Running Active-Active Geo-Distributed Redis on VMware Multi-Cloud with VMware NSX

vmware

Table of Contents

Overview

Introduction

Redis Enterprise Software is a self-managed data platform that unlocks the full potential of Redis at enterprise scale. In Redis Enterprise, Active-Active geo-distribution is based on CRDT ((conflick-free replicated data types) technology. With Active-Active databases, applications can read and write to the same dataset from different geographical locations seamlessly and with latency less than one ms (millisecond) , without changing the way the application connects to the database.

We already validated Redis on Tanzu Kubernetes Grid. This paper provides the guidelines of Active-Active geo-distribution Redis on VMware vSphere® with multi-cloud using VMware NSX®. And it covers the generic design and deployment, functional, and performance testing.

Technology Overview

The technology components in this solution are:

- VMware Cloud™ on AWS
- VMware NSX
- Redis Enterprise Software
- Benchmark Tool: Memtier_Benchmark

VMware Cloud on AWS

VMware Cloud on AWS brings VMware's enterprise class Software-Defined Data Center software to the AWS Cloud and enables customers to run production applications across VMware vSphere-based private, public and hybrid cloud environments, with optimized access to AWS services. Jointly engineered by VMware and AWS, this on-demand service enables IT teams to seamlessly extend, migrate, and manage their cloud-based resources with familiar VMware tools –minimizes the hassles of learning new skills or utilizing new tools.

VMware Cloud on AWS integrates VMware's flagship compute, storage, and network virtualization products (VMware vSphere, VMware vSAN, and VMware NSX) along with VMware vCenter® management as well as robust disaster protection, and optimizes it to run on dedicated, elastic, Amazon EC2 bare-metal infrastructure that is fully integrated as part of the AWS Cloud. This service is delivered and supported by VMware and its partner community. This service is sold by VMware, AWS, and their respective partners.

VMware NSX

VMware NSX provides an agile software-defined infrastructure to build cloud-native application environments. NSX focuses on providing networking, security, automation, and operational simplicity for emerging application frameworks and architectures that have heterogeneous endpoint environments and technology stacks. NSX supports cloud-native applications, bare metal workloads, public clouds, and multiple clouds.

Redis Enterprise Software

Redis Enterprise Software is a self-managed and enterprise-grade version of Redis. It maintains the simplicity and high performance of Redis, while adding many enterprise-grade capabilities including:

- Linear scalability
- High availability
- Geo-replicated, active-active data distribution
- Predictable performance

Active-Active Geo-distributed Redis is achieved by implementing CRDTs in Redis Enterprise using a global database that spans multiple clusters. CRDT can offer local latency on read and write operations, even if most of the geo-replicated regions in a CRDT are down, the remaining geo-replicated regions are uninterrupted and can continue to manage read and write operations, ensuring business continuity.

Benchmark Tools

Memtier_benchmark: It is a command line utility for load generation and benchmark NoSQL key-value databases. It supports both Redis and Memcache protocols, read-write ratio, random payload, random or ranged key expiration and Redis cluster mode.

Solution Configuration

This section introduces the resources and configurations:

- Architecture diagram
- Hardware resources
- Software resources
- VM configurations
- Network configuration

Architecture Diagram

In our solution, we deployed Active-Active Geo-distributed Redis on VMware vSphere with multi-cloud deployment using VMware NSX. On the private site, we deployed a Redis cluster and it also contains the VMware NSX Edge nodes.

