Alibaba Cloud ApsaraDB for Redis

Product Introduction

Issue: 20190627

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. **You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.**
2. **No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.**
3. **The content of this document may be changed due to product version upgrades , adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.**
4. **This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults " and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity , applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use**

or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified , reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates . The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 Warning: Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click OK.
`Courier font`	It is used for commands.	Run the `cd / d C :/ windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*

Style	Description	Example
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich {stand | slave}`

Contents

1 What is ApsaraDB for Redis

Welcome to use ApsaraDB for Redis. This topic describes the architecture and features of ApsaraDB for Redis, and helps you know more about this service.

ApsaraDB for Redis is compatible with open-source Redis protocols and provides the database service. The service supports hybrid storage of memory and hard disks. Based on the highly reliable two-node hot standby structure and smoothly scalable cluster architecture, this service is suitable for scenarios that require high read/write performance and flexible configuration changes.

ApsaraDB for Redis supports various types of data, such as strings, lists, sets, sorted sets, and hash tables. The service also supports advanced features, such as transactio ns, message subscription, and message publishing.

Based on the hybrid storage of memory and hard disks, ApsaraDB for Redis can provide high-speed data read/write capability and support data persistence.

You can deploy ApsaraDB for Redis instances in a flexible architecture. The instances are classified into these editions: standard dual-replica edition and dual-replica cluster edition. These instances are suitable for different scenarios.

- Standard dual-replica edition: The system synchronizes data between the master node and replica node in real time. If the master node fails, the system automatically performs the failover operation and restores services within a few seconds. The replica node takes over services. This automatic process does not affect your business. The master-replica architecture ensures high availability of system services.
- Dual-replica cluster edition: Cluster instances run in a distributed architecture. Each node uses a master-replica high-availability structure to automatically perform failover and disaster recovery. Multiple types of cluster instances are applicable to various businesses. You can scale the database system to improve performance as needed.

As a cloud computing service, ApsaraDB for Redis works with hardware and data deployed in the cloud, and provides comprehensive infrastructure planning, network security protections, and system maintenance services. This service allows you to focus on business innovation.

References

For more information about common types and parameters of all ApsaraDB for Redis editions, such as the maximum number of connections, CPU processing capacity, and performance reference values, see *Types and performance*.

2 Product series

2.1 Overview

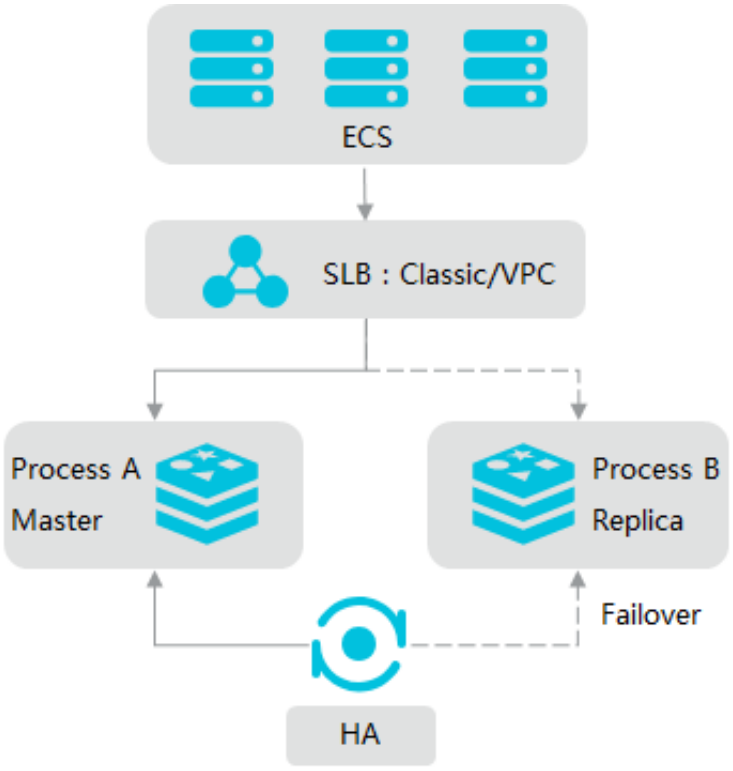
ApsaraDB for Redis provides standard dual-replica edition and dual-replica cluster edition.

Type	Description	Scenario
Standard dual-replica edition	A standard dual-replica instance runs in a master-replica structure .	· Compatibility with Redis protocols. · Persistent data storage based on ApsaraDB for Redis. · Limited QPS performance. · Redis commands are simple and contain few sorting and compute commands.
Dual-replica cluster edition	A dual-replica cluster instance is deployed in a cluster architectu re. Each shard server runs in the dual-replica mode.	· Large data volumes. · High-QPS applications. · Throughput-intensive applicatio ns.

2.2 Standard dual-replica edition

Overview

A standard dual-replica instance runs in a master-replica structure. The master node provides services for your business, and the replica node works to ensure high availability (HA). When the master node fails, the system switches services to the replica node within 30 seconds to ensure stability of services.



