Couchbase

WHITEPAPER

High Availability and Disaster Recovery for Globally Distributed Data

Couchbase Cross Datacenter Replication (XDCR) Technology Overview

Contents

OVERVIEW

In today's digital economy, businesses can't afford downtime and need data to be readily available across geographies. With such massive, mission-critical data in today's distributed databases, high availability and performance are chief parameters for a database to deliver on. Ensuring the high availability of your data is not simply backing up one database as an up-to-the-second copy of another, data environments are much too complex and have multiple sources and modes of computing. There is an increasing trend toward cloud adoption to help with effective data management, and these emerging hybrid data infrastructures call for new approaches to data replication.

INTRODUCTION

Data Replication is the key to an effective distributed system that provides enhanced performance, high availability, and fault tolerance. There are two major types of replication:

- **Immediate consistency with synchronous replication**
  Systems that only acknowledge mutations to the application after replicated copies acknowledges the mutation. This is coupled with higher latencies.
- **Eventual consistency with asynchronous replication**
  Systems that acknowledge mutations immediately to the application as soon as single copy acknowledges the mutation. This is coupled with lower latencies.

Cross datacenter replication (XDCR)

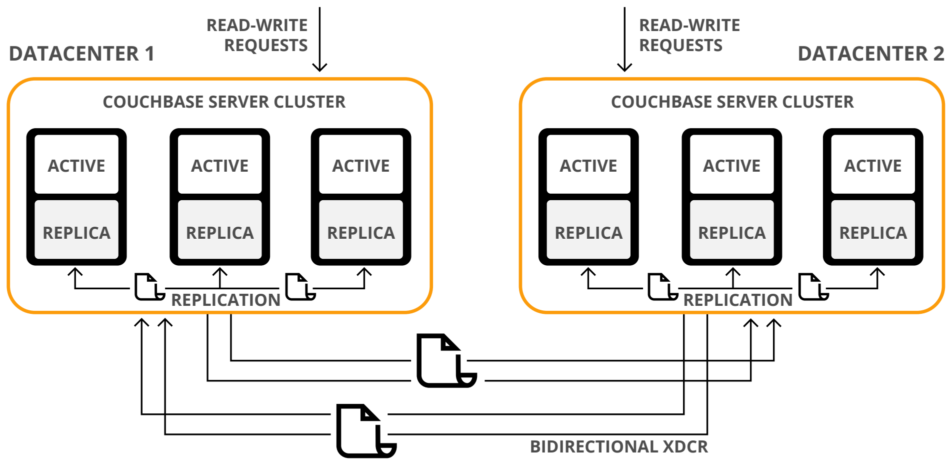
COUCHBASE'S CROSS DATACENTER REPLICATION (XDCR) TECHNOLOGY ENABLES CUSTOMERS TO DEPLOY GEO-DISTRIBUTED APPLICATIONS WITH HIGH AVAILABILITY IN ANY ENVIRONMENT (ON-PREM, PUBLIC AND PRIVATE CLOUD, OR HYBRID CLOUD).

Couchbase's cross datacenter replication (XDCR) technology enables customers to deploy geo-distributed applications with high availability in any environment (on-prem, public and private cloud, or hybrid cloud). This whitepaper provides a synopsis of the XDCR technology with an emphasis on architecture design, deployment flexibility, and enterprise-grade functionalities to meet today's data replication needs. This paper also describes how XDCR is well-suited to be used for hybrid infrastructure deployments and cloud migration.

XDCR offers data redundancy across sites to guard against catastrophic datacenter failures. It also enables global deployments for globally distributed applications. XDCR is based on a peer-to-peer scheme that treats each cluster as an independent unit of recovery, interconnected through a high throughput, low latency, memory-to-memory replication stream. Replication latency can be in the low single-digit millisecond range, excluding WAN delay. Replication latency is paramount for high availability in order to minimize data loss in catastrophic situations. Since updates are replicated asynchronously, XDCR does not interfere with the application write traffic.

XDCR is easy to set up. Administrators simply need to configure a replication between two databases (buckets) in the source and target clusters. Once set up, XDCR will automatically replicate JSON documents as necessary from the source cluster to the target cluster. This is all done transparently without any further action required by the administrator, even during topology changes (rebalancing and failover) on either clusters. Furthermore, XDCR can replicate documents across different infrastructures (bare metal, virtual machine, cloud, container) over the network, thus, making it easy for infrastructure migration. XDCR is secure and supports replication between clusters running on different Couchbase versions.

XDCR complements the local replication mechanism that is used to maintain replicas across nodes of a single cluster. Local replication has worked well under lower latencies, providing high availability during local failures. XDCR can also be deployed across sites to provide disaster recovery and high availability to accommodate cluster-wide failures. XDCR supports both IPv4 and IPv6 configurations; the source and destination clusters can be either all IPv4, IPv6, or a combination from IPv6 (source) to IPv4 (target). XDCR has the ability to write to any cluster and provides two conflict resolution mechanisms to resolve the potential conflicts covered in the following sections. Lastly, XDCR's user interface is built on a RESTful API-driven framework that can be accessed across all types of deployment platforms.



