WHITE PAPER

5 Use Cases that Showcase Why You Need Active-Active Redis Enterprise

Roshan Kumar, Redis

CONTENTS

Executive Summary

For geographically distributed applications, active-active database architecture is fast becoming a must-have. Under this superior replication technique offered by Redis Enterprise 5.0, all database instances are available for read and write operations, providing unprecedented availability, performance, and resource utilization. Additionally, built-in conflict resolution through its CRDT-based architecture significantly simplifies application development when trying to achieve local latencies and high performance across distributed datasets.

As the only NoSQL database on the market to leverage CRDT-based active-active architecture, Redis Enterprise offers groundbreaking impact for use cases in ecommerce, IoT, metering, personalization, fraud detection, and more—where high availability, resilience for failure scenarios, and tracking of geo-distributed, simultaneous events are paramount.

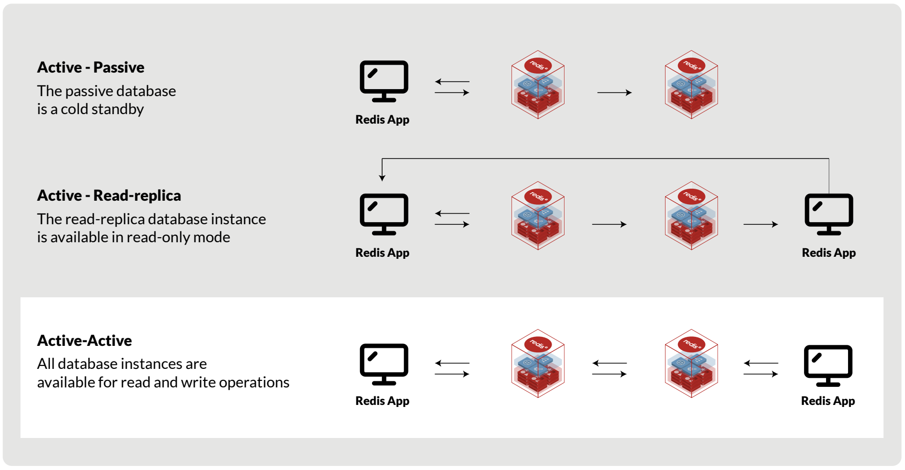
In this white paper, we'll overview CRDT-based active-active architecture, its unique benefits, and the use cases in which it offers decided advantages to your geo-distributed application. Additionally, we'll compare Redis Enterprise's active-active implementation to traditional replication techniques such as active-passive, as well as to other types of active-active implementations available on the market today.

How Does Active-Active Redis Enterprise Work?

Under active-active architecture, all database instances are available for read and write operations and are bidirectionally replicated. It offers local latency on read and write operations, regardless of the number of geo-replicated regions and their distance from each other. It enables seamless conflict resolution ("conflict-free") for simple as well as complex data types. Even if the majority of geo-replicated regions are down, the remaining geo-replicated regions are uninterrupted and can continue to handle read and write operations.

For globally distributed applications, active-active databases provide unprecedented levels of availability, performance, consistency, and resource utilization (no passive or underutilized servers) for a maximum return on investment.

The following figure contrasts active-active replication with other common replication techniques, all of which are supported by Redis Enterprise.



Active - Passive
The passive database is a cold standby
Redis App
Active - Read-replica
The read-replica database instance is available in read-only mode
Redis App
Redis App
Active-Active
All database instances are available for read and write operations
Redis App
Redis App

Figure 1: Redis Enterprise 5.0 supports three replication techniques

CRDT Approach

Redis Enterprise achieves active-active replication using CRDTs (conflict-free replicated data types). CRDTs provide strong eventual consistency through built-in smart conflict resolution. According to Wikipedia, a conflict-free replicated data type (CRDT) is "a data structure which can be replicated across multiple computers in a network, where the replicas can be updated independently and concurrently without coordination between the replicas, and where it is always mathematically possible to resolve inconsistencies which might result."[1]

[1]Conflict-free replicated data type. In Wikipedia, The Free Encyclopedia.

Benefits of CRDT-based Active-Active Architecture

CRDT-based active-active architecture offers considerable advantages to your database platform when compared to other active-active methods such as LWW (last writer wins), quorum based replication, synchronized active-active, and others. As the only active-active NoSQL database to leverage CRDT architecture, Redis Enterprise delivers the following unique benefits:

- High performance
- Built-in conflict resolution
- Strong eventual consistency
- Simplified app development
- More efficient resource utilization

High Performance

A CRDT-based active-active architecture, as a result of its ability to update all master servers independently and concurrently, guarantees local, sub-millisecond latencies for both read and write operations.