Benefits

- Reliability
  - Reliable services

    In the master-replica architecture, the master node and the replica node run on different physical servers. The master node provides services for your business . You can use Redis command-line interface (CLI) commands and common clients to add, modify, search, and delete data in a database. When the master node fails, the proprietary HA system performs the failover operation to ensure stability of services.
  - Reliable data

    By default, the dual-replica standard instance enables data persistence to write all data to disks. The system supports data replication. You can roll back or clone instances based on replica sets, and effectively solve issues caused by accidental operations. The system also supports zone-disaster recovery.
- Compatibility

  The standard edition is developed on the basis of Redis 2.8 and 100% compatible with Redis commands. You can smoothly migrate an on-premises Redis database

to the standard edition of ApsaraDB for Redis. You can also use Data Transmission Service (DTS) to smoothly migrate incremental Redis data.

- Proprietary features
  - HA system

    ApsaraDB for Redis encapsulates the HA system to detect failures on the master node in real time. In this way, the system can effectively solve issues such as disk I/O errors and CPU failures and perform the failover operation in time to ensure high availability of services.
  - Master-replica replication mechanism

    Alibaba Cloud has customized the master-replica replication mechanism in ApsaraDB for Redis. You can replicate data in the format of incremental logs between the master node and the replica node. When the replication is interrupted, system performance and stability is unchanged and free from issues caused by the master-replica replication mechanism of the original Redis database.

    Some issues caused by the master-replica replication mechanism of the original Redis database are described as follows:

    - When the replication is interrupted, the replica node runs the Partial Resynchronization (PSYNC) command to resynchronize partial data. During this process, the resynchronization fails. The master node has to fully synchronize .rdb files to the replica node.
    - To fully synchronize .rdb files, the master node has to perform full replicatio n first due to the single-thread model. As a result, the master node lags for several milliseconds or seconds.
    - The single-thread process leads to the use of the copy-on-write (CoW or COW) mechanism. The mechanism utilizes memory on the master node. This may run out of memory and cause the application to exit abnormally.
    - The replica files that the master node generates utilize disk I/O and CPU resources.
    - The replication of GB-level files may lead to outgoing traffic bursts on the server and increase the sequential I/O throughput of disks. This affects normal response of services and cause more issues.

Scenarios

- **Compatibility with Redis protocols**

  The standard edition is fully compatible with Redis protocols, so you can smoothly migrate your business.
- **Persistent data storage based on ApsaraDB for Redis**

  The standard edition supports data persistence, replication and recovery to ensure data reliability.
- **Limited QPS performance**

  The original Redis database runs in a single-thread mechanism. In scenarios where QPS is 100,000 or less, we recommend that you use this edition. If you require higher performance, select a cluster edition.
- **Redis commands are simple and contain few sorting and compute commands**

  CPU may be bottlenecked due to Redis single-thread mechanism. If you require plenty of sorting and compute operations, we recommend that you select a cluster edition.