Use cases

High availability: Mission-critical applications today face near-zero downtime. XDCR, with its peer-to-peer architecture and no single point of failure, provides redundancy across sites to maintain high availability.

Disaster recovery: When catastrophic failures occur at the datacenter or regional level, customers need a solution that ensures minimal impact to their business continuity. XDCR can be set up in a unidirectional manner to create a hot backup system or in a bidirectional manner to create a load-balanced set of clusters to service their workloads. If some of the clusters in the XDCR setup experience cluster-wide disasters, such as power or network loss or total connectivity loss, remaining clusters can continue to serve the application workload.

Geo locality: Network latency around the world can be high, so many geo-distributed applications bring their data closer to their users to provide a responsive and fluid experience. XDCR allows replication of data effectively between continents and improve user experience by bringing data closer to them.

Workload separation: Today, many applications require predictably low latencies from their data processing platforms. Variances in latencies can create complications. Separating low latency workloads from heavy analytics processing prevents these complications. Using XDCR, users can separate data into different clusters specific to the use case. Many customers provision a separate analytics environment with XDCR and fine-tune their indexing and performance to meet the specific needs of the workloads.

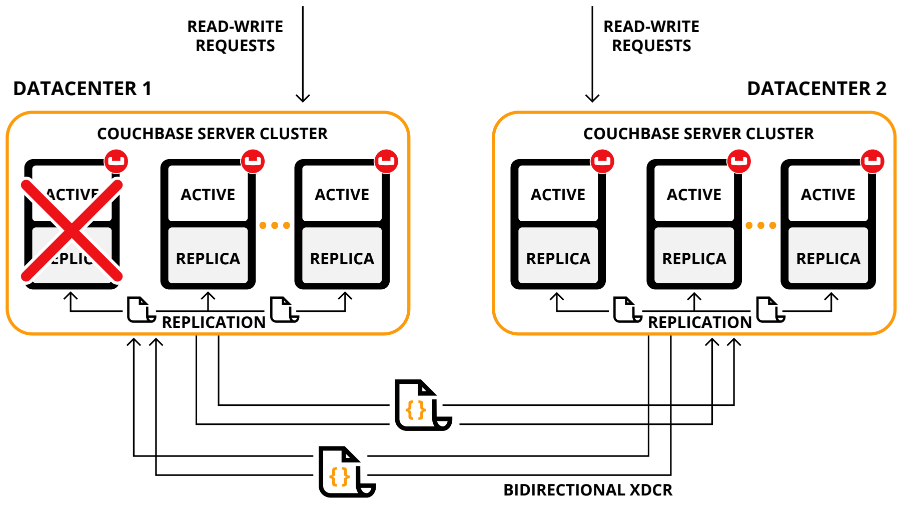
Hybrid deployment: A number of enterprises are adopting a hybrid deployment model with a combination of different clouds or even different infrastructures like on-prem and cloud. XDCR is infrastructure agnostic. Using XDCR, Couchbase can be deployed in a hybrid infrastructure where different clusters can be hosted on different platforms and replication can be set up between these clusters.

Cloud migration: XDCR can also be used to migrate Couchbase data between different clouds or even between on-prem and cloud. This prevents cloud vendor lock-in and provides customers the freedom to deploy anywhere they choose.

Benefits

- **High throughput:** XDCR is a memory-to-memory replication system. Couchbase splits all data in a bucket into 1,024 vbuckets. When moving data, vbuckets in source and destination pair up and independently move data without any centralized funnel. This parallelism in the architecture allows higher throughput and lower latency for data movement between clusters, both for the initial synchronization as well as the continuous data streaming.
- **Powerful topology support:** XDCR provides unidirectional replication as the building block for complex topologies, including bidirectional, circular, peer-to-peer, cascading, chained, plus hub-and-spoke for data aggregation scenarios. More details follow in the next section.
- **Independently scalable systems:** XDCR is used to replicate data between two independent Couchbase clusters, which means each of the clusters can have homogeneous or heterogeneous node distribution within the cluster. They can be scaled independently depending on the application's needs. The replication only depends on how fast the source cluster can stream out data and the target cluster can accept them. Source and destination clusters can have different hardware profiles, node counts, bucket settings or view definitions, and freely add/remove nodes.
- **Simplified administration:** XDCR provides a greatly simplified setup and administration interface. XDCR only requires a target cluster reference and a bucket-to-bucket replication setup. Once replication begins, XDCR continuously replicates data between the two clusters without any user intervention.

Further, XDCR requires no intervention under cluster topology changes (rebalance and failover). XDCR supports replication between different versions of Couchbase Server, even while a cluster is in the process of an online rolling upgrade. Even if there are individual node failures, transient or long-term network issues, XDCR resumes and restarts automatically.