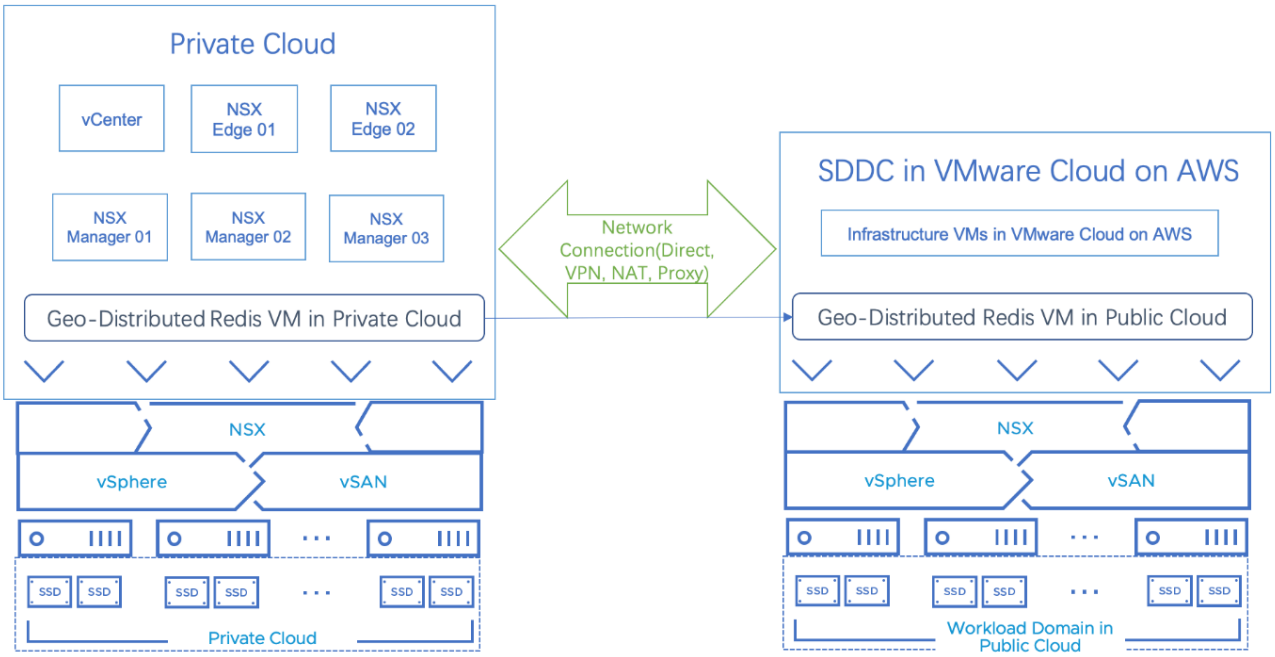


Figure 1. Redis Enterprise Software Running on VMware vSphere and Multi-Cloud

Hardware Resource

In the private cloud environment, we validated the solution with a four-node vSphere cluster. Table 1 shows the specification of each node.

Table 1. Hardware Configuration in Private Cloud

PROPERTY	SPECIFICATION
CPU	2 x Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz, 28 core each
RAM	480GB
Network adapter	2 x Intel 10Gb Ethernet Controller X550
Storage adapter	1 x Dell HBA330 Mini Adapter
Disks	Cache - 2 x 1.46TB NVMe Capacity - 8 x 1.75TB SSD

We used three Amazon EC2 'i3en.metal' instances when deploying VMware Cloud on AWS SDDC. Each server has the following configuration.

Table 2. Hardware Configuration in VMware Cloud on AWS

PROPERTY	SPECIFICATION
Server model name	Amazon EC2 i3en.metal-2tb
CPU	2 x Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz, 24 core each
RAM	768GB
Network adapter	1 x Amazon, Inc Elastic Network Adapter, 100Gbit/s
Storage adapter	None
Storage	Total raw cache capacity: ~6.36TiB (~7TB) Total usable storage capacity: ~45.84TiB (~50TB)

Software Resource

We used the following software components in this solution.

Table 3. Software Resources

SOFTWARE	VERSION	PURPOSE
VMware vSphere	8.0.1	VMware vSphere is a suite of products: vCenter Server and ESXi.
Redis Enterprise Software	7.2	Redis is a popular NoSQL database and an in-memory data store supporting multiple abstract data structures. Redis Enterprise is a self-managed and enterprise-grade version of Redis.
VMware Cloud on AWS SDDC	1.16v5	The software stack of VMware Cloud on AWS.

VM Configuration

We configured the VMs as follows in Table 4 in this solution.

Table 4. VM Configuration

VM Role	vCPU	Memory (GB)	Storage (GB)	VM Count
NSX Manager	6	24	200	3
NSX Edge nodes	4	16	200	2
Redis VM	16	64	250	2

Network Setting

Active-Active Redis is a data resiliency architecture that distributed the database information over multiple data centers via independent and geographically distributed clusters and nodes. It is achieved by implementing CRDTs in Redis Enterprise using a global database that spans multiple clusters.

