TencentDB for Redis

Product Introduction

Product Documentation

Contents

Product Introduction
Overview

Last updated : 2020-07-07 10:38:51

TencentDB for Redis is a cache database provided by Tencent Cloud based on the Redis protocol that features high availability, reliability, and flexibility. Compatible with Redis 2.8, 4.0, and 5.0 protocols and available in both standard and cluster architectures, it supports up to 4 TB of storage capacity and tens of millions of concurrent requests, meeting the needs of different scenarios such as caching, storage, and computing.

Product Features

- Master/slave hot backup: master/slave hot backup, automatic failure monitoring, and automatic disaster recovery are supported.
- Data backup: standard and cluster architectures support data persistence, providing cold backup and rollbdack.
- Elastic scaling: instance capacity can be flexibly expanded or reduced. Plus, the nodes and replicas can be scaled in or out.
- Network protection: VPC is supported for improved cache security.
- Distributed storage: your data is distributed across multiple physical machines, helping you get rid of standalone capacity and resource constraints.

Product Editions

TencentDB for Redis is available in both standard and cluster architectures for your choice based on your actual performance requirements. The standard architecture features higher compatibility, but its performance is restrained to one single node. The cluster architecture is not as compatible as the standard one, but it can be scaled out to support up to tens of millions of concurrent requests.

Instance Type	Number of Replicas	Read/write Separation	Description
Memory Edition (standard architecture)	0-5	Supported	Compatible with Redis 2.8, 4.0, and 5.0 protocols. The 2.8 Edition offers Redis shards with no replica and is a cost-effective solution for pure cache use cases. The 4.0/5.0 Edition offers Redis shards with 1–5 replicas. The capacity and replica can be scaled up seamlessly with no service interruption.
Memory Edition (cluster architecture)	1-5	Supported	Compatible with Redis 4.0 and 5.0 protocols. The cluster architecture supports 3–128 shards with a maximum of 4 TB storage capacity and tens of millions of QPS. Master/slave hot backup, real-time data synchronization, and failover in a matter of seconds are supported.

Strengths

Last updated : 2020-08-18 16:35:47

Wide Variety of Product Specifications

TencentDB for Redis offers a choice of 0.25 GB–4 TB capacity specifications available in Standard and Cluster Editions.

Elastic Scaling

An instance can be scaled quickly in the console without having to stop the services. No operations are required at your side.

Ultra-high Performance

The Standard Edition has a performance of up to 100,000+ QPS, and the Cluster Edition supports tens of millions of QPS. Their ultra-high performance can perfectly meet the needs in most business scenarios ranging from gaming, mobile apps to advertising and ecommerce.

Rich Monitoring Capabilities

TencentDB for Redis boasts a rich set of metrics and alarms that are monitored and visually displayed for clear insights into the data, helping you identify risks before they appear and troubleshoot problems quickly.

Convenient DTS Service

TencentDB for Redis supports cold and hot data migration in various self-created database environments such as self-created Tencent Cloud-based, VPN-based, Direct Connect-based, and IDC-based environments.

Automatic Disaster Recovery

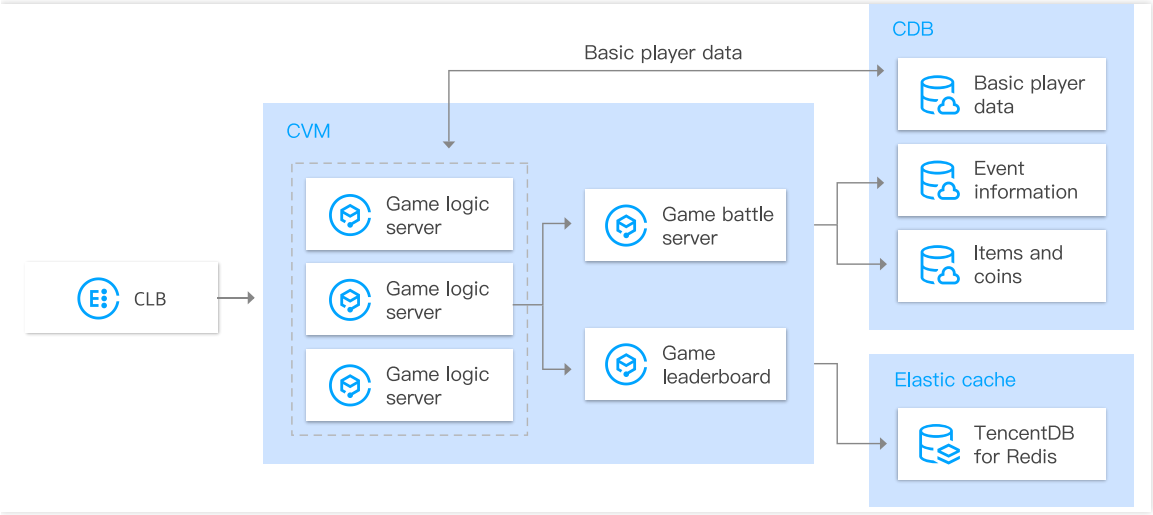
TencentDB for Redis adopts a primary/replica hot backup architecture. In case of failure of the primary server, the access can be switched to the replica in a matter of seconds. The switch process does not require any operations at your side, reducing the labor and time costs of developing a primary/replica system architecture.

Application Scenarios

Last updated : 2020-07-02 17:51:52

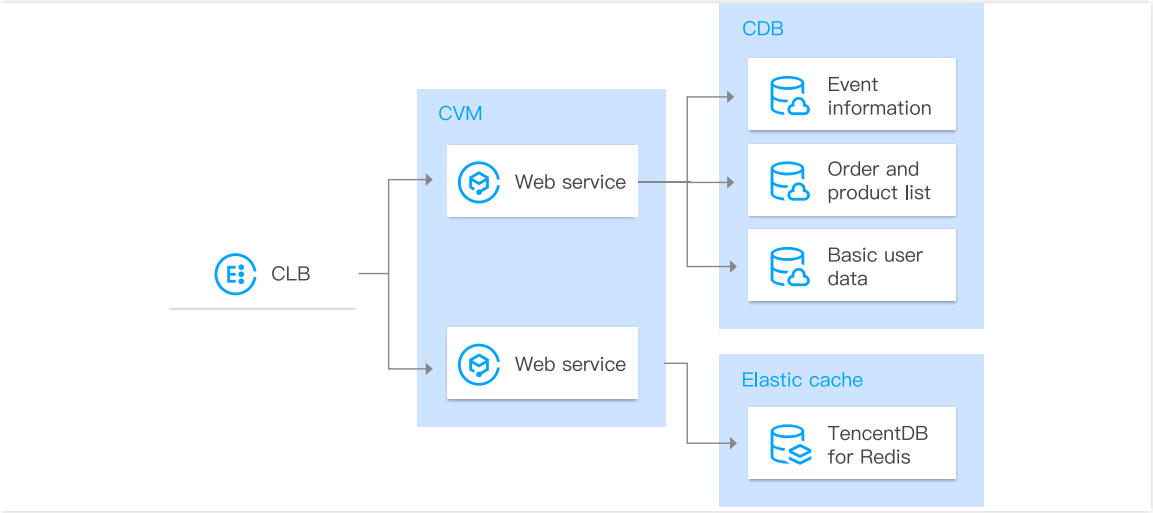
Gaming

In the gaming industry, non-role data such as leaderboard data can be stored in TencentDB for Redis for quick access, and the native SortedSet data type of Redis can readily help develop player data.



Internet and Apps

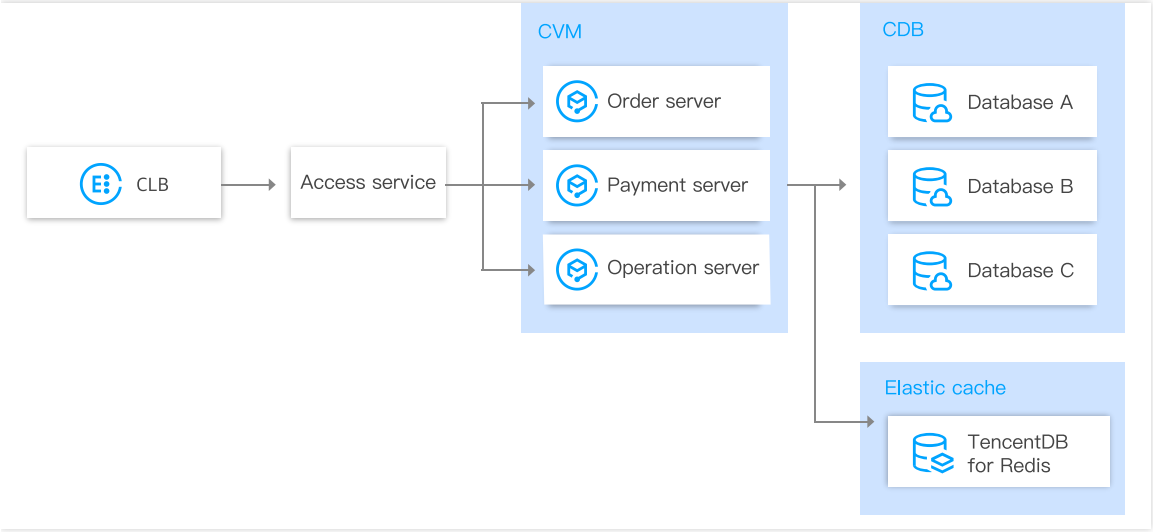
In internet and app businesses, basic user data can be cached into TencentDB for Redis to improve the read performance. Further, static images and resources can be cached there too to speed up application loading.