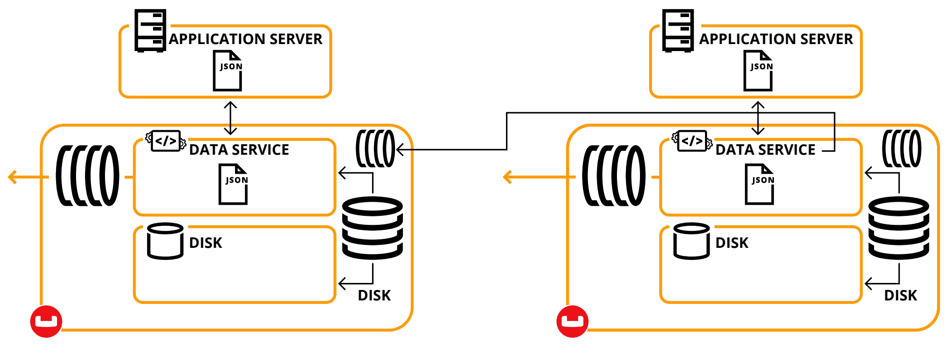


Highly customizable system: XDCR provides advanced users with a number of parameters to tune their replication, such as parallelizing replication streams and filtering data for best performance optimization. The UI, with embedded tooltips, provides all the guidance for an administrator to leverage these parameters to tune the replication. For more details, please check out our documentation.

COUCHBASE XDCR ARCHITECTURE

XDCR is a memory-to-memory, highly performant replication system that replicates data at the speed of the network and memory. The application data from the app server is written to our built-in memory (cache) through the client SDKs. Once this data is in memory, it is channelized into different queues for intra-cluster replication, inter-cluster replication (XDCR), and persistence. Couchbase utilizes a data synchronization streaming protocol called Database Change Protocol (DCP) which detects the incoming mutations and relays them to clients such as XDCR.

CROSS DATACENTER REPLICATION

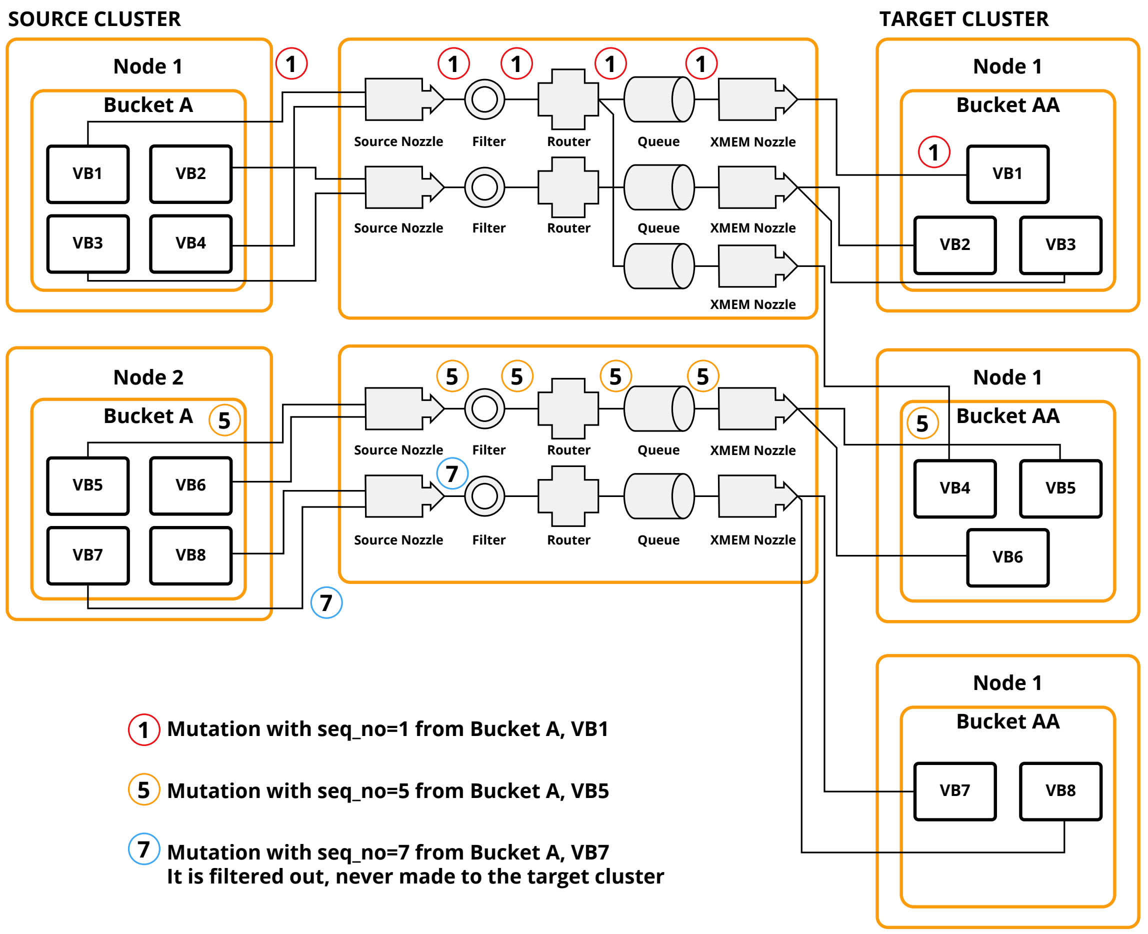


Each Couchbase Server bucket contains logical partitions called vbuckets (shards), which are equally distributed across the nodes of a cluster. Based on the placement, each node ends up with a number of active and replica vbuckets. XDCR retrieves the vbucket server map for both the source and target cluster as the input to construct a replication pipeline.