In our solution, we ran one CRDT database instance on the private data center, connecting to the other CRDT database instance in VMware cloud on AWS, we need to expose the network in VMware Cloud on AWS to the public internet. We used ‘VPN’ configuration from the VMware Cloud on AWS SDDC Manager portal. It required a public IP address in VMware Cloud on AWS

and another public IP address in the private data center and configured as the VPN endpoint. A VPN connection was established between on-premises and VMware Cloud on AWS.

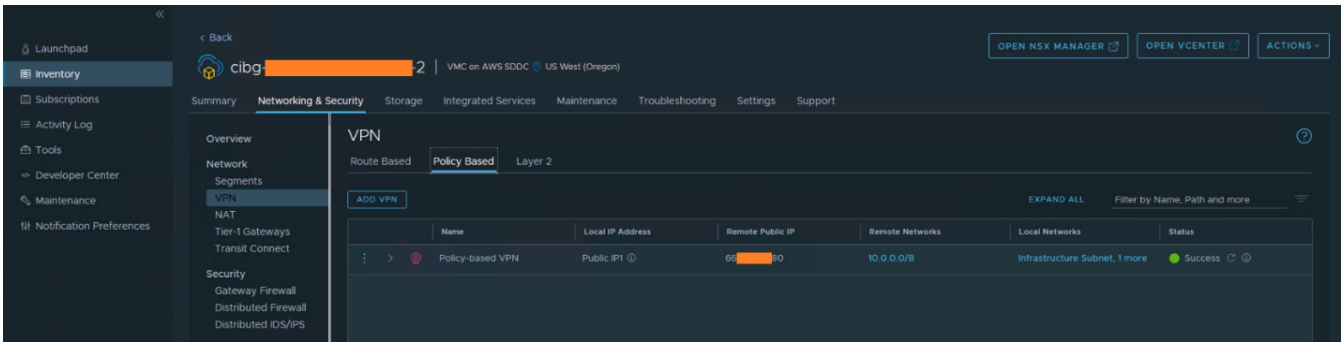


Figure 2. Configure VPN for the Network Connection from the Local Data Center to Public VMware Cloud on AWS

In the networking and security overview page of the NSX Manager UI, we can check whether the VPN configuration or other NAT configurations were set properly.

