Couchbase

Couchbase vs. MongoDB™ for Global Deployment

Introduction

Couchbase is a distributed NoSQL document-oriented database with a core architecture that supports a flexible data model, easy scalability, consistent high-performance, always-on 24x365 characteristics, and advanced security. It has been adopted across multiple industries and in the largest enterprises for their most business-critical applications. Many of those customers, including: cars.com, DirecTV and Staples, have gone through rigorous evaluations of Couchbase alongside MongoDB, and have chosen Couchbase based on a strong set of differentiated capabilities and architectural advantages, including:

- **Scale-Out and High Availability**
- **Global Deployment**
- **SQL/SQL++ Query for JSON**
- **Hybrid Operational and Analytical Processing**
- **Full-Text Search**
- **Server-Side Eventing and Functions**
- **Embedded Mobile Database**

In this paper, we will focus on how MongoDB compares to Couchbase when it comes to global deployment. Couchbase Server provides support for both intra-cluster replication and 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).

Global Deployment that is Scalable and Highly Available

MongoDB is Master-Slave At Best

MongoDB's multi-datacenter deployment is just an extension of its intra cluster distributed cluster management mechanism, with primary and replica sets. In case of multi datacenter deployment, the members of the replica set are distributed between different datacenters.

- **Write Unavailability.** One of the major issues with the architecture of MongoDB is that a shard can have only one primary for writes. That primary is a bottleneck and single point of failure. If it goes down, that shard will not be able to take any writes until another primary is elected which has a median ~12 seconds. Since secondaries copy the data from primary's oplogs, this implies for a specific data partition, when primary goes down, application is read-only.
- **Topology.** As only the primary can take writes, the only network topology that can be supported is unidirectional star topology. Rapidly expanding high availability applications with global data distribution need the flexibility to configure any topology (e.g. ring, hub-and-spoke, tree), as well as changing the topology for future needs with no impact on performance.

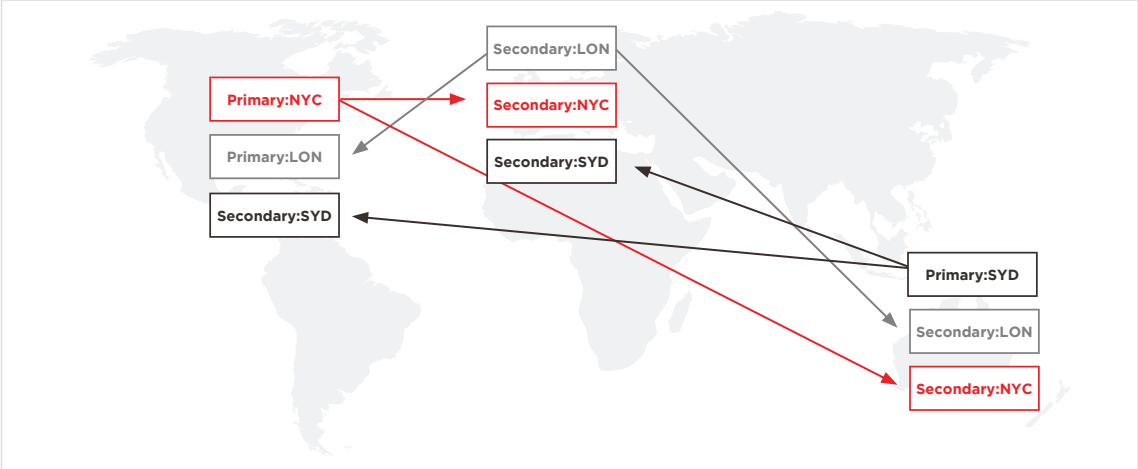
- **Multiple Hops and Increased Latency.** For multi-datacenter replication, the replica sets (mongod processes) are deployed in different datacenters. MongoDB provides horizontal scale-out across multiple nodes using sharding. As indicated in the figure below, applications issue queries to a query router that dispatches the query to the appropriate shards. So, for any query to be directed to the remote cluster, there are multiple hops from driver to query router to primary to secondary which increases latency.

  By default, the reads are always directed to the primary. Since there is only one primary per shard, any non-local reads add to latency. MongoDB provides nearest read preference which can be exercised to ensure the reads are directed to the nearest datacenter. However, since only primary can take writes, the non-local writes will continue to be very expensive and MongoDB cannot claim read to be consistent with writes for global deployments with read preference other than primary.

