Couchbase

Extend Your Data Tier with XDCR

Intro to cross datacenter replication

Interactive web applications need a database that is always on (24x365), period. While replication within a database cluster helps with single server failures giving you high availability, your system still remains at risk from catastrophic failures – loss of power, natural disaster, etc. You also need a way to improve the response time for users when they are globally distributed. Is your database ready?

Couchbase Server 2.0 supports cross datacenter replication (XDCR), providing an easy way to replicate data from one cluster to another for disaster recovery as well as better data locality (getting data closer to its users).

This whitepaper describes how XDCR works in Couchbase Server including the difference between intra-cluster replication and cross datacenter replication, various XDCR topologies, and use cases where XDCR is beneficial.

Intra-cluster replication vs. cross datacenter replication (XDCR)

Couchbase Server provides support for both *intra-cluster replication* and cross datacenter replication (XDCR).

Intra-cluster replication is the process of replicating data on multiple servers within a cluster in order to provide data redundancy should one or more servers crash. Data in Couchbase Server is distributed uniformly across all the servers in a cluster, with each server holding active and replica documents (see Figure 1). When a new document is added to Couchbase Server, in addition to being persisted, it is also replicated to other servers within the cluster (this is configurable up to three replicas). If a server goes down, failover promotes replica data to active data.

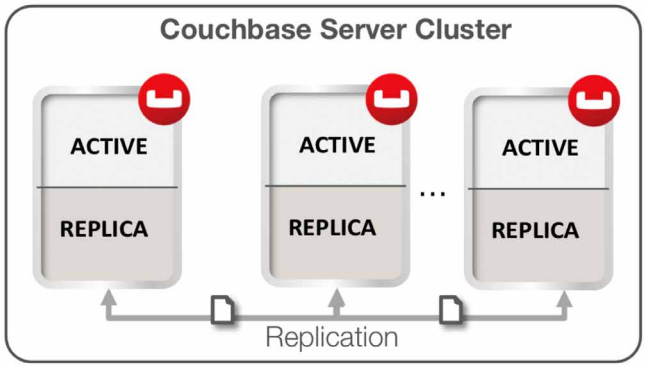

Figure 1: Replication within a cluster in Couchbase Server

Cross datacenter replication in Couchbase Server involves replicating active data to multiple, geographically diverse datacenters either for disaster recovery or to bring data closer to its users for faster data access, as shown in Figure 2.

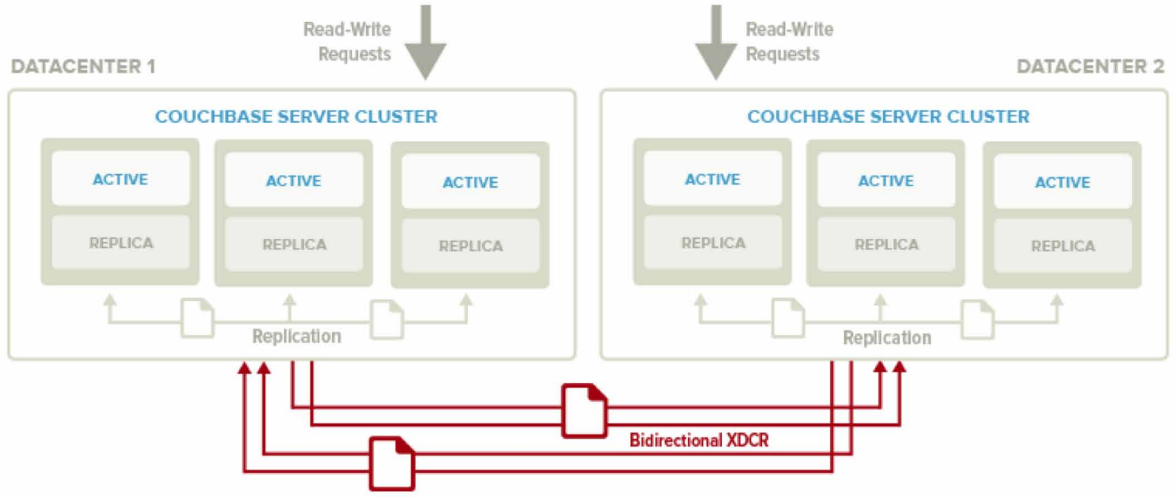

Figure 2: Cross datacenter replication in Couchbase Server

You can also see in Figure 2 that XDCR and intra-cluster replication occurs simultaneously. Intra-cluster replication is taking place within the clusters at both Datacenter 1 and Datacenter 2, while at the same time XDCR is replicating documents across datacenters. Both datacenters are serving read and write requests from the application.