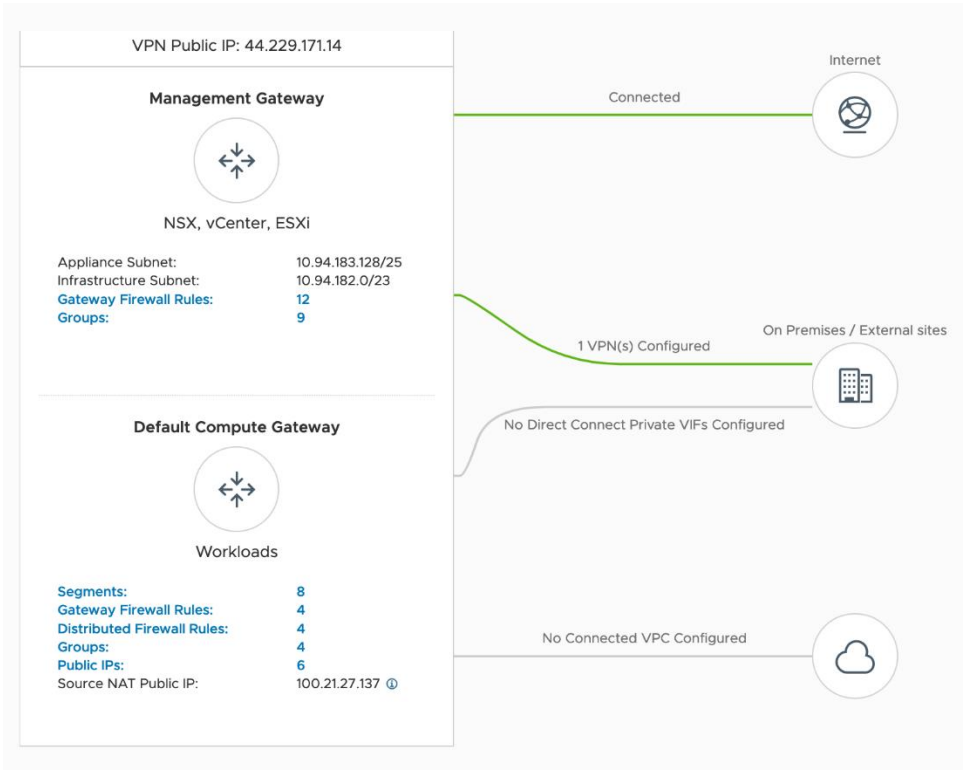


Figure 3. The Networking Overview of the SDDC in VMware Could on AWS

Solution Deployment

This solution testing highlights the deployment and performance with Active-Active Geo distributed Redis on VMware vSphere with multi-cloud using VMware NSX.

Application Deployment

In the previous paper, we deployed Redis Enterprise Cluster on the Tanzu Kubernetes Grid workload cluster. You can follow the steps to deploy the participating clusters.

The participating Redis Enterprise Software clusters that host the instances can be in distributed geographic locations. Every instance of an Active-Active database can receive write operations and all operations are synchronized to all of the instance without conflict. With the Geo-distributed Redis database created, you can check the status of the database from the admin console of the cluster.

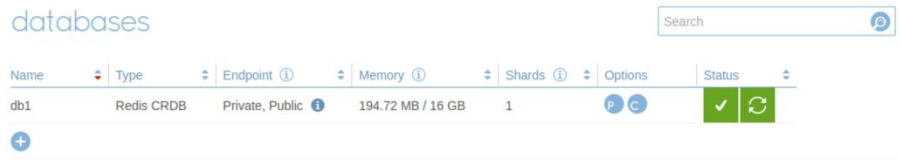


Figure 4. Redis CRDT Database

Perform the following steps for Redis Geo-distributed Database:

1. Create two participating clusters with a dedicated user account with the admin role.

2. Create Active-Active database and add participating clusters, make sure that the participating clusters are configured in the DNS. You can configure the memory size and the replica of a database.


```
10.159.230.194  cluster2.local
10.72.255.10    cluster1.local
10.72.255.10   redis-12000.cluster1.local
10.159.230.194  redis-12000.cluster2.local
```

3. Wait for the status of database to be active and synced. Run on one cluster and establish a connection to an endpoint on the other cluster.

```
ubuntu@vmc-ubuntu1:~$ sudo netstat -anp | grep syncer
[sudo] password for ubuntu:
tcp        0      0 10.72.255.10:50274      10.159.230.194:12001    ESTABLISHED 368398/rl_syncer
```

Redis CRDT and Conflict Resolution Examples

Each CRDT database instance separately maintains a vector clock for each dataset object. This vector clock is updated upon any update operation at the instance level or when another update operation for the same object arrives from another CRDT database instance. The following process is carried out separately at each CRDT database instance upon receiving an update operation from another instance.

In many concurrent update state cases, an update can be processed completely conflict-free based on the properties of the applicable data type. All operations are either associative or idempotent and are conflict-free when the data type is Set (mapped to a CRDT's Add-Wins Observed-Removed Set), and the concurrent updates are ADD operations. **Example:** In the fraud detection, when the application is tracking dubious events associated with an ID or credit card. The Redis Set cardinality associated with the ID or credit card is used to trigger an alert if it reaches a threshold.

Table 5. Example 1: Conflict-free Concurrent SADD Operations to a Set

Time	Instance A	Instance B
t1	SADD key1 A	SADD key1 B
t2	Sync	
t3	SMEMBERS key1 => A B	SMEMBERS key1 => A B

A conflict-resolution algorithm should be applied in case of concurrent updates in a non-conflict-free data type. We have used the Last Write Wins (LWW) approach to resolve such situations by leveraging the operation timestamp as a tiebreaker. Assume Instance A's timestamp is always ahead of the other instances' timestamps (for example, in the case of a tiebreaker, Instance A always wins). This ensures behavior is eventually consistent. **Example:** A password is changed for a user account accessed by multiple geographically distributed entities. In this case, the change would log out other users, which might be the right behavior for license enforcement scenarios.