In contrast, databases that rely on standard active-passive replication can geo-distribute reads, but writes must be sent to the only active replica, thereby incurring higher latencies. Active-active databases that don't use CRDTs are also problematic because the inefficiencies of their consistency models impose WAN latencies for either read or write operations (e.g. quorum-based replication must wait for a majority of servers to vote to execute a read or write transaction).

Built-in Conflict Resolution

CRDTs provide a mathematical model for handling conflicting writes. Their goal is to provide well-defined and automatic conflict resolution behaviors for common use cases. When combined with Redis data types and commands, CRDTs detect the "developers intent" during conflict resolution. For example, when using Redis commands such as INCR (increment), Redis CRDTs behave in a way that allow you to build distributed counters. When using a Set data type with SADD or SREM commands, Redis CRDTs use a "Add-Wins with Observed-Remove Set" behavior defined in CRDTs. The use of CRDTs in Redis Enterprise allows all databases to converge automatically to the same state with strong eventual consistency.

Strong Eventual Consistency

Strong eventual consistency guarantees that any two clusters receiving the same set of updates will converge to the same state. Redis Enterprise is the only NoSQL database to offer strong eventual consistency. Other NoSQL databases offer only eventual consistency, which guarantees that updates will be observed eventually, but does not guarantee that the clusters receiving those updates will resolve to the same state. CRDTs are the most efficient approach to ensuring strong eventual consistency.

Simplified Application Development

The built-in conflict resolution that comes with Redis CRDTs vastly simplifies and accelerates application development and global deployment. Developers simply choose the data types and Redis CRDTs automatically resolve conflicts using the well-defined rules for each data type and command. To achieve this effect, Redis CRDTs keep additional metadata per type, which is later used in bidirectional replication to synchronize all participating clusters. Non-CRDT-based active-active databases, in contrast, require developers to manage tedious conflict complexities in the application layer.

More Efficient Resource Utilization

Under the active-passive replication model, passive replicas are vastly underutilized in their roles as cold standbys. With active-active, load is better distributed (and more throughput is possible) for a maximum return on investment.

Use Cases that Highlight the Need for Active-Active Redis Enterprise

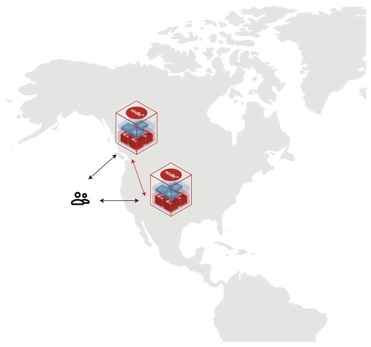
CRDT-based active-active databases provide unprecedented levels of availability and high performance. These attributes are critical for geo-distributed applications that depend upon providing cutting-edge and lightning-fast customer experiences.

We've identified some use cases that showcase how active-active Redis Enterprise can set your application apart from the competition, and ensure your company never misses a beat in today's high-speed landscape.

- User session migration across data centers
- Node failure handling without data loss
- Immediate data consolidation
- Load distribution
- Geo-distributed application functionality

User Session Migration Across Data Centers

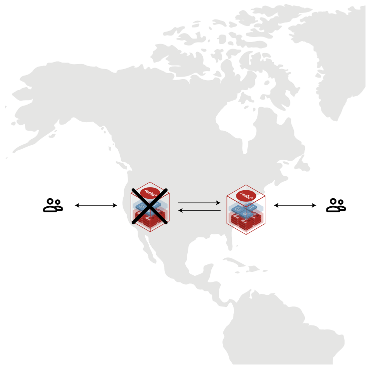
In order to achieve optimal user experience, there are times when it's ideal to shift users to other data centers. For example, a user entering the range of a new, closer data center will experience lower latencies if they are able to establish a connection to the most proximal data center. Or, there may be an advantage to shifting to the next nearest data center if the current, closest data center has reached or exceeded capacity. But the ability to shift users among data centers is a challenging ask of your database if users are mid-session.



With active-active Redis Enterprise, mid-session users get routed from one data center to another in real time, seamlessly. All session states are preserved during the transition and both databases converge automatically to the same state with strong eventual consistency. In contrast, under an active-passive model, users remain connected to the original data center, experiencing ever-increasing latencies as they move further away.

Node Failure Handling without Data Loss

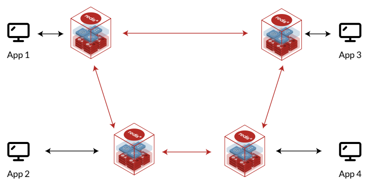
As with user session migrations, node failure handling is seamless under active-active Redis Enterprise; your users, along with their current session data, are automatically routed to the next nearest server. When the original server comes back up, connectivity with the original server is automatically resumed in order to once again provide local latencies. Additionally, all data is immediately consolidated, without data loss, between the two servers.