- **Setup and Manageability.** The approach to set up a successful replication across geographically diverse data centers is a cumbersome process especially due to replica set distribution challenges the admins have to encounter. For example:

  - Right distribution of replica set members to ensure they can form a majority for voting to elect primary and sustain datacenter loss. When the number of active members in the replica set is insufficient to provide a majority, no primary node can be elected for the replica set and it cannot provide write services. In this case, it is read-only. In a setup with simple primary and a single DR cluster for backup, the backup might not accept writes when primary goes down as there is no majority to elect it. A basic DR might not work in this setup.
  - Difference in distribution best practices for deployment in a 3 member replica set vs a 5 member replica set. This complexity increases with more replicas. Setting up a datacenter isn't cheap and such deployment requirements should not be imposed . An odd number of replica sets is advocated, if there is an even number without an arbiter, forming a quorum to elect primary might be challenging.
  - Configuration of member eligibility to become primary, defining priority 0 / hidden members, read preferences etc.
  - Configuration of majority success policy to ensure one primary overrules when multiple primaries exist. When a network partition occurs, multiple primary nodes may exist for a short time. In this case, it is necessary to set a majority success policy to ensure even if there are multiple primary nodes, a single primary node still can write data to the majority of nodes. Application does not know when cluster will go down and it could be expensive to use majority write concern.

- **Not Active-Active.** With a set up as indicated in the diagram below, MongoDB claims to configure each datacenter to accept writes. This is not truly an active-active setup because each datacenter is capable of taking writes only specific to that shard. Non-local writes (ex. SYD to NYC) are very expensive. Even with Global clusters introduced in 4.0, MongoDB is able to pin shards where ever the datacenter is located and define policies to direct the reads and writes to the local datacenter, but it is not a master-master architecture and all writes cannot be local. All other challenges indicated above will continue to persist.

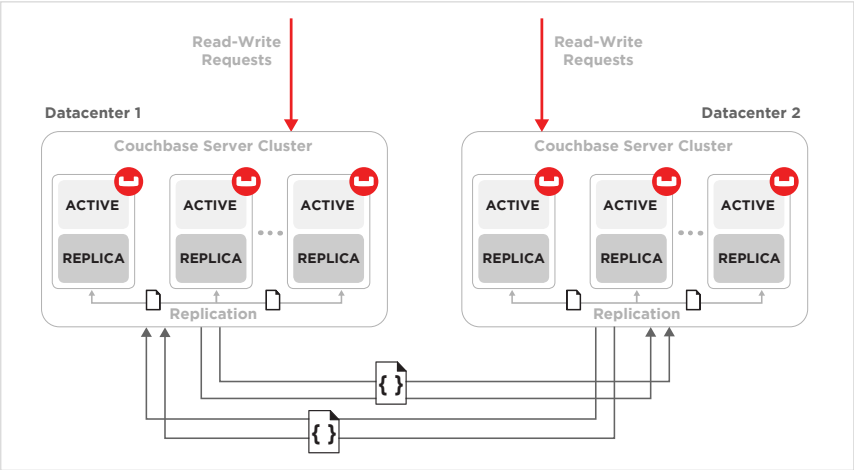


- **No Conflict Resolution.** There is no concept of conflict resolution as there is only one primary node for writes.

Couchbase XDCR is Flexible, Easy and Truly Active-Active

eBay has deployed XDCR with a 3 cluster bidirectional active-active ring topology for their session management application, Mirror Image uses XDCR for their online advertising application with a hybrid topology of bidirectional active-active linear topology and unidirectional star topology and Paypal uses XDCR with a 4 cluster bidirectional ring topology for an identity management application.

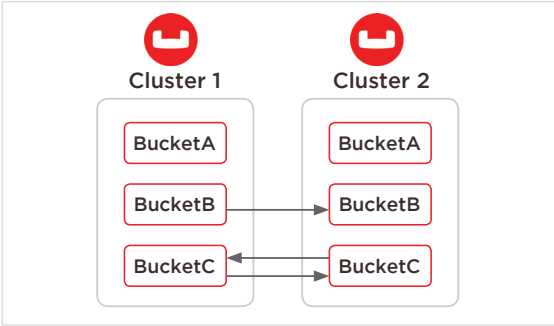
In Couchbase, cross datacenter replication (XDCR) is independent of intra cluster replication. Cross datacenter replication involves replicating *active* data to multiple, geographically diverse data centers.



- **High Performance Architecture.** Couchbase XDCR occurs between two or more completely independent clusters which both handle reads and/or writes. It is a memory-to-memory, highly parallel, stream based replication which can be tuned and even capped to efficiently manage bandwidth. Furthermore, XDCR can be paused/resumed manually if needed and is automatically resilient to any network disruption.

  All application access (read and/or write) occurs to the local cluster for best performance and highest availability. Depending on the desired behavior of the application, multiple clusters may handle write traffic or may be dedicated to read traffic only.