Basic topologies

XDCR can be configured to support a variety of different topologies; the most common are unidirectional and bidirectional.

Unidirectional Replication is one-way replication, where active data gets replicated from the source cluster to the destination cluster. You may use unidirectional replication when you want to create an active offsite backup, replicating data from one cluster to a backup cluster.

Bidirectional Replication allows two clusters to replicate data with each other. Setting up bidirectional replication in Couchbase Server involves setting up two unidirectional replication links from one cluster to the other. This is useful when you want to load balance your workload across two clusters where each cluster bidirectionally replicates data to the other cluster.

In both topologies, data changes on the source cluster are replicated to the destination cluster only after they are persisted to disk. You can also have more than two datacenters and replicate data between all of them.

XDCR can be setup on a per bucket basis. A bucket is a logical container for documents in Couchbase Server. Depending on your application requirements, you might want to replicate only a subset of the data in Couchbase Server between two clusters. With XDCR you can selectively pick which buckets to replicate between two clusters in a unidirectional or bidirectional fashion.

As shown in Figure 3, there is no XDCR between *Bucket A (Cluster 1)* and *Bucket A (Cluster 2)*. Unidirectional XDCR is setup between *Bucket B (Cluster 1)* and *Bucket B (Cluster 2)*. *There is bidirectional XDCR between Bucket C (Cluster 1) and Bucket C (Cluster 2)*.

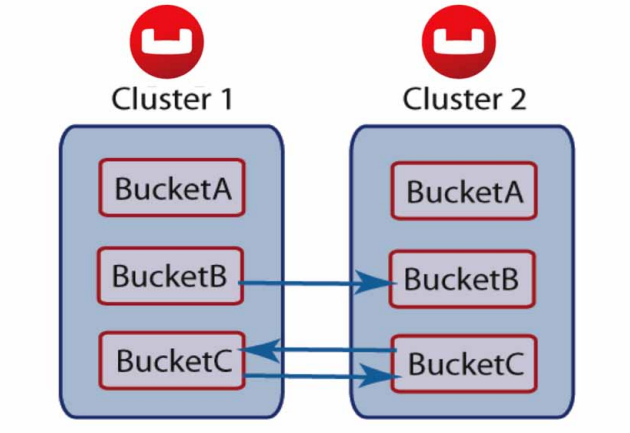

Figure 3: Replicating selective buckets between two Couchbase Server clusters

As shown in Figure 4, after the document is stored in Couchbase Server and before XDCR replicates a document to other datacenters, a couple of things happen within each Couchbase Server node.

1. Each server in a Couchbase cluster has a managed cache. When an application stores a document in Couchbase Server it is written into the managed cache.

2. The document is added into the intra-cluster replication queue to be replicated to other servers within the cluster.

3. The document is added into the disk write queue to be asynchronously persisted to disk. The document is persisted to disk after the disk-write queue is flushed.

4. After the documents are persisted to disk, XDCR pushes the replica documents to other clusters. On the destination cluster, replica documents received will be stored in cache. This means that replica data on the destination cluster can undergo low latency read/write operations.