Each XDCR pipeline consists of the following parts:

- **Source nozzle –** streams out data from source Couchbase node using DCP
- **Filter –** filters the data according to the filter expression in the replication specification
- **Router –** routes the data to the right path leading to the target vbucket
- **Queue –** provides buffering between source and target
- **Outgoing nozzle –** sends data to the target Couchbase node
    - **XMEM nozzle –** uses memcached protocol to propagate the data to the target vbucket

The following diagram shows a replication request of rdata in Bucket A from source cluster (2 nodes) to Bucket AA in the target cluster (3 nodes). They are implemented by two pipelines running in parallel to replicate data from each of the source nodes to target nodes. Each pipeline runs on the source cluster's node. This parallelism enables XDCR to achieve high throughput and performance.



Checkpointing

XDCR maintains checkpoints to create a resume point to handle replication failure scenarios as well as pause/resume scenarios. A checkpoint reflects a point-in-time snapshot of the replication. The frequency of checkpointing is controlled by "checkpoint_interval" in replication specification metadata. Checkpoint history consists of a list of checkpoints.

When replication resumes after a communication failure, the last checkpoint serves as the resume point instead of re-streaming data from the beginning. The checkpoints are persisted on disk soon after creation, and can be retrieved as needed.

The XDCR checkpoints are stored as XDCR metadata that are independent of the document metadata.

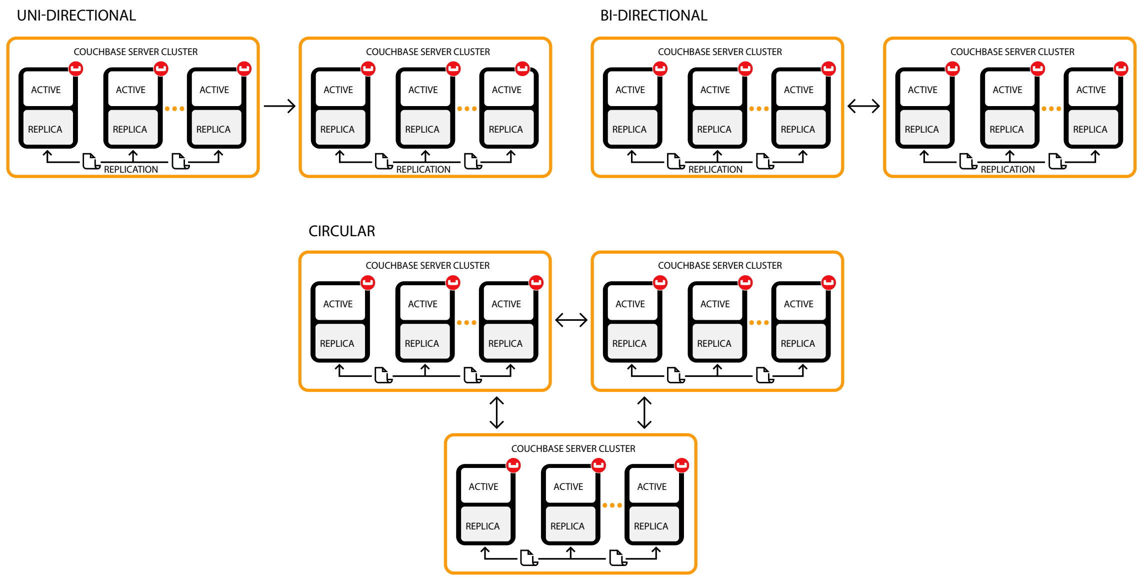
XDCR checkpointing interval

Checkpointing interval defines the interval at which XDCR performs the checkpointing operations. When the interval is short, checkpointing operations are carried out more frequently, and less data will be lost if replication has to be restarted. The default is 600 seconds.

The checkpoint interval can also be configured through the "XDCR Checkpoint Interval" setting. More details in documentation.