Ecommerce Display

In ecommerce display, data such as product images and recommendations can be stored in TencentDB for Redis for quick access. In addition, the high QPS performance (up to tens of millions) of Redis can sustain high concurrent requests, making it ideal for large-scale campaigns (e.g.,

flash sales).



Regions and Availability Zones

Last updated : 2020-09-28 15:32:37

TencentDB for Redis is available in various regions. It can be deployed wherever CVM instances are deployed.

Network Description

- Tencent Cloud services in the same region can communicate with each other over the private network.
- A CVM instance in a VPC can access a TencentDB for Redis instance on the classic network in a different AZ in the same region only after a subnet is configured and a VPC IP is assigned.
- Tencent Cloud services on the classic network in different regions cannot communicate with each other over the private network. They can communicate across VPCs only after a peering connection is configured.

Note :

When you purchase TencentDB for Redis, we recommend you select the same region as your CVM instance to reduce access delay.

Regions and Availability Zones

China

Region	Availability Zone	ZoneId
South China (Guangzhou) ap-guangzhou	Guangzhou Zone 1 ap-guangzhou-1	100001
	Guangzhou Zone 2 ap-guangzhou-2	100002
	Guangzhou Zone 3 ap-guangzhou-3	100003
	Guangzhou Zone 4 ap-guangzhou-4	100004
South China (Shenzhen Finance) ap-shenzhen-fsi	Shenzhen Finance Zone 1 (only financial institutions and enterprises can apply for activation by submitting a ticket) ap-shenzhen-fsi-1	110001
	Shenzhen Finance Zone 2 (only financial institutions and enterprises can apply for activation by submitting a ticket) ap-shenzhen-fsi-2	110002
	Shenzhen Finance Zone 3 (only financial institutions and enterprises can apply for activation by submitting a ticket) ap-shenzhen-fsi-3	110003
East China (Shanghai) ap-shanghai	Shanghai Zone 1 ap-shanghai-1	200001
	Shanghai Zone 2 ap-shanghai-2	200002
	Shanghai Zone 3 ap-shanghai-3	200003
	Shanghai Zone 4 ap-shanghai-4	200004
	Shanghai Zone 5 ap-shanghai-5	200005

	Shanghai Zone 6 ap-shanghai-6	200006
	Shanghai Zone 7 ap-shanghai-7	200007
East China (Shanghai Finance) ap-shanghai-fsi	Shanghai Finance Zone 1 (only financial institutions and enterprises can apply for activation by submitting a ticket) ap-shanghai-fsi-1	700001
	Shanghai Finance Zone 2 (only financial institutions and enterprises can apply for activation by submitting a ticket) ap-shanghai-fsi-2	700002
	Shanghai Finance Zone 3 (only financial institutions and enterprises can apply for activation by submitting a ticket) ap-shanghai-fsi-3	700003
East China (Nanjing) ap-nanjing	Nanjing Zone 1 ap-nanjing-1	330001
	Nanjing Zone 2 ap-nanjing-2	330002
North China (Beijing) ap-beijing	Beijing Zone 1 ap-beijing-1	800001
	Beijing Zone 2 ap-beijing-2	800002
	Beijing Zone 3 ap-beijing-3	800003
	Beijing Zone 4 ap-beijing-4	800004
	Beijing Zone 5 ap-beijing-5	800005
North China (Beijing Finance) ap-beijing-fsi	Beijing Finance Zone 1 (only financial institutions and enterprises can apply for activation by submitting a ticket) ap-beijing-fsi-1	460001
North China (Tianjin) ap-tianjin	Tianjin Zone 1 ap-tianjin-1	360001
	Tianjin Zone 2 ap-tianjin-2	360002
Southwest China (Chengdu) ap-chengdu	Chengdu Zone 1 ap-chengdu-1	160001
	Chengdu Zone 2 ap-chengdu-2	160002
Southwest China (Chongqing) ap-chongqing	Chongqing Zone 1 ap-chongqing-1	190001
Hong Kong/Macao/Taiwan (Hong Kong, China) ap-hongkong	Hong Kong (China) Zone 1 (Hong Kong (China) nodes can cover the China regions of Hong Kong, Macao, and Taiwan) ap-hongkong-1	300001
	Hong Kong (China) Zone 2 (Hong Kong (China) nodes can cover the China regions of Hong Kong, Macao, and Taiwan) ap-hongkong-2	300002
Hong Kong/Macao/Taiwan (Taipei, China)	Taipei (China) Zone 1 ap-taipei-1	390001

ap-taipei		

Other countries/regions

Region	Availability Zone	ZoneId
Southeast Asia Pacific (Singapore) ap-singapore	Singapore Zone 1 (Singapore nodes can cover Southeast Asia Pacific) ap-singapore-1	900001
Southeast Asia Pacific (Bangkok) ap-bangkok	Bangkok Zone 1 (Bangkok nodes can cover Southeast Asia Pacific) ap-bangkok-1	230001
South Asia Pacific (Mumbai) ap-mumbai	Mumbai Zone 1 (Mumbai nodes can cover South Asia Pacific) ap-mumbai-1	210001
	Mumbai Zone 2 (Mumbai nodes can cover South Asia Pacific) ap-mumbai-2	210002
Northeast Asia Pacific (Seoul) ap-seoul	Seoul Zone 1 (Seoul nodes can cover Northeast Asia Pacific) ap-seoul-1	180001
Northeast Asia Pacific (Tokyo) ap-tokyo	Tokyo Zone 1 (Tokyo nodes can cover Northeast Asia Pacific) ap-tokyo-1	250001
West US (Silicon Valley) na-siliconvalley	Silicon Valley Zone 1 (Silicon Valley nodes can cover West US) na-siliconvalley-1	150001
	Silicon Valley Zone 2 (Silicon Valley nodes can cover West US) na-siliconvalley-2	150002
East US (Virginia) na-ashburn	Virginia Zone 1 (Virginia nodes can cover East US) na-ashburn-1	220001
	Virginia Zone 2 (Virginia nodes can cover East US) na-ashburn-2	220002
North America (Toronto) na-toronto	Toronto Zone 1 (Toronto nodes can cover North America) na-toronto-1	400001
Europe (Frankfurt) eu-frankfurt	Frankfurt Zone 1 (Frankfurt nodes can cover Europe) eu-frankfurt-1	170001
Europe (Moscow) eu-moscow	Moscow Zone 1 (Moscow nodes can cover Europe) eu-moscow-1	240001

Performance

Last updated : 2020-09-08 15:43:27

Specifications

Memory edition

Feature	Standard Architecture		Cluster Architecture
Compatible Redis version	2.8	4.0, 5.0	4.0, 5.0
Memory	256 MB-60 GB	1-60 GB	12 GB-4 TB
Number of shards	Unsupported	Unsupported	3-128
QPS	80,000-100,000	80,000-100,000	80,000-100,000 per shard
Max connections	10,000	10,000	10,000 per shard
Traffic limit	10-64 MB/s	10-64 MB/s	144 MB/s-6 GB/s
Multi-database	Supported	Supported	Supported
Mget, Mset	Supported	Supported	Supported
Lua	Supported	Supported	Supported (cross-slot access not supported)
Horizontal scaling	Unsupported	Unsupported	Supported
Replica scaling	Unsupported	Supported	Supported
Read/Write separation	Unsupported	Supported	Supported
GEO	Unsupported	Supported	Supported
Number of replicas	0-1	1-5	1-5

The 256 MB specification is a limited introductory version. It is only suitable for functionality verification in test environments but not recommended for production environments. Other specifications currently cannot be scaled down to the 256 MB specification.