- **Flexibility.** XDCR offers highly flexible replication:



  - *Unidirectional or bidirectional:* Unidirectional or bidirectional replication is supported by XDCR.
  - *Topologies:* Multi-master architecture supports any topology such as star, ring, chain, mesh, one-to-many, many-to-one, etc.
  - *Bucket level replication:* Depending on application requirements, buckets can be selectively replicated.
  - *Filtering:* Key based filtering allows a further subset of data to be replicated. Advanced filtering (non-key based) is on the roadmap and will further strengthen this capability.

- **Setup & Manageability.** For XDCR, set up is extremely user-friendly via remote cluster reference with IP and user credentials. Once configured, clusters respond dynamically to topology changes without any further administration.
- **Active-Active Conflict Resolution**. With XDCR, multiple clusters can act as write masters. Conflicts occur when the same document is modified in two different locations before it is replicated. To maintain consistency between these two locations, one version is automatically chosen as the 'correct' version. Couchbase supports two conflict resolution mechanisms, timestamp-based (last write wins) and revision-based (most update wins).
    - *Timestamp-based conflict resolution:* the document version which was updated most recently (based on an HLC) will be chosen.
    - *Revision-based conflict resolution:* The version of a document with the most updates will be chosen.
- **Network bandwidth optimization.** With XDCR, the customers can set the limit for the bandwidth utilization depending on the application's need. This allows them to control the rate of replication as XDCR operates at the speed of network and memory. With the upcoming release, users will even have the flexibility to assign the priorities for their replication streams and dictate the quality of service.
- **Multi-cluster autofailover.** Via this functionality, client SDK monitors the health of clusters and based on the priorities assigned to the cluster,it will direct the traffic to a different cluster. This relieves the application developer of the complexity involved to make this switch. In short, this feature provides automatic failover at the cluster level.

Summary of Couchbase XDCR versus MongoDB Replication

XDCR Capabilities	Couchbase	MongoDB
Architecture	Completely independent cluster, which can be scaled and managed without any dependencies	Extension of intra-cluster, not an independent system
Performance	Memory-memory, stream based, highly parallelized replication. The number of replication streams per node can be (2-100)	Secondaries replicate the data from primary's oplog or any other secondary's oplog. It is parallel but streams are 1-1 (primary-secondary)
Write Concerns	Any cluster can be configured to accept writes	Only primary can take writes which impacts write availability and non-local writes are very expensive
Read Concerns	Always local	Default primary which might be expensive, manual configuration required to read from secondaries
Auto-failover	Cross cluster automatic failover can be enabled at the SDK level	Automatic but unpredictable
Replication Flexibility	Very flexible - bucket level, advanced optimization techniques to tune to the need	Tuning, choosing speed, bandwidth is not possible
Topology	Support for complex topologies - Bidirectional, Star, mesh, chain, ring anything	No support for complex topology - Unidirectional, Star. Primary is a bottleneck

XDCR Capabilities	Couchbase	MongoDB
Active-Active	Supported	No support
Conflict Resolution	Yes - most write wins or last write wins	No conflict resolution. Only one primary supported
Setup and Configuration	Easy configurability with intuitive UI and CLI. No design challenge like no. of shards, policies, etc.	Replica set distribution is tricky and can be painful as the replica sets increase
Filtering to replicate subsets	Key-based filtering to replicate subsets of data using doc key IDs	No filtering supported

Conclusion

MongoDB provides a long list of checkbox features, but many fail to work in concert with each other, leading to a database that cannot scale, perform, nor adapt to meet today's enterprise requirements. Ultimately, MongoDB is best suited for flexible data access where low latency, high throughput, multiple access patterns, geographic replication, or offline access are not required.

Couchbase, on the other hand, is routinely used for caching layers, sources of truth, and systems of record across high-scale and high-flexibility use cases, including offline-first mobile applications. By design, Couchbase is accessed and managed through a consistent set of APIs, and scaled, upgraded, and diagnosed as a single unit, making Couchbase a complete database platform that not only addresses the needs of today, but offers the flexibility to adapt to the needs of tomorrow.

Learn more

To learn more, contact your Couchbase sales representative today or visit:

couchbase.com | couchbase.com/downloads

Couchbase's mission is to be the data platform that revolutionizes digital innovation. To make this possible, Couchbase created the world's first Engagement Database to help deliver ever-richer and ever-more-personalized customer and employee experiences. Built with the most powerful NoSQL technology, the Couchbase Data Platform was architected on top of an open source foundation for the massively interactive enterprise. Our geo-distributed Engagement Database provides unmatched developer agility and manageability, as well as unparalleled performance at any scale, from any cloud to the edge.