Flexible topologies

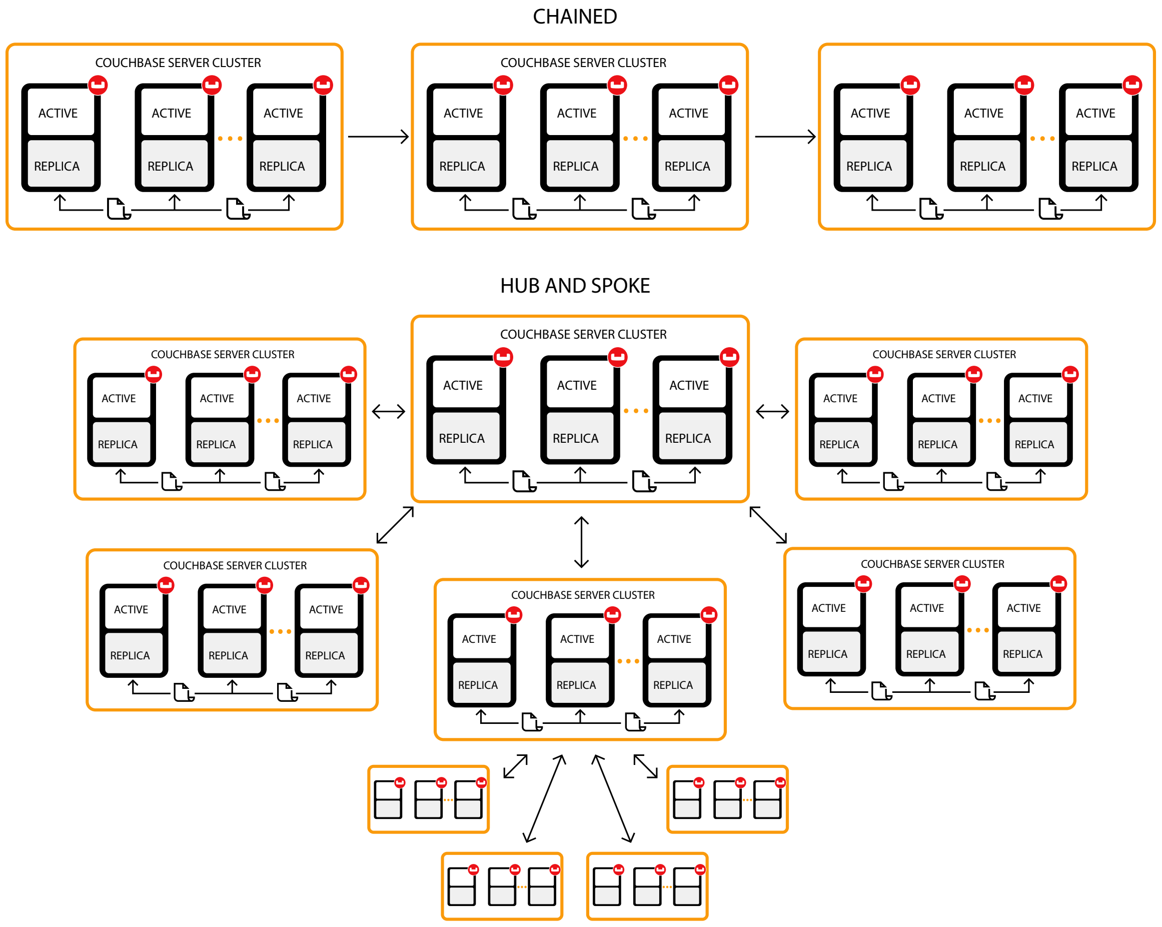
Due to its peer-to-peer scheme, XDCR provides the flexibility to set up any replication topology for data redundancy. Administrators can set up a simple peer-to-peer replication between a pair of source and target clusters, or assemble complex topologies from multiple peer-to-peer replication streams. XDCR comes with powerful topology support. Administrators can set up complicated circular and cascading replication topologies from star to mesh to full circular topologies. Some of the topologies are illustrated in the diagram below.



Unidirectional replication: As previously mentioned, unidirectional replication is the most fundamental topology, where data from the source cluster gets replicated to the destination cluster. Unidirectional topology is typically used to replicate to a standby cluster or distribute to a read-only cluster for lower latency reads or read heavy traffic. Another common use case for unidirectional XDCR is load separation. With this topology, one can replicate data with XDCR and index the destination cluster to optimize their analytics workload and isolate it from their online data processing on the source cluster.

Bidirectional replication: Bidirectional replication allows two clusters to replicate data to each other in a circular topology. Setting up bidirectional replication in Couchbase Server requires two unidirectional replication links between the clusters involved. Bidirectional replication is useful when one wants to load balance their entire read and write workload across two clusters. Some applications randomly distribute the workload, use multi-master writes, and can tolerate write conflicts. The majority of applications implement workload isolation for specific sets of data to reduce/eliminate overlapping writes and avoid conflicts in most cases.

Using these building blocks, one can set up infinite topologies between clusters. After the topology is set up, it is equally simple to change. Adding a new cluster to the topology is just a simple act of setting up new replications between the new cluster and the existing clusters in the topology. The figure below provides some supported examples of such complex topologies.



Complex topologies can offer a high degree of data availability with good resource utilization. For example, a ring topology allows each cluster to take on active traffic, while simultaneously acting as a hot standby for its neighboring clusters. This offers high availability, as a ring topology of four clusters can protect up to three cluster failures.