CKV edition

Feature	Standard Architecture	Cluster Architecture
Compatible Redis version	3.2	3.2
Memory	4-384 GB	12 GB-48 TB
Number of shards	-	3-128
QPS	80,000–120,000	Tens of millions
Max connections	12,000-24,000	12,000–24,000 per shard
Traffic limit	16-256 MB/s	72 MB/s-32 GB/s
Multi-database	Supported	Supported
Mget, Mset	Supported	Supported

Feature	Standard Architecture	Cluster Architecture
Lua	Supported	Limited support (to use Lua in the cluster edition, you need to make sure that the keys accessed in the Lua script are in the same slot, and the `key` field must be included in the command parameters)
Horizontal scaling	Unsupported	Supported
Replica scaling	Unsupported	Unsupported
Read/write separation	Unsupported	Unsupported
GEO	Supported	Supported

Traffic and Connections

Memory edition

Specification (GB)	Max Connections	Max Throughput (MB/s)
0.25	3,000	10
1	10,000	16
2	10,000	24
4	10,000	24
8	10,000	24
12	10,000	32
16	10,000	32
20	10,000	48
24	10,000	48
32	10,000	48
40	10,000	64
48	10,000	64
60	10,000	64

CKV edition

Specification (GB)	Max Connections	Max Throughput (MB/s)
4	10,000	24
8	10,000	24
16	10,000	32
24	10,000	32
32	10,000	32
48	18,000	64
64	18,000	64
80	18,000	64
96	18,000	64
128	24,000	128

160	24,000	128
192	24,000	128
256	24,000	256
320	24,000	256
384	24,000	256

Cluster edition connections = number of connections per shard * number of shards;
Cluster edition throughput = shard throughput * number of shards

> ⚠ **Note :**
> After scaling, legacy instances capable of up to 9,000 connections will be capable of 10,000 connections.

Performance Data

Performance references

The time needed to execute Redis commands varies. Businesses use different database commands in their production environments; therefore, the corresponding performance values will also vary. The test results listed here are obtained with specified parameters and are for your reference only. Please conduct tests in your actual business environment for more accurate results.

- Single-node test performance

Redis Instance Specification	Connections	QPS
Memory edition (standard architecture), 8 GB	10,000	80,000-100,000
Memory edition (cluster architecture), 8 GB (per shard)	10,000	80,000-100,000
CKV edition (standard architecture), 8 GB	12,000	80,000-120,000

- Cluster architecture test performance
  Memory edition (cluster architecture) performance = memory edition (standard architecture) performance * number of shards
  CKV edition (cluster architecture) performance = CKV edition (standard architecture) performance * number of shards

Test method

- Test environment

Number of CVMs in the Pressure Test Client	CVM Cores	CVM MEM	Region	Redis Instance Specification
3	2	8 GB	Guangzhou Zone 2	Memory edition (standard architecture), 8 GB
3	2	8 GB	Guangzhou Zone 2	CKV edition (standard architecture), 8 GB

- Test parameters

```
redis-benchmark -h 10.66.187.x -p 6379 -a crs-1znib6aw:chen2016 -t set -c 3500 -d 128 -n 25000000 -r 5000000
redis-benchmark -h 10.66.187.x -p 6379 -a crs-1z5536aw:chen2016 -t set -c 3500 -d 128 -n 25000000 -r 5000000
redis-benchmark -h 10.66.187.x -p 6379 -a crs-090rjlih:1234567 -t set -c 3500 -d 128 -n 25000000 -r 5000000
```

- QPS calculation
  Sum of the QPS values of 3 pressure test clients (tested by redis-benchmark).

Storage Engine

Last updated : 2020-05-26 10:33:56

Memory Edition Engine

The Memory Edition engine provides a native Redis experience and supports a myriad of scenarios. It supports both Redis standard and cluster deployment architectures to meet the requirements of different business scenarios.

Editions supported by the Memory Edition engine include:

- Memory Edition (standard architecture): when the number of replicas is greater than 0, data is synced between the master node and the replica nodes (slaves) in real time. When the master node fails, automatic failover will be performed in a matter of seconds, and a replica node will take over the business in an imperceptible manner. The master/slave architecture guarantees high availability of system services and provides 0.25–60 GB of storage capacity.
- Memory Edition (cluster architecture): a cluster instance uses a distributed architecture, which allows flexible selection of shard quantity, shard capacity, and replica quantity and enables scaling imperceptible to the business. It provides 6 GB–4 TB of storage capacity and a performance of tens of millions of QPS.

Hybrid Storage Edition Engine

Tendis, the Hybrid Storage Edition engine, is Tencent's proprietary RocksDB storage engine compatible with the Redis protocol. It features high performance, compression ratio, and stability and has been widely used in Tencent's various businesses.

Editions supported by the Hybrid Storage Edition engine include:

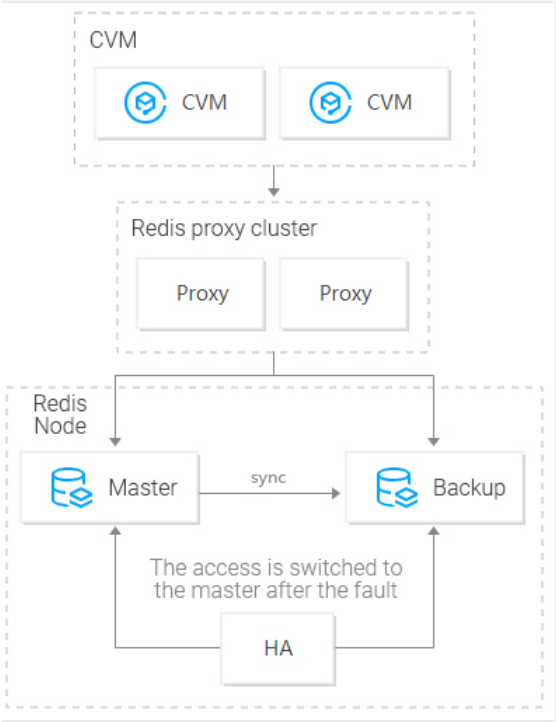
- Hybrid Storage Edition (cluster architecture): it is compatible with the version commands of Redis Cluster Edition v4.0 and provides 240 GB–32 TB of storage capacity. It supports 6 replicas of disk data to fully guarantee data reliability.

Product Series
Memory Edition (Standard Architecture)

Last updated : 2020-06-12 14:45:18

TencentDB for Redis Memory Edition (standard architecture) refers to the edition that supports zero or more replicas (a replica refers to a node that is not a master node), which is the most common Redis edition. It is compatible with protocols and commands of Redis v2.8, v4.0, and v5.0 and features data persistence and backup, making it suitable for scenarios where high data reliability and availability are required. A master node provides daily service access, while a slave node ensures high availability (HA). In case that the master node fails, the system will automatically switch to the slave node to guarantee business continuity.

Memory Edition (standard architecture) - one replica:



Replica Description

Memory Edition (standard architecture) supports 0–5 replicas to meet the different requirements for availability and performance of your business in different scenarios. All replicas of Memory Edition (standard architecture) play a role in supporting system's high availability, so the more replicas, the higher the availability. If the number of replicas is greater than 1, read/write separation can be enabled to extend the read performance through replica nodes.

Definition:

- Master node: a Redis node that provides read and write capabilities.
- Replica node: a Redis node that provides high availability or read-only capability. A master node cannot be a replica node.

Replica support:

Instance Edition	Supported Replica Quantity	Read/Write Separation
2.8 Memory Edition (standard architecture)	0–1	Not supported
4.0 Memory Edition (standard architecture)	1–5	Supported
5.0 Memory Edition (standard architecture)	1–5	Supported

Zero-Replica instance:

- Data may be lost when a node fails; therefore, do not use a zero-replica instance in data storage scenarios.
- A zero-replica instance is only suitable for pure cache scenarios. The application system can continue to run even in case of Redis failure, but as a zero-replica instance has only one database node, when the node fails, the system will start a new Redis process (which has no data), and after node failover is completed, the application needs to warm up the data again to avoid access pressure on the backend database.

Read-only replica (read/write separation):

- Supported editions: 4.0 Memory Edition (standard architecture) and above instance. When the number of replicas is greater than 1, automatic read/write separation can be enabled to extend the read performance vertically. Up to 5 replica nodes can be supported.
- How it works: after read-only replica is enabled, write requests will be routed to the master node, while read requests will be routed to all replica nodes through the load balancing algorithm and no longer be processed by the master node. This read/write separation feature is provided by the Proxy component built in TencentDB for Redis.
- Enabling/Disabling: you can enable or disable read-only replica on the instance creation page in the TencentDB for Redis Console or through TencentCloud API.

Features

- **Service reliability (1–5 replicas)**
  With a dual-server master/slave architecture, the master and slave nodes reside on different physical machines with the master node providing external access. You can perform data CRUD using the Redis command line or client. In case that the master node fails, the proprietary high availability (HA) system will automatically perform master-slave switch to ensure smooth operation of the business.
- **Data reliability (1–5 replicas)**
  The data persistence feature is enabled by default. Memory Edition (standard architecture) supports data backup. You can roll back or clone instances for backup sets to effectively cope with data misoperations and other issues.