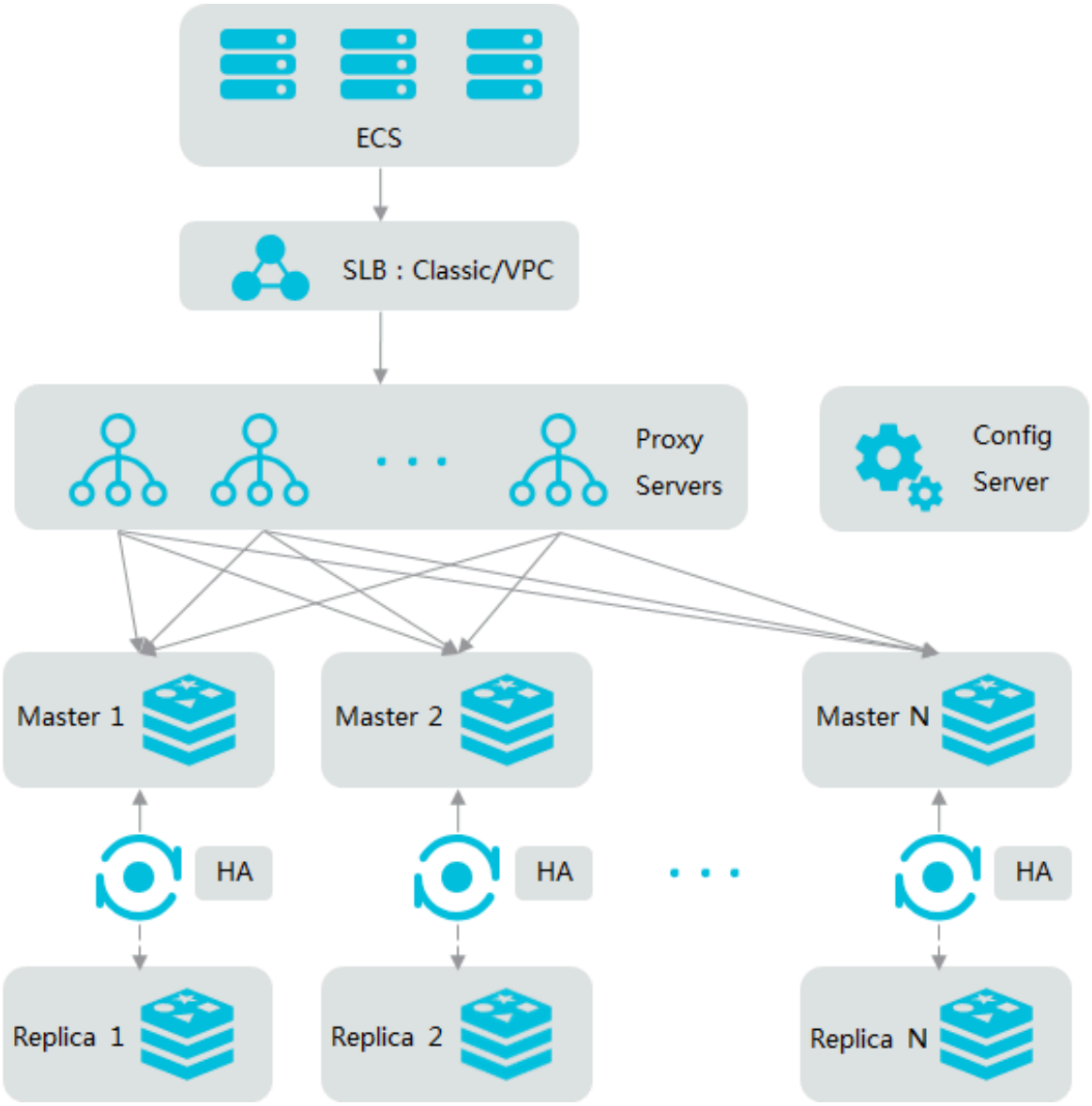
2.3 Dual-replica cluster edition

Overview

ApsaraDB for Redis provides dual-replica cluster instances to eliminate the bottleneck of Redis single-thread model and easily process large-capacity or high-performance services. A cluster instance of ApsaraDB for Redis supports built-in data sharding and reading algorithms. These transparent algorithms relieve you from efforts for research and development (R&D) and operations and maintenance (O&M) when you use the cluster instance of ApsaraDB for Redis.

Components

A dual-replica cluster instance of ApsaraDB for Redis consists of multiple proxy servers, multiple shard servers, and a configuration server.



- **Proxy servers**

  A proxy server runs in a single-node structure. The cluster edition contains multiple proxy servers. The system automatically performs load balancing and failover among these proxy servers.

- **Shard servers**

  Each shard server runs in a dual-replica high-availability structure. If the master node fails, the system automatically performs the failover operation to ensure high availability of services.

- Configuration server

  The configuration server is used to store cluster configurations and partitioning policies. The server runs in a dual-replica high-availability structure to ensure high availability of services.

Cautions

- The system has defined the number and configuration of these components when you purchase the corresponding type of cluster edition. You cannot customize these components. The types of cluster editions are described in *Types and performance*.
- The cluster edition of ApsaraDB for Redis only exposes one domain name. You can access the ApsaraDB for Redis service and perform data operations by using this domain name. The system automatically manages the proxy servers and configuration server, so you do not need to access these servers.
- You can purchase a cluster edition, or upgrade a standard master-replica edition to a cluster edition.

Scenarios

- Large data volumes

  The cluster edition of ApsaraDB for Redis supports data capacity scaling. Compared with the standard edition, the cluster edition supports a larger storage capacity of 64 GB, 128 GB, and 256 GB. You can effectively scale out the data storage structure.

- High-queries per second (QPS) applications

  A standard edition of ApsaraDB for Redis cannot process high QPS. You can deploy multiple nodes to eliminate the performance bottleneck of Redis single-thread model. The cluster edition of ApsaraDB for Redis provides five types of cluster configurations: 16 GB, 32 GB, 64 GB, 128 GB and 256 GB, and supports 8-node and 16-node deployment modes. Therefore, the QPS performance is 8 or 16 times that of the standard edition.

- Throughput-intensive applications

  Compared with the standard edition, the cluster edition provides higher throughput over an internal network. You can easily read hot data and provide high-throughput services by using the cluster edition.

- Applications insensitive to Redis protocols

FAQ

- For more information about memory usage exceptions on child nodes of the cluster edition, see *How do I search for large keys?*.
- For more information about data distribution in memory, see *Redis memory usage analysis*.

3 Types and performance

This topic describes the types and parameters of all ApsaraDB for Redis editions and the method of testing queries per second (QPS). To call the APIs related to a type of instance, use the `InstanceClass` parameter for the corresponding type described in this topic.

 Note:
The maximum internal network bandwidth in below tables includes the maximum upstream bandwidth and the maximum downstream bandwidth. If network resources are sufficient, the bandwidth is unlimited for ApsaraDB for Redis instances. However, if network resources are insufficient, the maximum bandwidth takes effect for the instances.