Table 6. Example 2: Last Write Wins for Strings

Time	Instance A	Instance B
t1	SET key1 "value1"	

t2		SET key1 “value2”
t3	sync	
t4	GET key1 => “value2”	GET key1 => “value2”

Table 7. Example 3: Add/Update Wins over Delete for Strings

Time	Instance A	Instance B
t1	SET key1 “value1”	
t2	sync	
t3	APPEND key1 “hello”	
t4		DEL key1
t5	sync	
t6	GET key1 => “value1 hello”	GET key1 => “value1 hello”

Note: APPEND is an ‘update' operation treated like ‘add' and therefore wins the DEL operation.

Redis Enterprise Cluster Performance Validation

We used built-in memtier_benchmark tool to generate loads for Redis. In the performance testing, we provisioned a Geo-distributed database over 2 clusters on two regions, each Redis cluster generate load with memtier_benchmark.

Scenario 1: We used the memtier_benchmark tool to operate a performance benchmark of Redis clusters on two regions.

- Use a SET/GET/3:7 W:R ratio
- Use the object size of 256 bytes:
  - Database persistence is kept to default.
  - Replication is disabled.
  - One database instance is used, which is composed of 16 vCPUs and 64GB of RAM.

- Execute 10M commands with 4 command per requests (--pipeline=4)
- Create 3 clients with 24 working threads each

```
./memtier_benchmark --ratio 3:7  -t 24 -c 1 --test-time 180 --distinct-client-seed -d 256 -s redis-12001.cluster2.local -p 12001 --key-maximum=10000000 --hide-histogram --pipeline 4
```

You would get the output as shown in Table 8.

Table 8. Baseline Performance Results

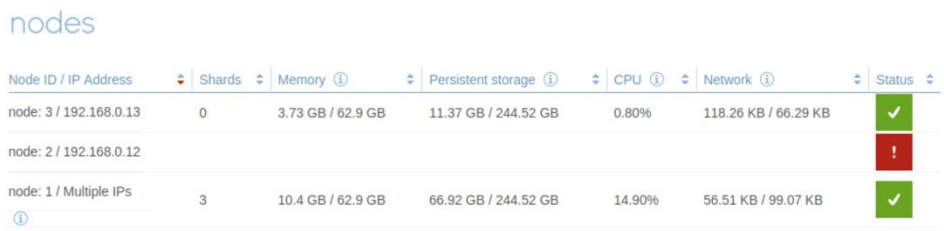
Command	Pipeline	Ops/s	Latency (ms)
SET	4	299,252	2.71
GET	4	1,240,519	0.23
W:R 3:7	4	714,942	4.7

Memtier_benchmark has several configuration options that can be tuned to better emulate the workload on your Redis server, in addition to offering cluster support, to simulate real-world environments, Memtier_benchmark will test for GET and SET requests and mixed request on a ratio of 3:7. This is more representative of a common web application using Redis as a database or cache.

Scenario 2: Power off one instance of the Redis Cluster on one region to observe the impact of Active-Active Redis Database.

Table 9. Performance Results with High Available

Database	Cluster1 Throughput	Cluster1 Latency	Cluster2 Throughput	Cluster2 Latency
3:7	273,894 ops/s	P99=4.7 ms	441,048 ops/s	P99=4.1 ms
Power off one instance	265,448 op/s	P99=4.79ms	449,691 ops/s	P99=4.22 ms



