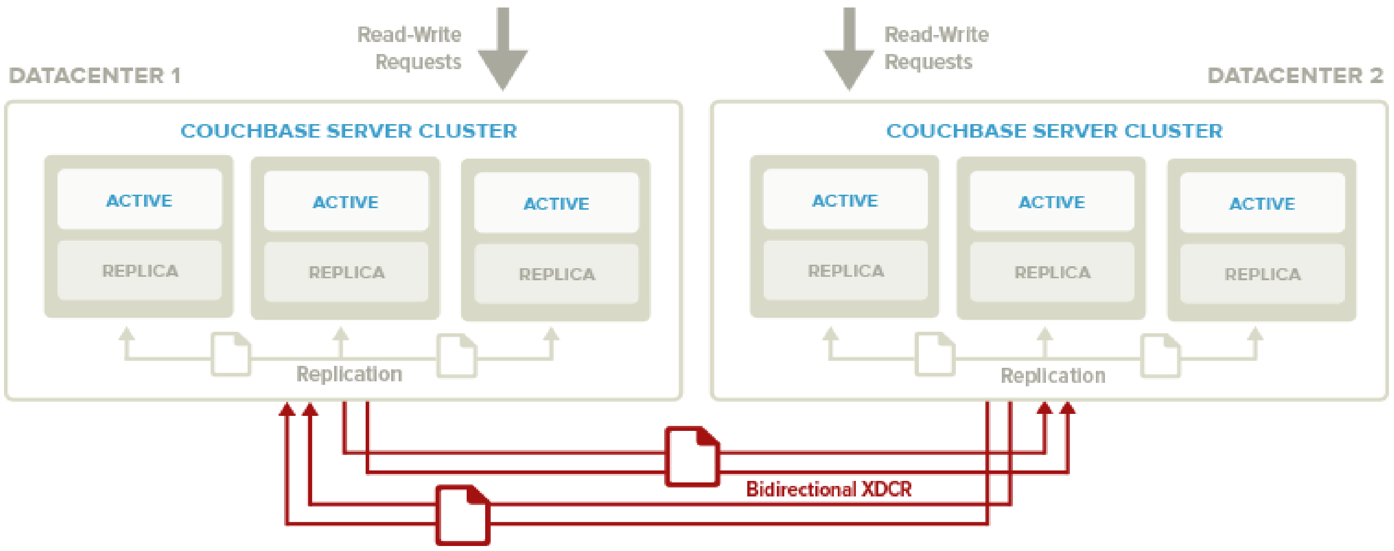


Indexing and Querying

Using incremental map reduce Javascript queries, a subset of the data can be queried by the application for simple real-time analytics. Indexes are queried in a distributed fashion: query requests scatter and fetch data from different Couchbase Servers and return the aggregated result back to the client.

XDCR Architecture

Couchbase Server easily replicates data from one cluster to another. It replicates active data to multiple, geographically diverse datacenters either for disaster recovery or to bring data closer to the users for faster data access. Cross datacenter replication (XDCR) and replication within a cluster occur simultaneously. In the figure below, replication takes place within the clusters at datacenter 1 and datacenter 2, while XDCR replicates documents across datacenters. Both datacenters are serving read and write requests from the application


Read-Write Requests
DATACENTER 1
COUCHBASE SERVER CLUSTER
ACTIVE
REPLICA
ACTIVE
REPLICA
ACTIVE
REPLICA
Replication
Read-Write Requests
DATACENTER 2
COUCHBASE SERVER CLUSTER
ACTIVE
REPLICA
ACTIVE
REPLICA
ACTIVE
REPLICA
Replication
Bidirectional XDCR