Standard dual-replica edition

Type	InstanceClass (used in API operations)	Maximum concurrent connections	Maximum internal network bandwidth (MB)	CPU processing capacity	QPS reference value	Description
1 GB master-replica	redis.master.small.default	10,000	10	Single-core	80,000	Master-replica instance
2 GB master-replica	redis.master.mid.default	10,000	16	Single-core	80,000	Master-replica instance
4 GB master-replica	redis.master.stand.default	10,000	24	Single-core	80,000	Master-replica instance
8 GB master-replica	redis.master.large.default	10,000	24	Single-core	80,000	Master-replica instance

Type	InstanceClass (used in API operations)	Maximun concorrei connectic s	Maximun internal network bandwidt (MB)	CPU processin capacity	QPS reference value	Description
16 GB master-replica	redis.master. 2xlarge. default	10,000	32	Single-core	80,000	Master-replica instance
32 GB master-replica	redis.master. 4xlarge. default	10,000	32	Single-core	80,000	Master-replica instance
64 GB master-replica	redis.master. 8xlarge. default	10,000	48	Single-core	80,000	Master-replica instance

Standard zone-disaster recovery edition

Note:
To create a zone-disaster recovery instance, you have to select the region and zone that support zone-disaster recovery, such as China (Hangzhou) Zone (G+H).

Type	InstanceCl ass (used in API operations)	Maximum concorrent connection s	Maximu internal network bandwi (MB)	CPU process capacity	QPS referenc value	Description
Zone-disaster recovery 1 GB	redis.logic .sharding. drredissdb 1g. 1db . 0rodb. 4proxy. default	40,000	10	Single-core	80,000	Master-replica zone-disaster recovery instance

Type	InstanceClass (used in API operations)	Maximum concorrent connections	Maximu internal network bandwi (MB)	CPU process capacity	QPS referenc value	Description
Zone-disaster recovery 2 GB	redis.logic.sharding.drredissdb2g.1db.0rodb.4proxy.default	40,000	16	Single-core	80,000	Master-replica zone-disaster recovery instance
Zone-disaster recovery 4 GB	redis.logic.sharding.drredissdb4g.1db.0rodb.4proxy.default	40,000	24	Single-core	80,000	Master-replica zone-disaster recovery instance
Zone-disaster recovery 8 GB	redis.logic.sharding.drredissdb8g.1db.0rodb.4proxy.default	40,000	24	Single-core	80,000	Master-replica zone-disaster recovery instance
Zone-disaster recovery 16 GB	redis.logic.sharding.drredissdb16g.1db.0rodb.4proxy.default	40,000	32	Single-core	80,000	Master-replica zone-disaster recovery instance

Type	InstanceClass (used in API operations)	Maximum concorrent connections	Maximu internal network bandwi (MB)	CPU process capacity	QPS referenc value	Description
Zone-disaster recovery 32 GB	redis.logic.sharding.drredissdb 32g. 1db. 0rodb. 4proxy. default	40,000	32	Single-core	80,000	Master-replica zone-disaster recovery instance
Zone-disaster recovery 64 GB	redis.logic.sharding.drredissdb 64g. 1db. 0rodb. 4proxy. default	40,000	48	Single-core	80,000	Master-replica zone-disaster recovery instance

Dual-replica cluster edition

Type	InstanceClass (used in API operations)	Numbe of nodes	Maximu concorr connec s	Maximu interna networl bandwi (MB)	CPU process capacit	QPS referen value	Description
16 GB cluster	redis.logic.sharding. 2g. 8db. 0rodb. 8proxy. default	8	80,000	768	8-core	640,000	High-performance cluster instance

Type	InstanceClass (used in API operations)	Number of nodes	Maximum concurrent connections	Maximum internal network bandwidth (MB)	CPU processing capacity	QPS reference value	Description
32 GB cluster	redis.logic.sharding.4g.8db.0rodb.8proxy.default	8	80,000	768	8-core	640,000	High-performance cluster instance
64 GB cluster	redis.logic.sharding.8g.8db.0rodb.8proxy.default	8	80,000	768	8-core	640,000	High-performance cluster instance
128 GB cluster	redis.logic.sharding.8g.16db.0rodb.16proxy.default	16	160,000	1,536	16-core	1,280,000	High-performance cluster instance
256 GB cluster	redis.logic.sharding.16g.16db.0rodb.16proxy.default	16	160,000	1,536	16-core	1,280,000	High-performance cluster instance
512 GB cluster	redis.logic.sharding.16g.32db.0rodb.32proxy.default	32	320,000	3,072	32-core	2,560,000	High-performance cluster instance

> **Note:**
> The number of nodes in the dual-replica cluster instance is the number of master nodes.

Zone-disaster recovery cluster edition

> **Note:**
> To create a zone-disaster recovery instance, you have to select the region and zone that support zone-disaster recovery, such as China (Hangzhou) Zone (G+H).