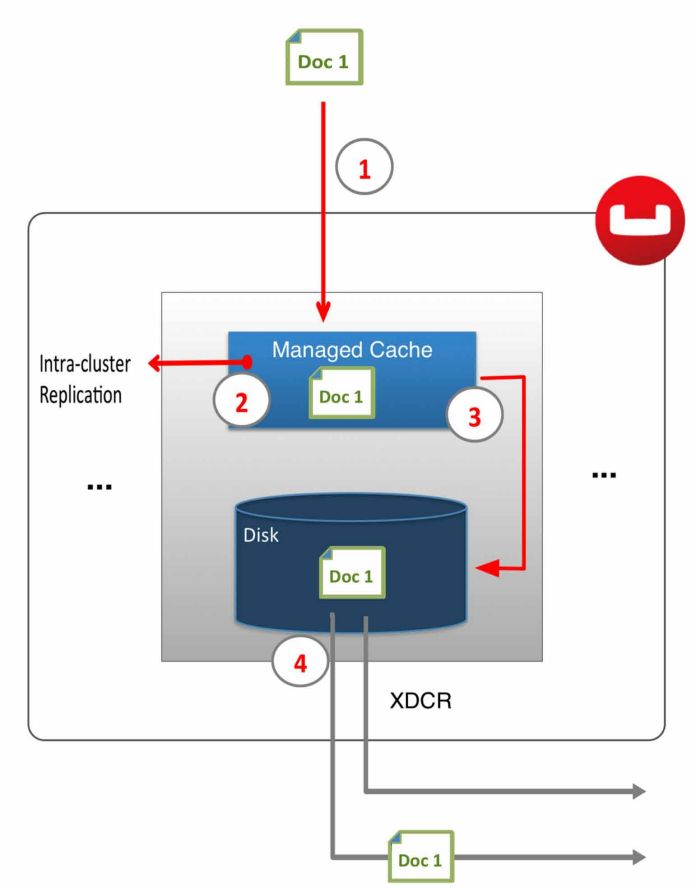

Figure 4: XDCR triggered after documents are persisted to disk

XDCR architecture in Couchbase Server

There are a number of key elements in Couchbase Server's XDCR architecture including:

Continuous replication

XDCR in Couchbase Server provides continuous replication across geographically distributed datacenters. Data mutations are replicated to the destination cluster after they are written to disk. There are multiple data streams (32 by default) that are shuffled across all shards (called vBuckets in Couchbase Server) on the source cluster to move data in parallel to the destination cluster. The vBucket list is shuffled so that replication is evenly load balanced across all the servers in the cluster. The clusters scale horizontally, more the servers, more the replication streams, faster the replication rate.

Cluster aware

XDCR is cluster topology aware. The source and destination clusters could have different number of servers. If a server in the source or destination cluster goes down, XDCR is able to get the updated cluster topology information and continue replicating data to available servers in the destination cluster.

Push-based connection resilient replication

XDCR in Couchbase Server is push-based replication. The source cluster regularly checkpoints the replication queue per vBucket and keeps track of what data the destination cluster last received. If the replication process is interrupted for example due to a server crash or intermittent network connection failures, it is not required to restart replication from the beginning. Instead, once the replication link is restored, replication can continue from the last checkpoint seen by the destination cluster.

Efficient

For the sake of efficiency, Couchbase Server is able to de-duplicate information that is waiting to be stored on disk. For instance, if there are three changes to the same document in Couchbase Server, and these three changes are waiting in queue to be persisted, only the last version of the document is stored on disk and later gets pushed into the XDCR queue to be replicated.

Active-active conflict resolution

Within a cluster, Couchbase Server provides strong consistency at the document level. On the other hand, XDCR also provides eventual consistency across clusters. Built-in conflict resolution will pick the same “winner” on both the clusters if the same document was mutated on both the clusters. If a conflict occurs, the document with the most updates will be considered the “winner.” If the same document is updated the same number of times on the source and destination, additional metadata such as numerical sequence, CAS value, document flags and expiration TTL value are used to pick the “winner.” XDCR applies the same rule across clusters to make sure document consistency is maintained.

As shown in Figure 5 below, bidirectional replication is set up between Datacenter 1 and Datacenter 2 and both the clusters start off with the same JSON document (Doc 1). In addition, two additional updates to Doc 1 happen on Datacenter 2. In the case of a conflict, Doc 1 on Datacenter 2 is chosen as the winner because it has seen more updates.

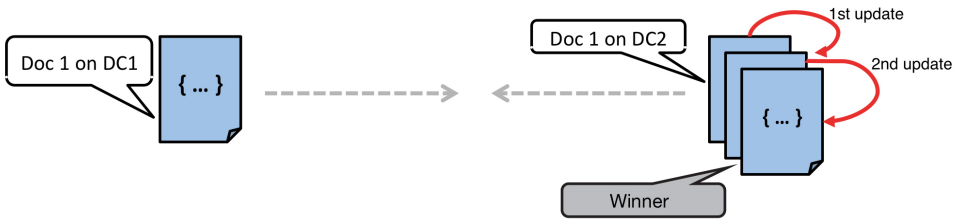

Figure 5: Conflict detection strategy in Couchbase Server

Easy configuration and monitoring

XDCR can easily be configured and managed through the Couchbase admin console. With just a few button clicks in the admin console UI, you can setup XDCR between your Couchbase clusters and get granular XDCR stats for source and destination clusters. For more details, you might want to refer to the XDCR documentation in the Couchbase Server 2.0 admin guide.