UNDERSTANDING XDCR

Couchbase Server guards against local failures by providing intra-cluster replication. Many mission-critical apps, however, require protection against cluster-wide failures. XDCR protects against these failures by allowing replication of data between clusters globally. There are a number of considerations for applications when targeting a single cluster versus multiple clusters connected through XDCR, including:

- Failover behavior with single versus multiple clusters
- Latency of replication within single versus multiple clusters
- Resolution of conflicts in an active-active setup
- Consistency model considerations with single versus multiple clusters
- Network bandwidth optimization during replication between clusters
- Security considerations for data transmission between clusters

The following details the differences and best practices for application developers developing applications targeting multiple clusters.

Auto-failover

Couchbase Server provides auto and manual failover capabilities within a single cluster and across clusters. Application traffic can be automatically failed over with the Multi-Cluster Awareness (MCA) feature introduced in Couchbase Server 5.0. MCA allows SDKs to monitor the health of clusters and detect cluster failures. Whenever a failure is detected, based on certain rules and priorities assigned to the clusters, traffic will be redirected to a different cluster. Users can specify the following parameters using SDKs:

- Prioritize a list of clusters
- Define rules to identify a failure
- Coordinate client interfaces via APIs
- Configure them using management interface

This relieves responsibility on the application developer to make the switch of directing the traffic from one cluster to another in case of catastrophic failures. In short, this feature provides automatic failover at the cluster level. However, the multi-cluster aware feature will not auto-failback upon the cluster/node as a protection for transient failures. Failback can be done without restarting app instances either programmatically or through Java management extensions. This can also be done manually, where applications are responsible for managing failover with XDCR.

Typically, applications perform failover based on server error messages and operation timeouts. Once the failure is detected, most applications implement a retry period coupled with a failover path. Failover happens by switching connection to another cluster.

Recovery time (RT) can depend on how aggressively the application detects the failure with retries and timeouts. With XDCR, applications have complete control over how this can happen. However, as XDCR replicates asynchronously, recovery point (RP) can be as long as the XDCR latency.

Aggressive application failovers may also increase the chances of false failovers. For example, clusters experiencing hiccups or transient resource contentions may cause applications to timeout, but they may also come back to life and continue to drain the remaining mutations in their queues.

Latency

Both intra-cluster replication and XDCR have a latency in receiving mutations from the master vbucket. Under steady state, latency in intra-cluster replication is much lower than that in XDCR due to the following:

- The number of hops between different clusters in the topology
- The rate of mutations on the source and the rate the destination can receive
- The network latency (WAN delays) between two clusters

Typical latency, excluding the network latency, is expected to be in the range of single digit millisecond; however, applications working with XDCR have to be aware of the increased latency that occurs during failover. Due to increased latency, recovery point (RT) may be larger in XDCR failovers compared to failovers within the cluster.

Conflict resolution

In an active-active setup, conflicts can occur in a replication environment that permits concurrent updates to the same data at multiple sites. For example, when two mutations originating from different sites modify the same document concurrently, a conflict can occur. A conflict resolution mechanism has to be chosen for the system data to converge.

XDCR supports two types of conflict resolution on non-overlapping keys:

a. Rev ID-based conflict resolution

b. Timestamp-based conflict resolution

MOST UPDATES WINS (REV ID) CONFLICT RESOLUTION

Rev ID indicates the number of revisions on a specific document. In a revision ID-based conflict resolution, Rev ID is used to resolve two conflicting writes to a single cluster. Revision IDs are maintained per key and are incremented with every update to the key. Hence, this conflict resolution can be best characterized as "the most updates wins." For example, in the following case, imagine an east and a west cluster setup with bidirectional replication. Applications updates the doc#1 in the east cluster at time T1. XDCR replicates data over to the west cluster at T2.

Another application makes three additional updates to doc#1 on the west cluster before the replication is propagated to the east cluster, at time T6, a third application updates the east cluster.

However, since only revision ID 0 is present in the east cluster, the update increments revision ID to 1. Once XDCR catches up, the last value written at time T6 is not picked as the winner, as revision ID 3 > 1.

East Cluster	West Cluster
T1. add doc#1 (rev 0)	
	T2. XDCR repl – add doc#1 (rev 0)
	T3. update doc#1 (rev 1)
	T4. update doc#1 (rev 2)
	T5. update doc#1 (rev 3)
T6. update doc#1 (rev 0)	
T7. XDCR repl – update doc#1 (rev 3) (not picked as winner)	

Tn (time)

Update conflicts based on updates and deletes are also resolved with the same mechanics. For example, if an application updates a given key in the east cluster three times in a row and before XDCR kicks in, on the west cluster a delete operation is performed on the same key; therefore, the east cluster with 3 updates wins the XDCR exchange and data is resurrected in the west cluster due to its higher Rev ID.

There are additional fields to help resolve conflicts in cases where revision IDs are the same. Refer to our documentation for more details.

LAST WRITE WINS CONFLICT RESOLUTION

Last write wins (timestamp-based) conflict resolution resolves conflicts by using the document timestamp as the defining parameter instead of the count of revisions.