Type	InstanceClass (used in API operations)	Number of nodes	Maximu concor connec s	Maximu interna networ bandwi (MB)	CPU process capacit	QPS referen value	Description
Zone-disaster recovery 16 GB cluster	redis.logic .sharding. drredismdb 16g. 8db . 0rodb. 8proxy. default	8	80,000	768	8-core	800, 000	Zone-disaster recovery cluster instance
Zone-disaster recovery 32 GB cluster	redis.logic .sharding. drredismdb 32g. 8db . 0rodb. 8proxy. default	8	80,000	768	8-core	800, 000	Zone-disaster recovery cluster instance
Zone-disaster recovery 64 GB cluster	redis.logic .sharding. drredismdb 64g. 8db . 0rodb. 8proxy. default	8	80,000	768	8-core	800, 000	Zone-disaster recovery cluster instance

Type	InstanceClass (used in API operations)	Number of nodes	Maximum concurrent connections	Maximum internal network bandwidth (MB)	CPU processing capacity	QPS reference value	Description
Zone-disaster recovery 128 GB cluster	redis.logic.sharding.drredismdb128g.16db.0rodb.16proxy.default	16	160,000	1,536	16-core	1,600,000	Zone-disaster recovery cluster instance
Zone-disaster recovery 256 GB cluster	redis.logic.sharding.drredismdb256g.16db.0rodb.16proxy.default	16	160,000	1,536	16-core	1,600,000	Zone-disaster recovery cluster instance
Zone-disaster recovery 512 GB cluster	redis.logic.sharding.drredismdb512g.32db.0rodb.32proxy.default	32	320,000	3,072	32-core	3,200,000	Zone-disaster recovery cluster instance

QPS performance reference

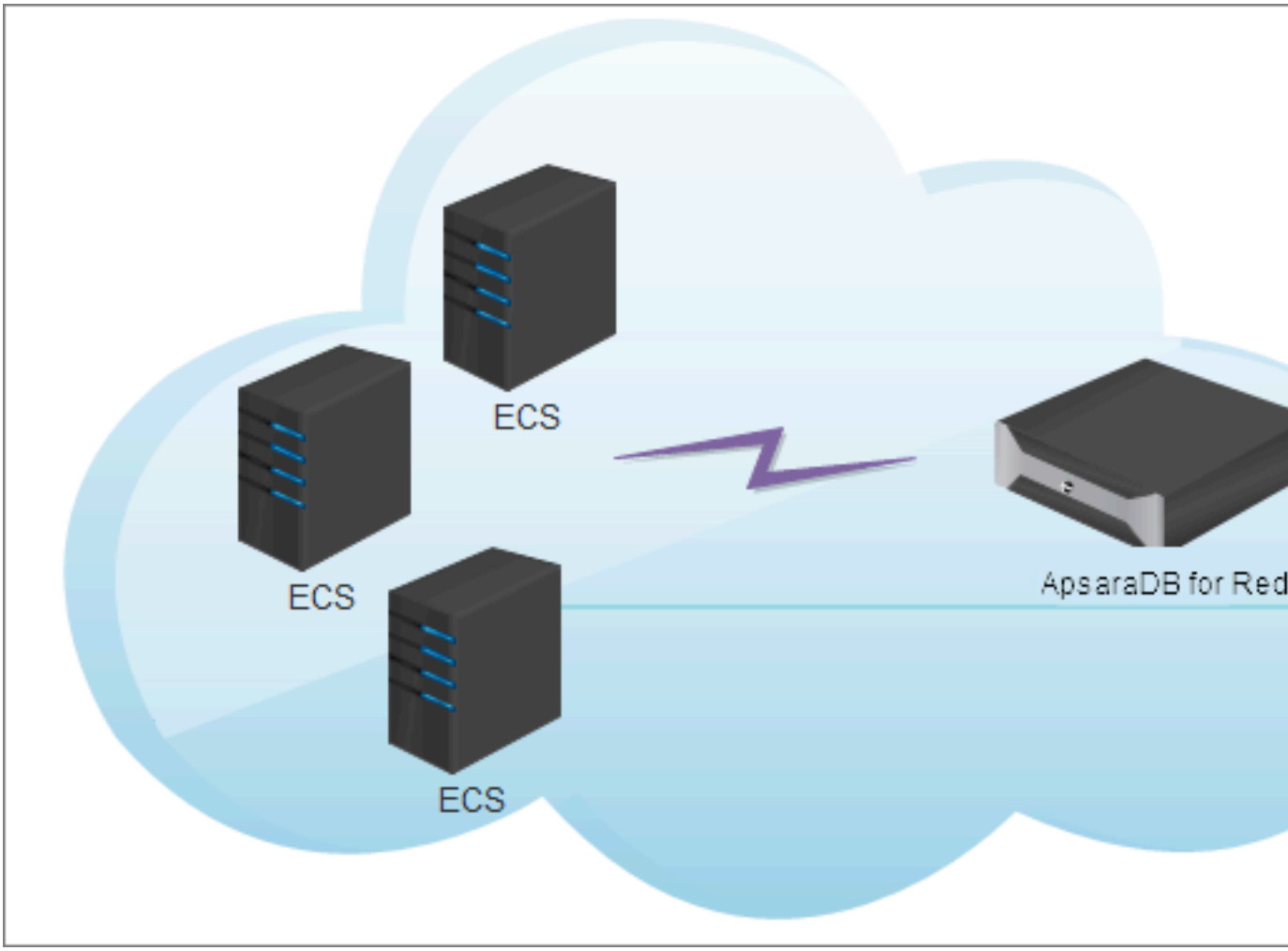
QPS performance

Type	Maximum concorrent connections	Maximum internal network bandwidth (MB)	CPU processing capacity	QPS reference value
8 GB	10,000	24	Single-core	80,000

> **Note:**
>
> QPS reference values of non-cluster instances range from 80,000 to 100,000. QPS reference values of cluster instances are the number of nodes multiplied by the value from 80,000 to 100,000.