Use Limits

- Memory Edition (standard architecture) supports 0.25–60 GB of storage capacity. For higher specifications, please use Cluster Edition that supports up to 4 TB of capacity.
- Memory Edition (standard architecture) supports up to 100,000 QPS (set command concurrencies). If you need a higher QPS, you can choose multi-replica read/write separation or use Redis Cluster Edition that supports tens of millions of QPS.
- As a zero-replica instance cannot ensure data reliability, and the business data needs to be warmed up after a node failure, if your business requires high data availability, you are not recommended to use zero-replica instances; instead, you can choose single-replica or multi-replica instances.

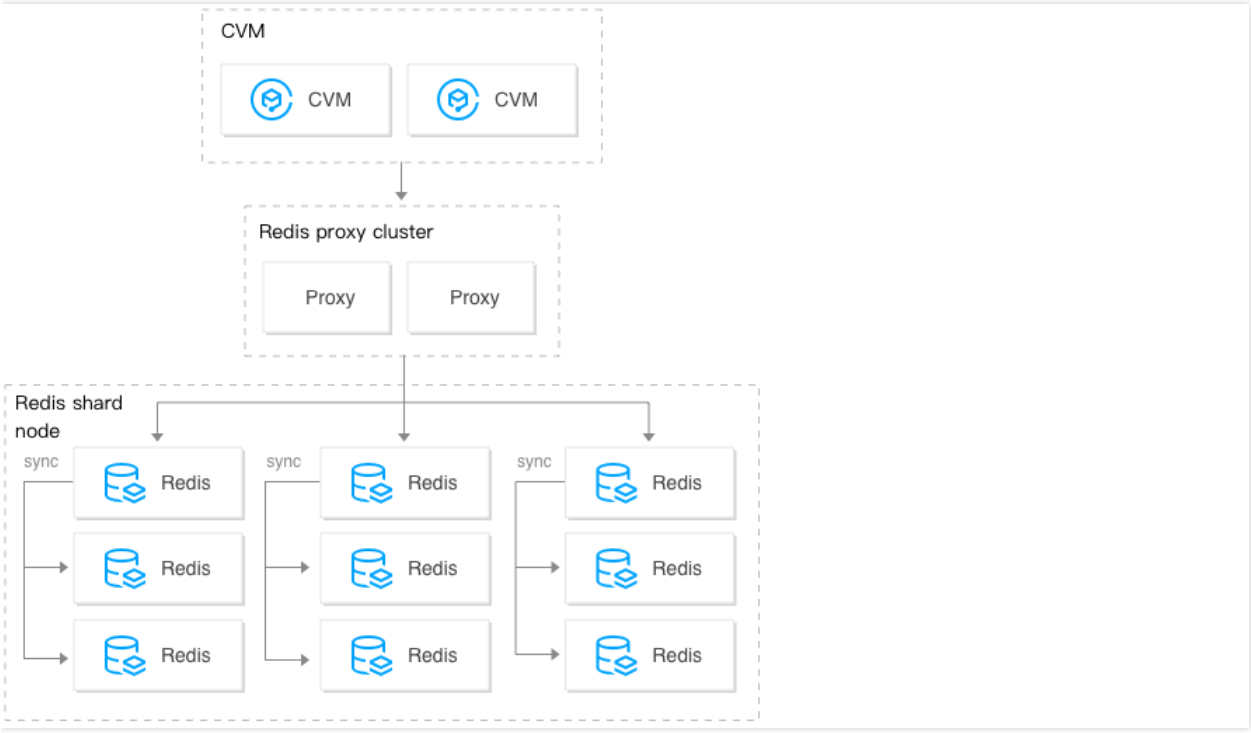
Command Compatibility Description

For more information on the supported commands, please see Command Compatibility.

Memory Edition (Cluster Architecture)

Last updated : 2020-09-08 10:16:40

TencentDB for Redis Memory Edition (cluster architecture) is a new edition of Redis built by Tencent Cloud based on Community Edition of Redis Cluster that is compatible with Redis 4.0 and 5.0 commands. It uses a distributed architecture to enable elastic scaling and features high flexibility, availability, and performance of tens of millions of QPS. Specifically, it supports horizontal scaling of 3–128 shards and replica scaling of 1–5 replica sets, where the scaling and migration are virtually imperceptible to the business, maximizing the system availability.



Cluster Specification

- Shard size (GB): 2, 4, 8, 12, 16, 20, 24, 28, 32
- Number of shards: 3, 5, 8, 12, 16, 24, 32, 64, 96, 128
- Number of replicas: 1, 2, 3, 4, 5

> **Note :**
> With the shard specification of 2 GB, up to 8 shards can be supported.

Cluster Mode

- In cluster mode, data is automatically sharded. The system provides data load balancing and migration capabilities.
- The cluster mode supports shards of 2–32 GB specifications.
- The cluster mode is compatible with certain commands of the non-cluster mode, mainly reflected in cross-slot data access. For more information, pleas see Command Compatibility Description.

Replica Description

- When there is only one replica, Redis provides master/slave real-time hot backup for high data reliability and high availability (cross-server in the same AZ). When the HA system detects a node failure, it requests for switching to a slave node, and add a new slave node to the system.

- When the number of replicas is greater than 1, Redis provides master/slave real-time hot backup with the slave nodes being read-only.

Features

Flexibility
Memory Edition (cluster architecture) supports horizontal scaling of 3–128 nodes and replica scaling of 1–5 replica sets, making it ideal for various scenarios through instance specification adjustment.
Availability
In Memory Edition (cluster architecture), scaling of shard quantity and replica quantity are virtually imperceptible to the business, maximizing the system availability.
Compatibility
Memory Edition (cluster architecture) supports use cases of native clusters of Redis Community Edition and Codis and is compatible with clients such as Jedis.
OPS
Memory Edition (cluster architecture) maximizes system capability openness and has advanced features such as shard-level monitoring and management, data migration and load balancing, as well as monitoring of big and hot keys, which help facilitate total system management and OPS.

Use Cases

Master/Slave high-availability scenarios
This edition allows you to configure a replica set for a single node to achieve high master/slave availability. It features dual-server hot backup and automatic failover to ensure high reliability and availability of the Redis service.
Read/Write separation scenarios
When the number of replica nodes is greater than 1, automatic read/write separation can be enabled for the TencentDB for Redis instance to extend the read performance of a single node. Up to 5 replica sets can be supported and read access weights across the master node and replica nodes can be configured.
Multi-Shard high-performance scenarios
Memory Edition (cluster architecture) automatically enables auto-sharding and achieves horizontal scaling of system performance by assigning different keys to multiple nodes.

Command Compatibility Description

Memory Edition (cluster architecture) stores data in a distributed manner, and its biggest difference from the standard architecture lies in whether a single command supports multikey access. For the cluster architecture, commands can be categorized into supported, partially supported, and unsupported. For the complete list of compatible commands, please see Command Compatibility.

Unsupported commands

The system will return the following error:

```
keys *
(error) ERR unknown command 'keys'
```

Partially supported commands

Memory Edition (cluster architecture) is compatible with smart clients such as Jedis Cluster. For compatibility with Jedis Cluster, TencentDB for Redis modifies the IP list returned by the supported commands, and the IP address of each node in the returned information is the instance's VIP.

- CLUSTER NODES
- CLUSTER SLOTS
- CONFIG GET

Supported cross-slot commands

Currently, cross-slot access commands supported by Memory Edition (cluster architecture) include MGET, MSET, and DEL but not other multikey commands.