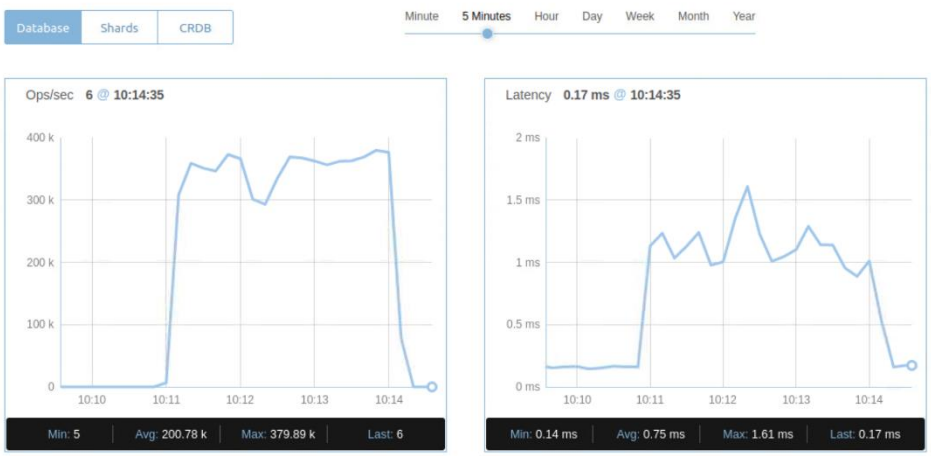


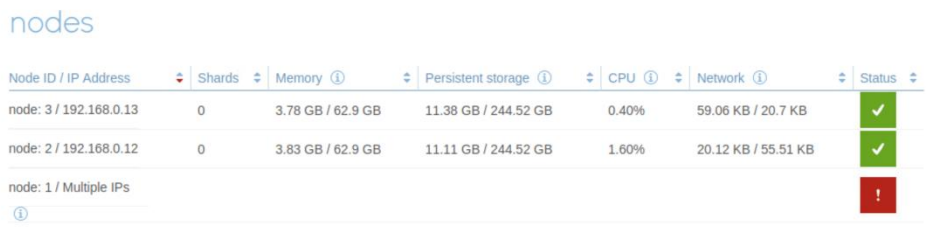
Figure 5. Cluster 2 Performance Metric

Figure 6. Cluster 1 Performance Metrics

When we powered off instance 1 during the performance benchmark, you can check the status of the node from the admin console of the cluster. We noticed clients connected two clusters working normally. The performance metric of two clusters were not affected.

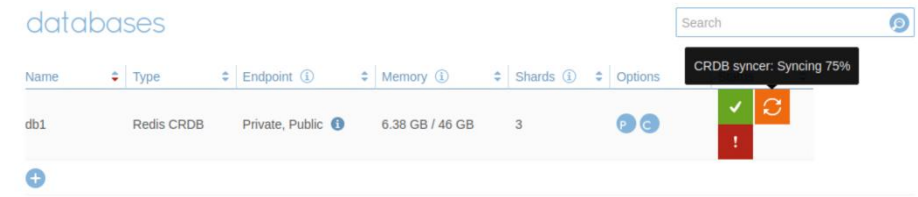
Scenario 3: Power off all instances of one site or one instance hold all shards

In our testing, all the master shards are located on instance 2. When we powered off instance 1 of the Redis cluster 2 during the performance benchmark, you can check the node status is down from the admin console of the cluster.

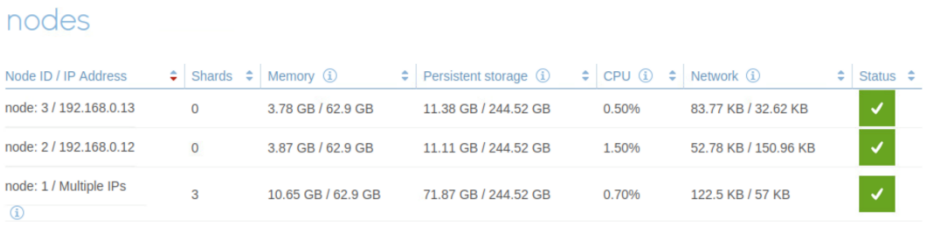


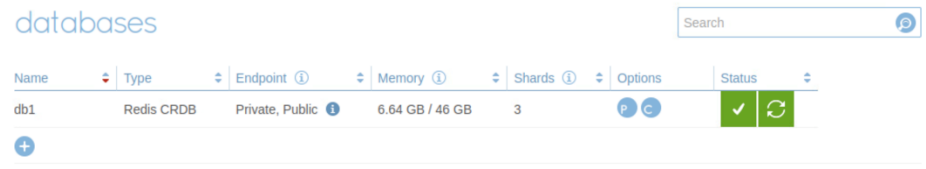
When the clients connected to the cluster2 was interrupted, the performance benchmark worked normally on cluster 1.

```
[RUN #1 100%, 180 secs]  0 threads:    80232355 ops,  502198 (avg:  445733) ops/sec, 143.39MB/sec (avg: 127.27MB/sec),  0.76 (avg:  0.86) msec latency

24        Threads
4         Connections per thread
180       Seconds


ALL STATS
========================================================================================================================
Type         Ops/sec     Hits/sec   Misses/sec    Avg. Latency     p50 Latency     p99 Latency   p99.9 Latency       KB/sec
------------------------------------------------------------------------------------------------------------------------
Sets       133720.74          ---          ---         0.86834         0.79100         2.36700         5.72700     39553.24
Gets       312013.13    312013.13         0.00         0.85679         0.77500         2.35100         5.59900     98766.82
Waits           0.00          ---          ---             ---             ---             ---             ---          ---
Totals     445733.87    312013.13         0.00         0.86025         0.78300         2.35100         5.63100    138320.06
```