QPS testing method

Figure 3-1: Network topology



ECS instance type

Operating system	CPU	Memory	Region	Quantity
Ubuntu 14.04 64-bit	1	2,048 MB	China (Shenzhen)	3

Procedure

1. Download the source code package for redis-2.8.19 to three ECS instances.

```
$ wget http :// download . redis . io / releases / redis - 2 . 8
. 19 . tar . gz
$ tar xzf redis - 2 . 8 . 19 . tar . gz
$ cd redis - 2 . 8 . 19
$ make
$ make install
```

2. Run the following command on these ECS instances at the same time.

```
redis - benchmark - h ***********. m . cnsza . kvstore .
aliyuncs . com - p 6379 - a password - t set - c 50 -
d 128 - n 25000000 - r 5000000
```

3. Summarize the testing data on these ECS instances. The total QPS is the sum of the QPS on each ECS instance.

4 Disaster recovery

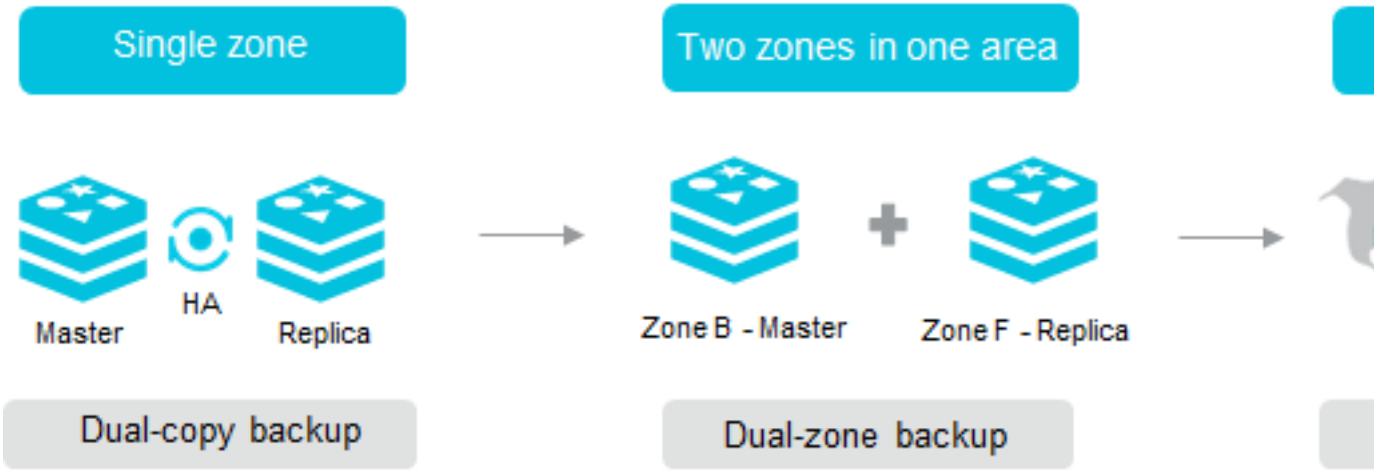
Data is the core element for most businesses. As a carrier of data, a database plays a decisive role in businesses. ApsaraDB for Redis is a high-performance key-value database system. The database stores large amounts of important data for businesses. This topic describes the mechanism of disaster recovery based on ApsaraDB for Redis.

Evolution of the disaster recovery architecture based on ApsaraDB for Redis

Some issues may occur when programs are running, such as software bugs, device faults, and power failures at data centers. An excellent disaster recovery mechanism can ensure data consistency and service availability in these cases. ApsaraDB for Redis improves the disaster recovery capability to ensure high availability (HA) of services, and provides high-availability solutions in various scenarios.

The following figure shows the evolution of the disaster recovery architecture based on ApsaraDB for Redis.

Figure 4-1: Evolution of the disaster recovery architecture based on ApsaraDB for Redis



All of these solutions are available. You can choose them as needed. The following sections describe these solutions in details.