Custom commands

Through VIP encapsulation, Memory Edition (cluster architecture) provides a user experience in cluster mode comparable to the standalone edition, making it much easier for use in different scenarios. To increase the transparency to OPS, custom commands can be used. Access to each node in the cluster is supported by adding a parameter **node ID** on the right of the original command parameter list, such as COMMAND arg1 arg2 ... node ID. The node ID can be obtained through the `cluster nodes` command or in the console:

```
10.1.1.1:2000> cluster nodes
25b21f1836026bd49c52b2d10e09fbf8c6aa1fdc 10.0.0.15:6379@11896 slave 36034e645951464098f40d339386e9d51a9d7e77 0 1531471918205 1 connected
da6041781b5d7fe21404811d430cdffea2bf84de 10.0.0.15:6379@11170 master - 0 1531471916000 2 connected 10923-16383
36034e645951464098f40d339386e9d51a9d7e77 10.0.0.15:6379@11541 myself,master - 0 1531471915000 1 connected 0-5460
53f552fd8e43112ae68b10dada69d3af77c33649 10.0.0.15:6379@11681 slave da6041781b5d7fe21404811d430cdffea2bf84de 0 1531471917204 3 connected
18090a0e57cf359f9f8c8c516aa62a811c0f0f0a 10.0.0.15:6379@11428 slave ef3cf5e20e1a7cf5f9cc259ed488c82c4aa17171 0 1531471917000 2 connected
ef3cf5e20e1a7cf5f9cc259ed488c82c4aa17171 10.0.0.15:6379@11324 master - 0 1531471916204 0 connected 5461-10922

Native command:
info server
Custom command:
info server ef3cf5e20e1a7cf5f9cc259ed488c82c4aa17171

SCAN command examples:
scan 0 238b45926a528c85f40ae89d6779c802eaa394a2
scan 0 match a* 238b45926a528c85f40ae89d6779c802eaa394a2

KEYS command example:
keys a* 238b45926a528c85f40ae89d6779c802eaa394a2
```

Custom command list:

- INFO
- MEMORY
- SLOWLOG
- FLUSHDB
- PING
- KEYS (hashtag is supported and has matching priority)
- SCAN (hashtag is supported and has matching priority)
- MONITOR

Transactional support

Memory Edition (cluster architecture) supports transactional commands provided that the transactions are started by the WATCH command. The keys of a transaction should be stored in the same slot, and the keys of WATCH and transaction-related keys should also be stored in the same slot. HashTag is recommended for multikey transactions in cluster mode.

Multi-Database support

Memory Edition (cluster architecture) supports multiple databases (256 by default); therefore, it can support all commands related to database operations.

Hybrid Storage Edition (Cluster Architecture)

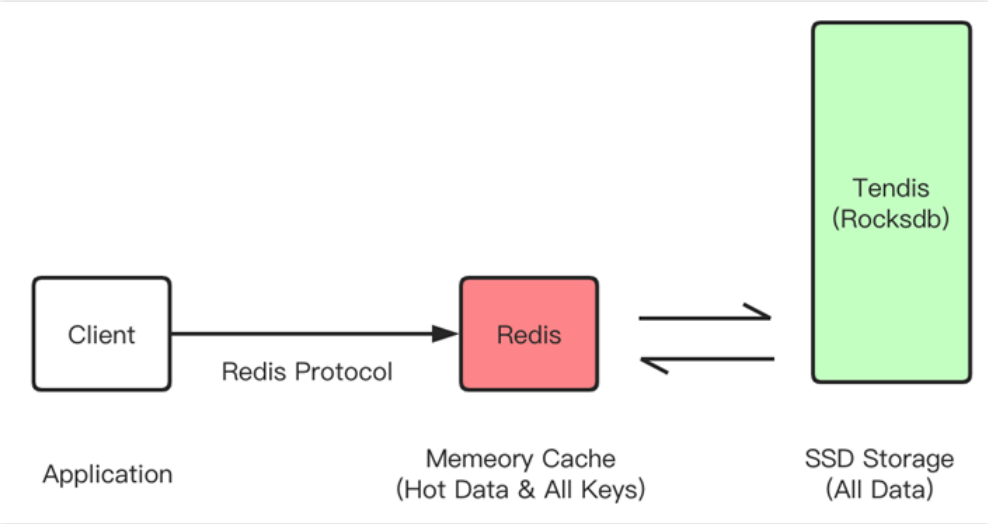
Last updated : 2020-09-17 11:36:53

TencentDB for Redis Hybrid Storage Edition (cluster architecture) is based on Tendis, a KV (key-value) RocksDB storage engine developed by and widely used in Tencent. It is compatible with Redis protocols and features high performance, high compression ratio and high stability. Tencent has extensive experience in Tendis operation.

Hybrid Storage Edition (cluster architecture) is composed of two components, i.e., Redis (cache) and Tendis (storage engine). It is suitable for KV storage scenarios, as it balances performance and cost, and greatly reduces your business operating costs by 80% in the scenarios where cold data takes up a lot of storage space.

Hybrid Storage Edition (cluster architecture) is fully compatible with Redis 4.0 Cluster Edition commands. It is easy to use and can make full use of a rich variety of data structures and operational commands of Redis for efficiency.

Hybrid Storage Edition (cluster architecture) stores all data on disk, and caches all keys and the values of hot keys in the memory. The system architecture is shown as below:



Features

Low cost

- Data is automatically cached and automatically degraded to cold data. All data is stored on disk, and hot data is cached in the memory. Hybrid Storage Edition reduces operating costs by 40% to 80% compared with TencentDB for Redis Memory Edition.
- Hybrid Storage Edition adopts the LZ4 data compression algorithm to automatically compress data once stored on disk, which balances performance and capacity and saves up to 90% of the disk capacity.

High efficiency

- With 100% compatibility with Redis protocols, all efficient Redis data structures and APIs can be used in the business.
- In Hybrid Storage Edition, the business does not need to swap hot and cold data, or deal with the data inconsistency, cache breakdown, cache avalanche and other problems existing in traditional caching schemes. Hybrid Storage Edition reduces the complexity of the business, improves the development efficiency and reduces the OPS cost.

High performance

- Up to 3,000,000+ QPS for hot data access, comparable to native Redis
- Up to 1,000,000 QPS for concurrent writes

Large capacity

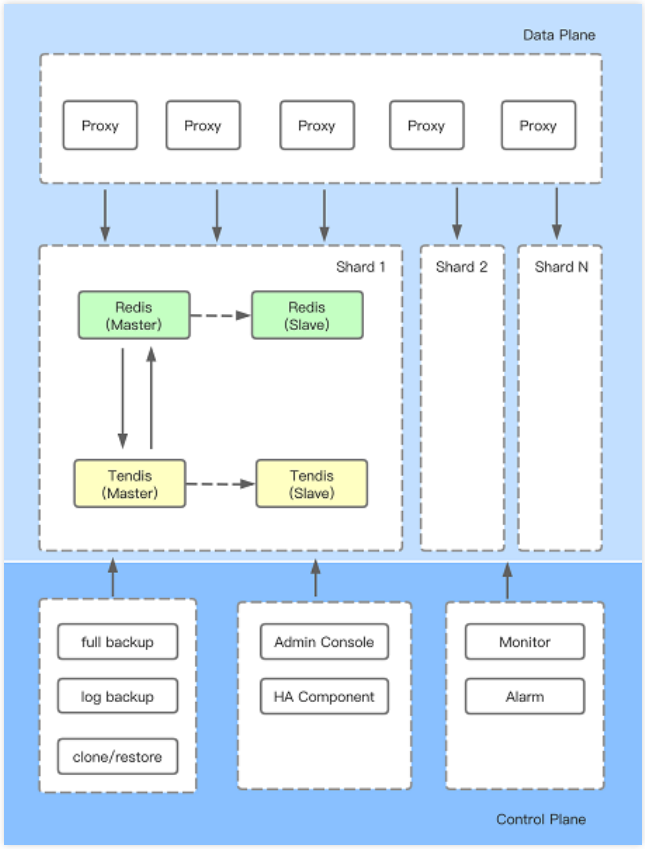
- Super-large storage capacity of 240 GB to 32 TB

- The data stored on disk can have 6 replicas, fully ensuring data reliability.

Architecture

The core components of TencentDB for Redis Hybrid Storage Edition (cluster architecture) include Proxy, Redis cache, and Tendis engine, as described below:

- Proxy: it routes and distributes client requests, distributes commands to the correct shard according to their keys, collects part of monitoring data, and disables high-risk commands online, etc.
- Redis cache: it is based on Redis 4.0 Cluster Edition. In order to achieve automatic degradation to cold data, Hybrid Storage Edition modifies core Redis features, including value eviction, value eviction based on time, synchronization of written data to Tendis, cold data access, and master/slave synchronization of hot data, etc. The modified Redis in Hybrid Storage Edition is 100% compatible with Redis 4.0 Cluster Edition commands.
- Tendis engine: it is a KV storage engine developed by Tencent and compatible with Redis protocols. Tendis has been used in Tencent for many years with its performance and stability being fully verified. In the hybrid storage system, its key features include the storage and reading of full data, data backup, incremental log backup, etc.



Application Scenarios

Recommended scenarios

Storage scenarios

The data access acceleration scenario where Redis is used as the primary storage engine rather than the cache.

Latency-insensitive scenario

Because cold data needs to be read from disks, the latency of accessing cold data in Hybrid Storage Edition will significantly increase to tens or even hundreds of milliseconds, while the access delay in Redis Memory Edition is less than 1 ms.

Distributed hot/cold data scenario

Hybrid storage is ideal for scenarios where hot data degrades to cold data over time. Hot data is accessed frequently and requires short response latency. Cold data can have a short response latency after prefetch.