Since XDCR is an eventually consistent system, timestamp-based conflict resolution is useful for ensuring that the most recent version of the document wins the conflict. This is also referred to as the last write wins (LWW) conflict resolution.

Time synchronization to order the mutations across the nodes for a distributed system like Couchbase is done using a hybrid logical clock (HLC). Hybrid logical clock is a combination of logical and physical clocks where the first 48 bits represent the physical time (up to 0.1 millisecond) and the last 16 bits is a logical counter. HLC maintains its logical clock to be always close to the NTP clock, and HLC is also used in applications such as snapshot reads in distributed key-value stores and databases. More information on HLC can be found here.

In an active-active setup, for conflict resolution between an incoming mutation from another cluster and a local mutation in the current cluster, if the incoming mutation does not have a higher timestamp value than the local mutation, it will be discarded and will not be stored. In the case of bidirectional replication, this process results in the mutation with the highest timestamp value taking precedence on both clusters.

The stat "mutations skipped by resolution" provides the number of conflict resolutions where the source mutations lost.

Safe failovers

Let's imagine two datacenters in Europe (EU) and the U.S. for the use case of a shopping cart for Christmas.

At T1, the user in U.S. adds an "Item 1- Tree stand" to the cart and U.S. datacenter fails immediately.

So, the application traffic has to be directed to EU datacenter.

There are two factors that XDCR takes into consideration before redirecting the traffic.

a. Replication latency between the two clusters – in this example U.S. and EU.

b. Absolute time skew between the two clusters – U.S. datacenter and EU datacenter.

Let's say the replication latency between U.S. and EU is 10ms and time skew between U.S. and EU datacenters are 5ms, XDCR waits for max (replication latency, time skew) before redirecting the traffic to the EU cluster. In this case, it would wait for max (10,5) = 10ms before redirecting the traffic to the EU datacenter.

This ensures that any mutations in-flight are received by the EU datacenter and the writes to the EU cluster occur after the last write to U.S. cluster.

Network bandwidth optimization

Network bandwidth is one of the crucial factors in general for data transfer throughput. There are a number of new features introduced in XDCR to optimize the bandwidth and to improve replication throughput, which in the past has been a limiting factor for XDCR.

Replication modes

Pessimistic replication: Pessimistic replication sends only the keys in the batch while replicating. This is especially applicable to big documents. If the keys are accepted by the destination, values are replicated with another round trip. If they are not accepted, the second trip is skipped, which conserves bandwidth.

Optimistic replication: In optimistic mode, keys and values are shipped together in a single batch. For small documents, the potential for network usage reduction is lower, and it simply sends everything in one single trip. The behavior is configurable based on your workload through the "XDCR Optimistic Replication Threshold" setting. More details in documentation.

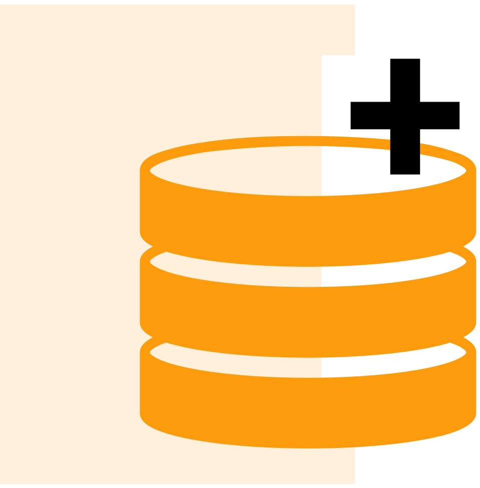
Replication filtering

In Couchbase Server 4.6.0, XDCR introduced key-based filtering to support the ability to replicate only a subset of data based on document keys. This aided the customers tremendously to filter their replication traffic based on their business logic. In the upcoming release 6.5, this feature will be extended to support value-based filtering where filtering can be based on keys, values, or extended attributes.

Using N1QL-like (Couchbase's SQL-based query language for JSON) syntax, filter expressions can be constructed using regex, arithmetic, boolean, or logical operators to filter the data specific to their business needs. Negative lookaheads and replication by excluding TTL will be supported. Customers will also be able to modify the filters on the fly for an ongoing replication, and choose to continue or restream the replication. These advanced filtering capabilities will enable customers to use XDCR for geo-fencing use cases in addition to providing bandwidth conservation.

Network bandwidth throttling

With Couchbase Server 5.0, XDCR introduced the ability to limit the bandwidth to be utilized by a cluster by introducing a parameter known as "network usage limit". This can be used to specify the upper bound in MB/sec, which is the maximum bandwidth available for a cluster for replicating data using XDCR. This available bandwidth will be equally distributed among the nodes in the cluster. One can allocate the bandwidth to the cluster based on their required replication throughput, and XDCR will perform accordingly. It is highly valuable to limit the bandwidth utilization for cloud deployments.