Single-zone high-availability mechanism

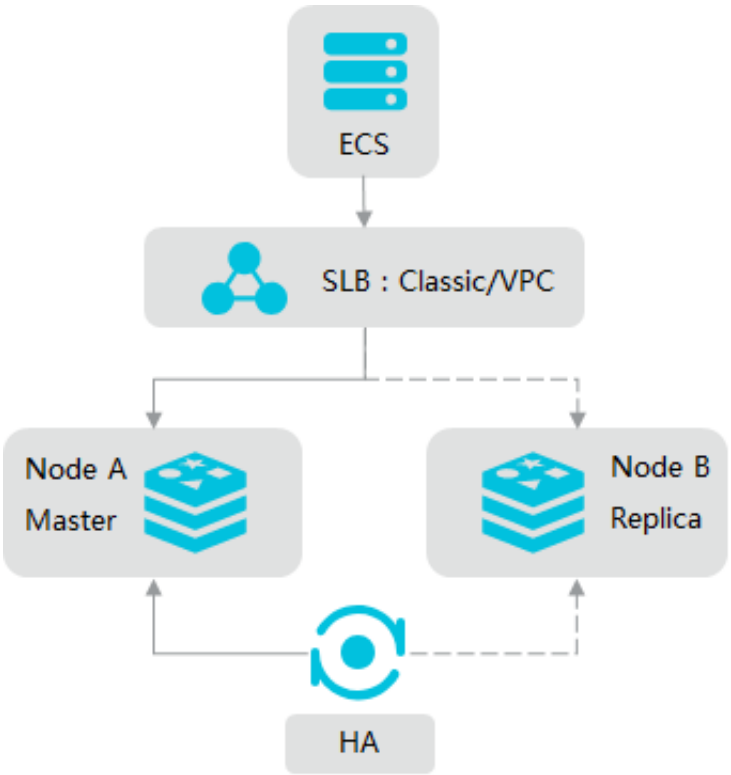
All types of ApsaraDB for Redis instances support the single-zone high-availability architecture. An HA monitoring module uses an independent platform architecture

, and provides a high-availability mechanism across zones. Therefore, ApsaraDB for Redis serves more stably than on-premises Redis systems.

Standard dual-replica edition

A *standard dual-replica* instance uses a master-replica architecture. When detecting a failure on a master node, the HA monitoring module automatically performs the failover operation. The replica node takes over services and becomes a master node, and upon recovery from the failure, the original master node works as a new replica node. The instances support data persistence by default, and allows automatic data replication. You can use the replication files to roll back or clone instances.