Scenarios not recommended

Caching scenario

- Insufficient performance: the caching scenario requires high performance, but the Hybrid Storage Edition's performance in cold data is much inferior to that of Memory Edition. For more information about performance, please see Performance.
- Cost increase: Hybrid Storage Edition increases the disk storage space and adds more disk storage compute nodes. Since the caching scenario requires memory capacity close to disk capacity, business costs will rise rather than fall.

Latency-sensitive scenario

Because cold data needs to be read from disks, the latency of accessing cold data in Hybrid Storage Edition will significantly increase to tens or even hundreds of milliseconds, while the access delay in Redis Memory Edition is less than 1 ms. Therefore, Hybrid Storage Edition is not recommended for a latency-sensitive scenario.

Specifications

Note :

- The minimum disk capacity must be greater than the memory capacity, otherwise the data may not be written.
- The memory caches all keys and only evicts values, so the memory may not be able to cache all keys due to too small disk capacity configuration. Please evaluate the disk space.
- Run the following `set` command with the 128-byte value to test the maximum write performance:

```
redis-benchmark -h 10.0.0.5 -p 6379 -c 100 -n 60000000 -r 1000000000 -d 128 -t set -a passwd
```

Shard Quantity	Total Cache Capacity (GB)	Total Disk Capacity Range (GB)	Maximum Write Performance (QPS)
4	64	240 - 520	60,000
4	128	480 - 960	60,000
4	256	1,000 - 2,000	60,000
8	128	480 - 960	120,000
8	256	960 - 1,920	120,000
8	512	2,000 - 4,000	120,000
16	256	960 - 1,920	240,000
16	512	1,920 - 3,840	240,000
16	1,024	4,000 - 8,000	240,000
32	512	3,840 - 7,680	480,000
32	1,024	7,680 - 15,360	480,000
32	2,048	16,000 - 32,000	480,000

Degradation to Cold Data

Value eviction policy

- **Value eviction only**
  When you set a timeout period for keys or use the `expire` command, both keys and values will be evicted from the memory; otherwise, only values will be evicted.
- **value-eviction-policy**
  - Through `value-eviction-policy`, you can set how many days a value has not been accessed before it is automatically evicted from the memory.
- The default value of this parameter is 7 days. It can be modified by users in the console.
- **maxmemory-policy**
  - Hybrid Storage Edition only supports `allkeys-lru` (default) and `allkeys-random`.
  - When Redis memory usage reaches `maxmemory`, the system evicts values from the memory according to `maxmemory-policy`.

Value cache policy

- **value-cache-policy**
  You can use this parameter to configure when disk data is cached into Redis memory: Redis caches a value into the memory if the number of times the value is accessed within 5 minutes is greater than `value-cache-policy`. This parameter can avoid cache invalidation caused by, such as, data traversal. If this parameter is configured to `1`, the cold data is immediately cached.

- **The `expire` command description**
  If you use `Expire Time` to set a timeout period for keys, Hybrid Storage Edition will follow the original semantics of this command to evict expired keys and values from the memory and disks. The same is true for keys set with `EXPIRE`, `EXPIREAT`, `PEXPIRE`, and `PEXPIREAT` commands.

- **Big key eviction**
  To ensure reading performance, Hybrid Storage Edition currently does not evict a value from the memory, if the value is larger than 8 MB or has complex (non-string) structures with more than 1,000 fields. Therefore, Hybrid Storage Edition does not have an ideal effect on the degradation of complex data structures, such as large Hash structures, which will be continuously optimized in the future.

Command Compatibility

Hybrid Storage Edition (cluster architecture) stores data in a distributed manner, and its biggest difference from the Memory Edition (standard architecture) lies in whether a single command supports multikey access. For the cluster architecture, commands can be categorized into supported, custom, and unsupported. For the complete list of compatible commands, please see Command Compatibility.

- **Unsupported commands**
  The system will return the following error:

```
keys *
(error) ERR unknown command 'keys'
```

- **Partially supported commands**
  Hybrid Storage Edition (cluster architecture) is compatible with smart clients such as JedisCluster. For compatibility with JedisCluster, TencentDB for Redis modifies the IP list returned by the supported commands, and the IP address of each node in the returned information is the instance's VIP.
  - CLUSTER NODES
  - CLUSTER SLOTS
  - CONFIG GET

- **Supported cross-slot commands**
  Currently, cross-slot access commands supported by Hybrid Storage Edition (cluster architecture) include MGET, MSET, and DEL but not other multikey commands.

- **Custom commands**
  Through VIP encapsulation, Hybrid Storage Edition (cluster architecture) provides a user experience in cluster mode comparable to the

standalone edition, making it much easier for use in different scenarios. To increase the transparency to OPS, custom commands can be used. Access to each node in the cluster is supported by adding a parameter **node ID** on the right of the original command parameter list, such as `COMMAND arg1 arg2 ... [node ID]` . The node ID can be obtained through the `cluster nodes` command or in the console.

```
10.1.1.1:2000> cluster nodes
25b21f1836026bd49c52b2d10e09fbf8c6aa1fdc 10.0.0.15:6379@11896 slave 36034e645951464098f40d339386e9d51a9d7e77 0 1531471918205 1 connected
da6041781b5d7fe21404811d430cdffea2bf84de 10.0.0.15:6379@11170 master - 0 1531471916000 2 connected 10923-16383
36034e645951464098f40d339386e9d51a9d7e77 10.0.0.15:6379@11541 myself,master - 0 1531471915000 1 connected 0-5460
53f552fd8e43112ae68b10dada69d3af77c33649 10.0.0.15:6379@11681 slave da6041781b5d7fe21404811d430cdffea2bf84de 0 1531471917204 3 connected
18090a0e57cf359f9f8c8c516aa62a811c0f0f0a 10.0.0.15:6379@11428 slave ef3cf5e20e1a7cf5f9cc259ed488c82c4aa17171 0 1531471917000 2 connected
ef3cf5e20e1a7cf5f9cc259ed488c82c4aa17171 10.0.0.15:6379@11324 master - 0 1531471916204 0 connected 5461-10922

Native command:
info server
Custom command:
info server ef3cf5e20e1a7cf5f9cc259ed488c82c4aa17171

SCAN command examples:
scan 0 238b45926a528c85f40ae89d6779c802eaa394a2
scan 0 match a* 238b45926a528c85f40ae89d6779c802eaa394a2

KEYS command example:
keys a* 238b45926a528c85f40ae89d6779c802eaa394a2
```

- **Transactional support**
  Hybrid Storage Edition (cluster architecture) supports transactional commands. The keys of a transaction should be stored in the same slot, and the keys of the `WATCH` command and transaction-related keys should be stored in the same slot too. HashTag is recommended for multikey transactions in cluster mode.

- **Multi-database support**
  Hybrid Storage Edition (cluster architecture) supports the `SELECT 0` command but not multiple databases.

- **Poor-performance commands**
  - `linsert` and `lrem` : the `linsert` and `lrem` commands in the List command family have poor performance and are not recommended thus. They will traverse the list nodes in the disk with the O(n) execution time complexity. If there are many list nodes, the command execution will time out.
  - `append` : the `append` command performs poorly when the character size exceeds 1 MB.

Read/Write Separation

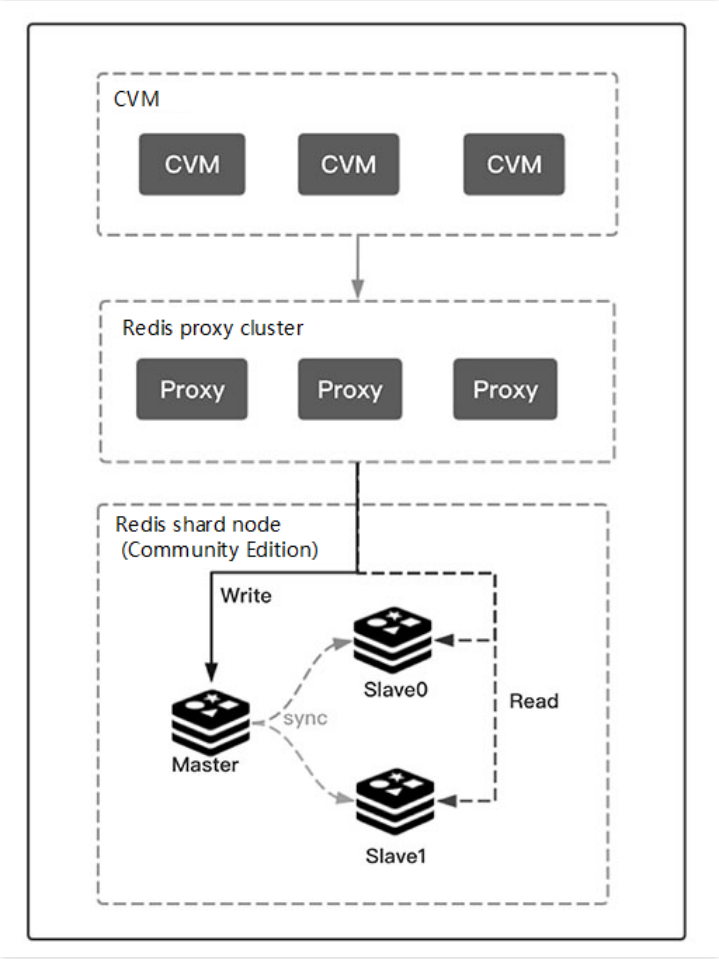
Last updated : 2020-07-07 10:47:38

TencentDB for Redis supports read/write separation for business scenarios with more reads but less writes, which can well cope with read requests concentrating on frequently read data. It supports up to 1-master 5-slave mode to offer 5x read performance.