Data compression

Data compression in XDCR is useful to replicate the data in compressed form which aids in bandwidth conservation. For the customer deployments in the cloud, this also leads to direct cost savings as they are charged based on bandwidth utilization. With the default behavior, if the incoming data to the XDCR queue is compressed, XDCR will replicate it as compressed. However, customers have the choice to opt in / opt out explicitly as well. XDCR uses the Snappy for compression. Snappy aims at providing reasonable compression with maximum performance. The compression ratio is subjective to factors such as size and content of the document.

Replication prioritization

By design, XDCR provides customers the flexibility to tune the number of replications for a given bucket depending on the desired performance. New replication requires streaming all existing documents in the bucket, and, therefore, it exhibits a higher mutation rate than ongoing mutations. As such, new replication uses more resources, which would sometimes negatively impact the throughput of existing replications in the meantime. With our upcoming 6.5 release, XDCR will provide users the ability to prioritize ongoing replications over initial replications by specifying the priority to be assigned as high, medium, or low. Resource allocation will happen based on the specified priority. This provides customers with improved control over their resource management.

Security

ROLE-BASED ADMIN ACCESS CONTROL (RBAC)

For XDCR, RBAC occurs under cluster scope for creating cluster references and at the bucket scope for managing replications. Full administrators, cluster administrators, bucket administrators, and replication administrators have the ability to manage the XDCR. While the bucket administrator can start/stop replications, he/she cannot create cluster references. The detailed role-based privileges are discussed in the documentation.

SECURE AUTHENTICATION

When connecting to the remote cluster, authentication can happen in three modes:

- Username and password
- Scram-sha authentication
- X.509-based client cert authentication on top of TLS

While the username and password is the simplest approach, critical applications demand highly secure connections which can be fulfilled either by using scram-sha for password hashing or a fully encrypted TLS-based authentication where the entire data and password is encrypted and client authentication happens via X.509 certificate. More details in documentation.

XDCR on Kubernetes

Kubernetes is a container orchestration technology enabling automated deployments, scaling, and management of containerized applications. One of the core motivations for Kubernetes adoption is its infrastructure agnostic support which aids multi-cloud / hybrid cloud deployments.

XDCR plays extremely well into this as it's the vehicle for enabling multi-cloud deployments for Couchbase. While Kubernetes provides the automated deployment of any Couchbase cluster, XDCR helps in migrating from one host to another and maintaining the data consistency across these clusters which might be deployed anywhere across any infrastructure.

The Couchbase Autonomous Operator enables one to automate the management of day-to-day activities of Couchbase cluster in terms of configuration, creation, scaling, and recovery. XDCR is also often used as a disaster recovery solution at the datacenter or regional level. With XDCR, Autonomous Operator can automatically recover a Couchbase cluster without any downtime.

Other programmability considerations

Couchbase API provides a number of capabilities across SDKs and it is worth mentioning a few of the behavioral differences in using the capabilities against a single cluster versus multiple clusters tied with XDCR.

CAS token: CAS (check-and-set) token provides optimistic concurrency. CAS token is maintained per key, and with every mutation to a value, CAS token is updated. Couchbase SDK provides capabilities to get CAS token and use CAS for ensuring

no other operation is interleaved between the get and the set. You can use CAS to detect changes from updates other than XDCR, as XDCR replicates the CAS value associated with each key. That means, with incoming XDCR mutations the CAS value is updated to the CAS of the original mutation on the source cluster and not a new independent CAS value.

Locking items: Couchbase Server also supports pessimistic concurrency through a lock operation. However, the locking operation is only valid for locking keys within the cluster. Locks do not extend to the same key values on other clusters connected with XDCR.

Expiring items: Expiry works as expected with XDCR so applications don't have to be concerned with this flag. The metadata on expiry is replicated with the keys. If an item expires before it is replicated through XDCR, the deletion of the item is still replicated. It is important to note expiry is based on the UTC time of the node (UTC represents a coordinated universal time that is time-zone independent). However, local clocks across nodes and cluster may have skews. If a node is running ahead, it may expire values first before other copies of the same data with the same expiration time.

CONCLUSION

As the dependency on data continues to grow without any room for downtime, data replication will continue to be an essential paradigm to meet the availability and continuity strategies. Performance, high availability, cloud interoperability, and support for hybrid environments are now the benchmarks for databases and replication systems. XDCR, with its ability to replicate at the speed of the network, offer low latency (range of single digit millisecond), and act as a vehicle for cloud migration, is definitely one of the best solutions available to modern-day enterprises. This technology has continuously evolved in all the above-mentioned dimensions over the past eight years, and has become a mature, robust, and easy to deploy solution for mission-critical applications.

To learn more, contact your Couchbase sales representative today or visit: www.couchbase.com

Modern customer experiences need a flexible database platform that can power applications spanning from cloud to edge and everything in between. Couchbase's mission is to simplify how developers and architects develop, deploy and consume modern applications wherever they are. We have reimagined the database with our fast, flexible and affordable cloud database platform Capella, allowing organizations to quickly build applications that deliver premium experiences to their customers—all with best-in-class price performance. More than 30% of the Fortune 100 trust Couchbase to power their modern applications.

For more information, visit www.couchbase.com and follow us on Twitter.