Advanced topologies

By combining unidirectional and bidirectional topologies, you have the flexibility to create several complex topologies such as the chain and propagation topology as shown below:

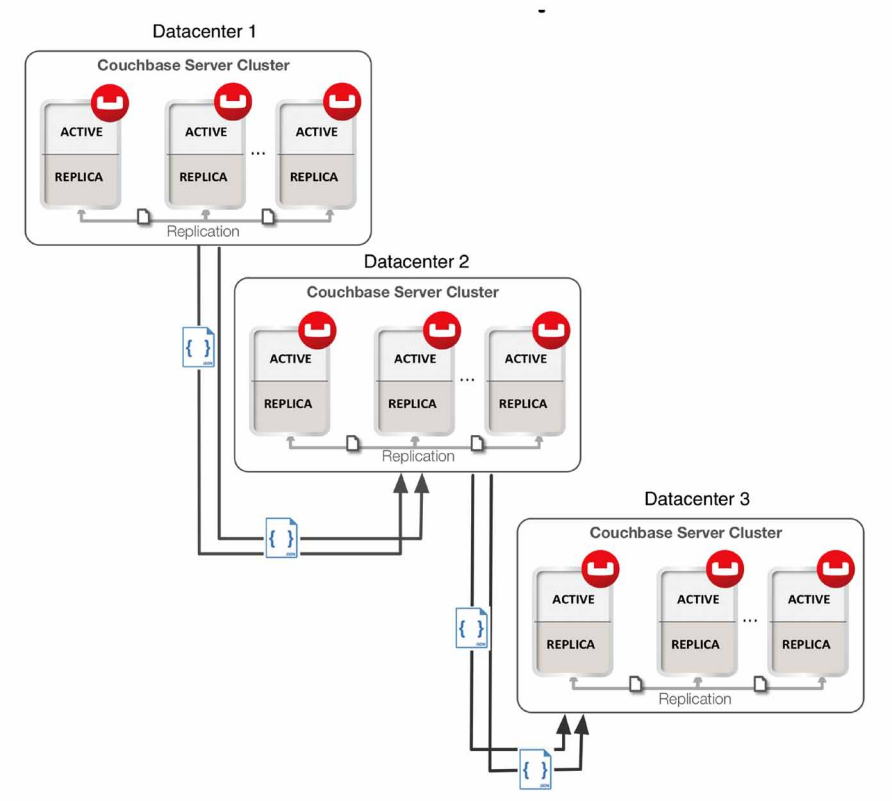

Figure 6: Designing a replication chain using unidirectional replication

In Figure 6, two unidirectional replication links are set up to form a chain replication topology between Datacenter 1 and Datacenter 2, and between Datacenter 2 and Datacenter 3. Chain replication topology is useful for minimizing datacenter network bandwidth because the cluster at the head of the chain only replicates to the next cluster along the chain rather than all the clusters.

In Figure 7, there is one bidirectional replication link between Datacenter 1 and Datacenter 2 and two unidirectional replication links between Datacenter 2 and Datacenters 3 and 4. Propagation replication can be useful in a scenario when you want to setup a replication scheme between two regional offices and several other local offices. Data between the regional offices is replicated bidirectionally between Datacenter 1 and Datacenter 2. Data changes in the local offices (Datacenters 3 and 4) are pushed to the regional office using unidirectional replication.

Figure 7: Using bidirectional and unidirectional replication to selectively replicate data between datacenters

Use cases

Disaster recovery is a common reason for implementing XDCR, but it is certainly not the only one:

Disaster recovery

Disaster can strike your datacenter at any time – often with little or no warning. With active-active cross datacenter replication in Couchbase Server, applications can read and write to any geo-location ensuring availability of data 24x365 even if an entire datacenter goes down.

Bringing data closer to users

Interactive web applications demand low latency response times to deliver an awesome application experience. The best way to reduce latency is to bring relevant data closer to the user. For example, in online advertising, sub-millisecond latency is needed to make optimized decisions about real-time ad placements. XDCR can be used to bring post-processed user profile data closer to the user for low latency data access.

Data replication for development and test needs

Developers and testers often need to simulate production-like environments for troubleshooting or to produce a more reliable test. By using cross datacenter replication, you can create test clusters that host subset of your production data so that you can test code changes without interrupting production processing or risking data loss.

Additional reading

Cross datacenter replication (XDCR) in Couchbase Server 2.0 Documentation

About Couchbase

We're the company behind the Couchbase open source project, a vibrant community of developers and users of Couchbase document-oriented database technology. Our flagship product, Couchbase Server, is a packaged version of Couchbase technology that's available in Community and Enterprise Editions. We're known for our easy scalability, consistent high performance, 24x365 availability, and a flexible data model. Companies like AOL, Cisco, Concur, LinkedIn, Orbitz, Salesforce.com, Shuffle Master, Zynga and hundreds of others around the world use Couchbase Server for their interactive web and mobile applications. www.couchbase.com