How It Works

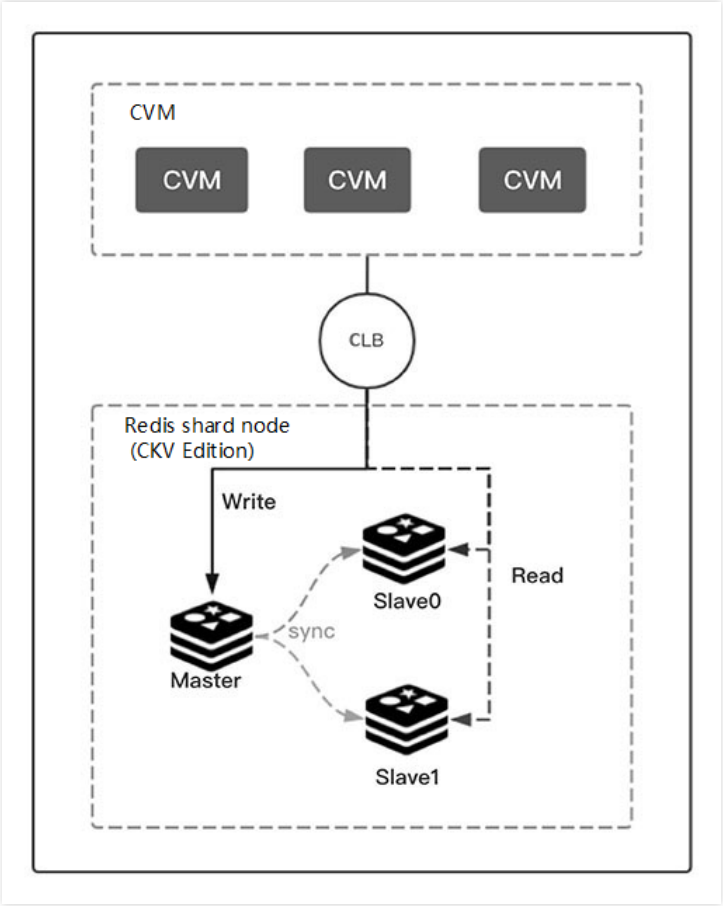
Memory edition

- **Read/write separation principle**: Redis v4.0 and above in standard and cluster architectures implement automatic read/write separation at the Proxy layer.
- **Read/write separation weight**: after read/write separation is enabled, Proxy will enable access by directing write requests to the master node only and distributing read requests evenly on slave nodes.



CKV edition

- **Read/write separation principle**: CKV Edition inherently supports the read/write separation architecture. All requests are distributed to nodes in clusters through the CLB gateway, and each node has the global slot routing information. After read/write separation is enabled for a node, if the read key hits it, the data will be read directly and returned; otherwise, the request will be forwarded to the corresponding node according to the routing information, which will read the data and return it to this node for final return to the client.
- **Read/write separation weight**: requests in the CKV Edition are distributed by CLB, so read and write weights are evenly distributed according to the quadruple of the TCP connection (source IP, source port, destination IP, and destination port).



Command Compatibility

Last updated : 2020-09-17 10:18:40

Command Compatibility of Different Editions

In the following table, ✓ indicates "supported", x indicates "unsupported", and - indicates that cross-slot access is not applicable to the command:

- For custom command descriptions, please see Custom Command.
- For more information on the command compatibility of the Memory Edition (Cluster Architecture), please see Memory Edition (Cluster Architecture) > Command Compatibility.
- For more information on the command compatibility of the Hybrid Storage Edition (Cluster Architecture), please see Hybrid Storage Edition (Cluster Architecture) > Command Compatibility.
- Command table download address.

Command Group	Command	2.8 Memory Edition (Standard Architecture)	4.0 Memory Edition (Standard Architecture)	4.0 Memory/Hybrid Storage Edition (Cluster Architecture)	5.0 Memory Edition (Standard Architecture)	5.0 Memory Edition (Cluster Architecture)	Cross-slot Support in Memory Edition (Cluster Architecture)
connection group	auth	✓	✓	✓	✓	✓	-
	echo	✓	✓	✓	✓	✓	-
	ping	✓	✓	Custom	✓	Custom	-
	quit	✓	✓	✓	✓	✓	-
	select	✓	✓	✓	✓	✓	-
	swapdb	x	✓	✓	✓	✓	-
hash group	hdel	✓	✓	✓	✓	✓	-
	hexists	✓	✓	✓	✓	✓	-
	hget	✓	✓	✓	✓	✓	-
	hgetall	✓	✓	✓	✓	✓	-
	hincrby	✓	✓	✓	✓	✓	-
	hincrbyfloat	✓	✓	✓	✓	✓	-
	hkeys	✓	✓	✓	✓	✓	-
	hlen	✓	✓	✓	✓	✓	-
	hmget	✓	✓	✓	✓	✓	-
	hmset	✓	✓	✓	✓	✓	-
	hset	✓	✓	✓	✓	✓	-
	hsetnx	✓	✓	✓	✓	✓	-
	hstrlen	✓	✓	✓	✓	✓	-
	hvals	✓	✓	✓	✓	✓	-
	hscan	✓	✓	✓	✓	✓	-
keys group	del	✓	✓	✓	✓	✓	✓
	scan	✓	✓	Custom	✓	Custom	-

	exists	✓	✓	✓	✓	✓	x
	expire	✓	✓	✓	✓	✓	-
	expireat	✓	✓	✓	✓	✓	-
	keys	✓	✓	Custom	✓	Custom	-
	type	✓	✓	✓	✓	✓	-
	move	✓	✓	✓	✓	✓	-
	ttl	✓	✓	✓	✓	✓	-
	persist	✓	✓	✓	✓	✓	-
	pexpire	✓	✓	✓	✓	✓	-
	pexpireat	✓	✓	✓	✓	✓	-
	pttl	✓	✓	✓	✓	✓	-
	randomkey	✓	✓	✓	✓	✓	-
	rename	✓	✓	✓	✓	✓	x
	renamenx	✓	✓	✓	✓	✓	x
	sort	✓	✓	✓	✓	✓	-
list group	lindex	✓	✓	✓	✓	✓	-
	linsert	✓	✓	✓	✓	✓	-
	llen	✓	✓	✓	✓	✓	-
	lpop	✓	✓	✓	✓	✓	-
	lpush	✓	✓	✓	✓	✓	-
	lpushx	✓	✓	✓	✓	✓	-
	lrange	✓	✓	✓	✓	✓	-
	lrem	✓	✓	✓	✓	✓	-
	lset	✓	✓	✓	✓	✓	-
	ltrim	✓	✓	✓	✓	✓	-
	rpop	✓	✓	✓	✓	✓	-
	rpoplpush	✓	✓	✓	✓	✓	x
	rpush	✓	✓	✓	✓	✓	-
	rpushx	✓	✓	✓	✓	✓	-
	blpop	✓	✓	✓	✓	✓	x
	brpop	✓	✓	✓	✓	✓	x
	brpoplpush	✓	✓	✓	✓	✓	x
pub/sub group	psubscribe	✓	✓	✓	✓	✓	-
	pubsub	✓	✓	✓	✓	✓	-
	publish	✓	✓	✓	✓	✓	-
	punsubscribe	✓	✓	✓	✓	✓	-