It's worth noting that Redis Enterprise already delivers this scenario, even with active-passive replication. The difference, however, is that in an active-active scenario, the replica can be used for more that just reads; it can be used to handle local session writes and thus deliver better overall resource utilization.

Immediate Data Consolidation

In many environments, especially those that employ microservices, apps have their own separate database instances. However, these instances share common datasets (user data is a typical example) which must remain consistent among the various apps. With active-active Redis Enterprise, data is resolved and consolidated immediately among the various database instances to ensure that all services are operating with real-time data (e.g. a customer's print magazine subscription renewal instantly cascades into online privileges to the publisher's electronic media). Traditional solu-

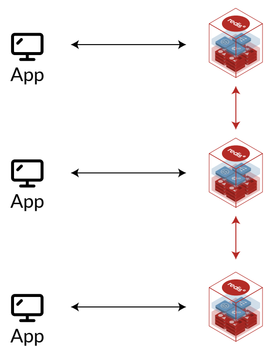


Microservices Environment

tions consolidate data manually or through the execution of periodic jobs, which means that a considerable amount of time could pass before data is consolidated among database instances.

Load Distribution

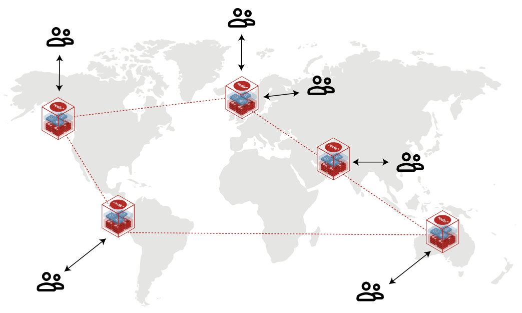
The ability to distribute load across multiple servers is critical for accommodating high volumes of traffic and data streaming operations. With active-active Redis Enterprise, all database instances are available for read and write operations, optimizing resource use, maximizing throughput, and minimizing response time.



Geo-distributed Application Functionality

It's common for app functionality to be distributed geographically. For example, social media apps have globally distributed counters for tracking engagements such as likes, shares, or retweets. As with social media apps, apps that power auction bidding boards or gaming lead-erboards must also consolidate events happening simultaneously across multiple regions and data centers. Another example is that of applications that may be tracking bids for stocks, options or other items in different currencies across the world. Application users need to know instantly who is in the lead.

In these cases, active-active Redis Enterprise is an ideal choice for its ability to immediately resolve and consolidate data among multiple databases. An alternative to ensuring consistent data in these types of scenarios is to employ a central database, but having only one database accepting writes results in high availability and latency issues.



Comparing Other NoSQL Active-Active Implementations

As mentioned previously, Redis Enterprise is the only active-active NoSQL database to offer strong eventual consistency through its CRDT architecture. Other active-active databases offer only eventual consistency, which guarantees that up-dates will be observed eventually, but does not guarantee that the masters receiving those updates will resolve to the same state.

	Redis Enterprise	DataStax	Couchbase
Architecture	CRDT	Quorum based: a majority of the servers must ack a write	LWW: prone to consolidation errors if the timestamps are out of sync
Local latencies	Yes	No	Yes
Data consistency	Strong Eventual	Eventual	Eventual
Availability	Always-on	Not 100%. Need maximum nodes to be communicating with each other.	Not 100%. Requires communication with NTP Server to be always on
Suitable for high-frequency writes	Yes	No	No

Figure 2: Redis Enterprise's active-active implementation as compared to DataStax and Couchbase

Other drawbacks of non-CRDT active-active databases include:

- WAN latencies for either read or write operations due to the inefficiencies of their consistency models (e.g. quorum-based active-active replication must wait for a majority of servers to vote to execute a read or write transaction).
- The requirement to use crude conflict management tools (e.g. Last Writer Wins (LWW)), which do not provide the fine-grained conflict resolution required by modern applications that use complex data types.
- The need for developers to resolve data conflicts at the application level

Conclusion

Active-active Redis Enterprise supports the responsiveness, scalability, and geo-distributed functionality required by today's modern applications—all while maintaining the high performance and simplicity that Redis is known for. Its built-in replication and conflict-free resolution features translate into real-world advantages across a wide variety of use cases and application scenarios, making a compelling case for building your geo-distributed applications in Redis Enterprise.

(For a deeper dive into the mechanics of a CRDT-based active-active Redis Enterprise deployment, see our white paper titled Under the Hood: Redis CRDTs.)