When powered on the instance 1, we noticed that database started to sync. Waiting for a while, the node status turned to be active, and the database was healthy and synchronized.





Best Practice

The following recommendations provide the best practices and sizing guidance to run Redis Enterprise Clusters on VMware multi-cloud.

- Network considerations:
  - Separate physical network using different network ranges and VLAN IDs for management, vSAN, VM network, and VXLAN VTEP network.
  - For NSX Data Center for vSphere design best practices for workload domain, see VMware NSX for vSphere Network Virtualization Design Guide.
  - Set up proper networking connections between the local data center and the public VMware Cloud on AWS data center; for example, set up VPN, AWS Direct Connect or NATs for the bi-directional traffic between them.
- vSAN Storage:
  - For different types of workloads, create a different vSAN storage policy for management and workloads. Then, bind them to different storage classes. Specifically, on VMware Cloud on AWS make sure to use a default management policy to stay compliant with VMware Cloud on AWS SLAs. If using custom policies, follow the SLA guidelines when defining the RAID level and FTT for the workloads.
- Redis Enterprise Software:
  - In a production environment, you must have enough resources to manage the load on the database and recover it from failures:
    i. At least three nodes are required to support a reliable and high available deployment that manages process failure and node failure. And it must be an odd number of nodes.
    ii. Redis Enterprise Software can run multiple Redis processes or shards on the same core without significant performance degradation. It is recommended to configure at least eight cores, 30GB RAM per node. Make sure the utilized CPU load of the cluster nodes is under 80% of the CPU, 30% of the RAM is available on each node at any given time.
    iii. It is recommended to run over 10Gb interface network used for processing application requests, inter-cluster communication, and storage access. Active-Active databases require FQDNs.
  - Database clustering allows spread the load over multiple servers. When the dataset is large enough (more than 25GB) or performing CPU-intensive operations, it is recommended to enable clustering to create multiple shards of the database and spread the requests and load across nodes.
  - The location of primary and replica shards on the cluster nodes can affect the database and node performance. The shard placement policy helps to maintain optimal performance and resiliency:
    i. Dense shard placement policy is recommended for Redis on RAM databases to optimize memory resources.

ii. Sparse shard placement policy is recommended for Redis databases to optimize disk resources.

- It is recommended that all active-active databases use replication for the best inter-cluster synchronization performance. In addition, you can enable replica HA to ensure that the replica shards are high available for this synchronization.

Conclusion

Active-Active Geo-distribution Redis Database running VMware vSphere and multiple-cloud support instant responses for businesses. It guaranteed local latencies for both read and write operations, utilizing consensus free protocols to maintain consistency. Built-in conflict resolution for simple and complex data-types can simplify app development and global deployment. Moreover, Safer cross geo failover, with automatic and intelligent sync between active databases can avoid incorrect overwrites and loss of state. Thus, the Geo-distributed Redis Database running on VMware Multi-Cloud provides a proven flexible and secure solution. This increased flexibility optimizes infrastructure costs and database performance for use case. By utilizing Active-Active technology Redis Enterprise allows organizations to smoothly migrate their application to the cloud or between clouds.

Additional Resources

For more information about Redis Enterprise Software and VMware Tanzu Kubernetes Grid, you can explore the following resources:

- VMware vSphere
- VMware vSAN
- Redis Enterprise Software

About the Author and Contributors

Yimeng Liu, Senior Technical Marketing Manager in the Workload Technical Marketing Team of the Cloud Infrastructure Big Group, wrote the original version of this paper. The following reviewers also contributed to the paper contents:

- Catherine Xu, Senior Manager of the Workload Technical Marketing Team in VMware

vmware by Broadcom