	subscribe	✓	✓	✓	✓	✓	-
	unsubscribe	✓	✓	✓	✓	✓	-
sets group	sadd	✓	✓	✓	✓	✓	-
	scard	✓	✓	✓	✓	✓	-
	sdiff	✓	✓	✓	✓	✓	x
	sdiffstore	✓	✓	✓	✓	✓	x
	sinter	✓	✓	✓	✓	✓	x
	sinterstore	✓	✓	✓	✓	✓	x
	sismember	✓	✓	✓	✓	✓	-
	smembers	✓	✓	✓	✓	✓	-
	smove	✓	✓	✓	✓	✓	x
	spop	✓	✓	✓	✓	✓	-
	srandmember	✓	✓	✓	✓	✓	-
	srem	✓	✓	✓	✓	✓	-
	sscan	✓	✓	✓	✓	✓	-
	sunion	✓	✓	✓	✓	✓	x
	sunionstore	✓	✓	✓	✓	✓	x
sorted sets group	zadd	✓	✓	✓	✓	✓	-
	zcard	✓	✓	✓	✓	✓	-
	zcount	✓	✓	✓	✓	✓	-
	zincrby	✓	✓	✓	✓	✓	-
	zinterstore	✓	✓	✓	✓	✓	x
	zlexcount	✓	✓	✓	✓	✓	-
	zrange	✓	✓	✓	✓	✓	-
	zrangebylex	✓	✓	✓	✓	✓	-
	zrangebyscore	✓	✓	✓	✓	✓	-
	zrank	✓	✓	✓	✓	✓	-
	zrem	✓	✓	✓	✓	✓	-
	zremrangebylex	✓	✓	✓	✓	✓	-
	zremrangebyrank	✓	✓	✓	✓	✓	-
	zremrangebyscore	✓	✓	✓	✓	✓	-
	zrevrange	✓	✓	✓	✓	✓	-
	zrevrangebylex	✓	✓	✓	✓	✓	-
	zrevrangebyscore	✓	✓	✓	✓	✓	-
	zscore	✓	✓	✓	✓	✓	-
	zrevrank	✓	✓	✓	✓	✓	-

	zscan	✓	✓	✓	✓	✓	-
	zunionstore	✓	✓	✓	✓	✓	x
	zpopmax	x	x	x	✓	✓	-
	zpopmin	x	x	x	✓	✓	-
	bzpopmax	x	x	x	✓	✓	-
	bzpopmin	x	x	x	✓	✓	-
strings group	append	✓	✓	✓	✓	✓	-
	bitcount	✓	✓	✓	✓	✓	-
	bitop	✓	✓	✓	✓	✓	x
	bitpos	✓	✓	✓	✓	✓	-
	decr	✓	✓	✓	✓	✓	-
	decrby	✓	✓	✓	✓	✓	-
	get	✓	✓	✓	✓	✓	-
	getbit	✓	✓	✓	✓	✓	-
	getrange	✓	✓	✓	✓	✓	-
	getset	✓	✓	✓	✓	✓	-
	incr	✓	✓	✓	✓	✓	-
	incrby	✓	✓	✓	✓	✓	-
	incrbyfloat	✓	✓	✓	✓	✓	-
	mget	✓	✓	✓	✓	✓	✓
	mset	✓	✓	✓	✓	✓	✓
	msetnx	✓	✓	✓	✓	✓	x
	psetex	✓	✓	✓	✓	✓	-
	setex	✓	✓	✓	✓	✓	-
	set	✓	✓	✓	✓	✓	-
	setbit	✓	✓	✓	✓	✓	-
	setnx	✓	✓	✓	✓	✓	-
	setrange	✓	✓	✓	✓	✓	-
	strlen	✓	✓	✓	✓	✓	-
	bitfield	x	✓	✓	✓	✓	-
transactions group	discard	✓	✓	✓	✓	✓	-
	exec	✓	✓	✓	✓	✓	-
	multi	✓	✓	✓	✓	✓	-
	unwatch	✓	✓	✓	✓	✓	-
	watch	✓	✓	✓	✓	✓	-
hyperloglog	pfadd	✓	✓	✓	✓	✓	-

group	pfcount	✓	✓	✓	✓	✓	x
	pfmerge	✓	✓	✓	✓	✓	x
scripting group	eval	✓	✓	✓	✓	✓	x
	evalsha	✓	✓	✓	✓	✓	x
	script debug	✓	✓	✓	✓	✓	-
	script exists	✓	✓	✓	✓	✓	x
	script flush	✓	✓	✓	✓	✓	-
	script load	✓	✓	✓	✓	✓	-
	script kill	✓	✓	✓	✓	✓	-
geo group	geoadd	x	✓	✓	✓	✓	-
	geohash	x	✓	✓	✓	✓	-
	geopos	x	✓	✓	✓	✓	-
	geodist	x	✓	✓	✓	✓	-
	georadius	x	✓	✓	✓	✓	-
	georadiusbymember	x	✓	✓	✓	✓	-
keys group	touch	✓	✓	✓	✓	✓	-
	restore	✓	✓	✓	✓	✓	-
	object	x	x	x	x	x	-
	unlink	x	✓	✓	✓	✓	x
	wait	x	x	x	x	x	-
	migrate	x	x	x	x	x	-
	dump	✓	✓	✓	✓	✓	-
server group	bgrewriteaof	x	x	x	x	x	-
	bgsave	x	x	x	x	x	-
	client kill	x	x	x	x	x	-
	sync	x	x	x	x	x	-
	psync	x	x	x	x	x	-
	client list	✓	✓	✓	✓	✓	-
	client getname	x	x	x	x	x	-
	client pause	x	x	x	x	x	-
	client reply	x	x	x	x	x	-
	client setname	x	x	x	x	x	-
	command count	x	x	x	x	x	-
	command getkeys	x	x	x	x	x	-
	command info	x	x	x	x	x	-
	slaveof	x	x	x	x	x	-

	config rewrite	x	x	x	x	x	-
	config set	x	x	x	x	x	-
	config resetstat	x	x	x	x	x	-
	debug object	x	x	x	x	x	-
	debug segfault	x	x	x	x	x	-
	role	x	x	x	x	x	-
	save	x	x	x	x	x	-
	lastsave	x	x	x	x	x	-
	shutdown	x	x	x	x	x	-
	MEMORY	x	✓	Custom	✓	Custom	-
	command	✓	✓	✓	✓	✓	-
	dbsize	✓	✓	✓	✓	✓	-
	info	✓	✓	Custom	✓	Custom	-
	time	✓	✓	✓	✓	✓	-
	client list	✓	✓	✓	✓	✓	-
	config get	✓	✓	✓	✓	✓	-
	monitor	✓	✓	Custom	✓	Custom	-
	flushdb	✓	✓	Custom	✓	Custom	-
	flushall	✓	✓	✓	✓	✓	-
	slowlog	✓	✓	Custom	✓	Custom	-
	cluster keyslot	x	x	✓	x	✓	-
	cluster nodes	x	x	✓	x	✓	-
	cluster getkeysinslot	x	x	✓	x	✓	-
	cluster (others)	x	x	x	x	x	-
	module	x	x	x	x	x	-
	lolwut	x	x	x	✓	✓	-
Stream group	xinfo	x	x	x	✓	✓	-
	xadd	x	x	x	✓	✓	-
	xtrim	x	x	x	✓	✓	-
	xdel	x	x	x	✓	✓	-
	xrange	x	x	x	✓	✓	-
	xrevrange	x	x	x	✓	✓	-
	xlen	x	x	x	✓	✓	-
	xread	x	x	x	✓	✓	x
	xgroup	x	x	x	✓	✓	-
	xreadgroup	x	x	x	✓	✓	x

	xack	x	x	x	✓	✓	-
	xlclaim	x	x	x	✓	✓	-
	xpending	x	x	x	✓	✓	-

Relevant Concepts

Last updated : 2019-10-16 13:31:55

TencentDB for Redis generally involves the following concepts:

Instance: A database environment running independently in Tencent Cloud. One database instance can contain multiple user-created databases.

VPC: A custom virtual network space that is logically isolated from other resources.

Security group: Security access control to Redis instances by specifying IP, protocol, and port rules for instance access.

Region and availability zone: The physical location of a Redis instance and other resources.

Tencent Cloud Console: Web-based UIs.

Project: A feature developed to enable developers to better manage Tencent Cloud products based on the concept of projects. You can implement project management by assigning different Tencent Cloud products to different projects.

Read-write separation: TencentDB for Redis supports switching Read-Write separation on or off. It targets at business scenarios with more Reads and less Writes, which can well cope with Read requests concentrating on hotspot data. It supports up to 1-master 5-slave mode to offer 5x Read performance scalablity.

Relevant Products

Last updated : 2019-09-09 11:39:54

TencentDB for Redis generally involves the following products:

You can deploy your computing services by purchasing Cloud Virtual Machine (CVM) instances. For more information, see CVM.

You can use Cloud Monitor to monitor the running status of your TencentDB for Redis instances. For more information, see Cloud Monitor.

You can write code to call TencentCloud APIs to access Tencent Cloud products and services. For more information, see the TencentCloud API documentation.