Figure 4-2: High-availability architecture of the standard dual-replica edition



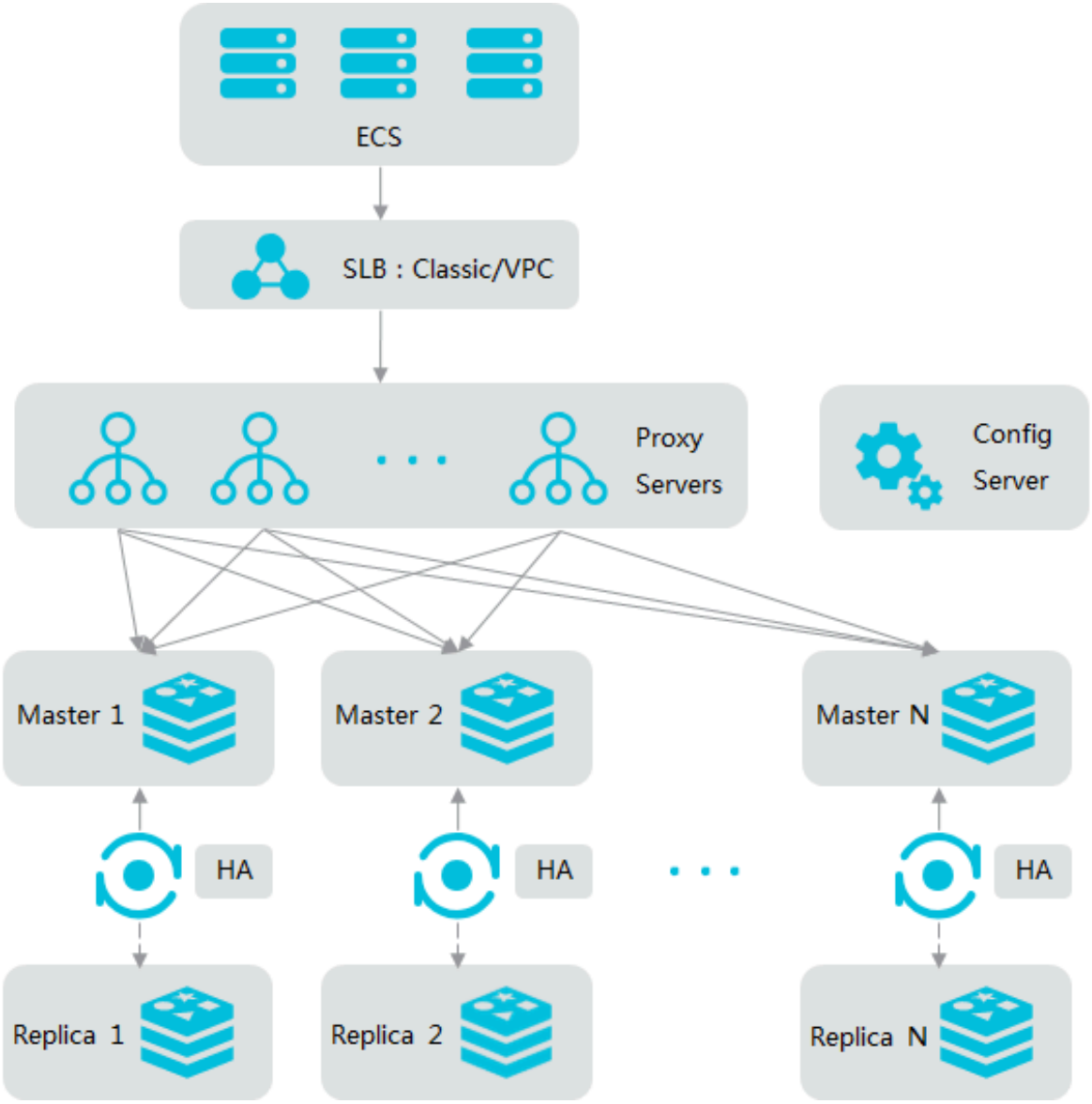
Dual-replica cluster edition

A *dual-replica cluster* instance consists of a configuration server, multiple proxy servers, and multiple shard servers. These servers are described as follows:

- The configuration server is a cluster management tool that provides global routing and configuration information. This server uses a triple-replica cluster architecture that follows the Raft protocol.

- A proxy server uses a single-node architecture. A cluster edition contains multiple proxy servers. ApsaraDB for Redis performs load balancing and failover operations for all proxy servers.
- A shard server uses a dual-replica high-availability architecture. Similar to a standard dual-replica instance, after the master node of the shard server fails, the HA module automatically performs the failover operation to ensure high availability of services, and updates information on the proxy servers and configuration server.

Figure 4-3: High-availability architecture of the dual-replica cluster edition



Zone-disaster recovery mechanism

The standard and cluster editions support zone-disaster recovery between two data centers. You can deploy your business in a single region, and require excellent disaster recovery. In this case, you need to select a zone that supports zone-disaster recovery when you create an ApsaraDB for Redis instance. For example, you can select Singapore Zone (B+C) as shown in the following figure.

Figure 4-4: Create a zone-disaster recovery instance

When you create instances that run in multiple zones, the replica instance at the replica data center is the same type of instance at the master data center. The instances at master and replica data centers synchronize data to each other through a specialized replication channel.

If power or network failures occur at the master data center, the replica instance takes over services and becomes the master instance. The system calls an operation on the configuration server to update routing information for the proxy server. The system performs the failover operation at the network layer according to the routing precision. In normal conditions, the system transmits data directly to the instance at the master data center through precise Classless Inter-Domain Routing (CIDR) blocks . However, the master data center does not upload routing details to the backbone when failures occur. The backbone only provides lower-precision CIDR blocks of the replica data center. The system has to route requests to the replica data center during failover.

ApsaraDB for Redis optimizes Redis synchronization mechanism. Similar to global transaction identifiers (GTIDs) of MySQL, ApsaraDB for Redis uses global operation identifiers (OpIDs) to indicate synchronization offsets and uses background lock-free threads to search OpIDs. The system synchronizes the append-only file (AOF) binary

log (binlog) asynchronously. You can throttle this synchronization to ensure service performance.

Cross-region disaster recovery

ApsaraDB for Redis provides the Redis Global Replica solution, so multiple sites can provide services simultaneously across regions worldwide. This is applicable to synchronizations among multiple regions. Different from traditional disaster recovery solutions, multiple sites work as master nodes at the same time in this solution. Based on this architecture, your business can run in multiple regions simultaneously. Child instances of the global replica instance in these regions synchronize data to each other in real time.

 Note:
Redis Global Replica is now on trial on the Alibaba Cloud China site. Other Alibaba Cloud sites currently do not supported it.

The global replica instance for ApsaraDB for Redis consists of multiple child instances, multiple synchronization channels, and a channel manager.

- A child instance is the basic service unit for a global replica instance. All child instances are readable and writable.
- The synchronization channels support two-way synchronizations between child instances in real time. Based on the resumption from a breakpoint, the system supports a service interruption for a few days.
- The channel manager controls the lifecycle of the synchronization channels, and processes the operations on child instances, such as master-replica failover and replica instance reconstruction, after a master instance fails. In this way, the global replica instance can provide high-availability services.

 Note:
Child instances perform asynchronous replications to synchronize data. This does not affect the service performance of ApsaraDB for Redis.

When you use the global replica instance for ApsaraDB for Redis, you can specify failover conditions on your application. Therefore, when failures occur in a region , your business switches services to the child instance in another region to ensure service availability.

ApsaraDB for Redis provides multiple high-availability architectures to perform instance-level, zone-level, and region-level disaster recovery. You can select the corresponding disaster recovery solution as needed.

5 Features

Flexible architecture

Dual-node hot standby architecture

The system synchronizes data between the master node and replica node in real time. If the master node fails, the system automatically performs the failover operation and restores services within a few seconds. The replica node takes over services. This automatic process does not affect your business. The master-replica architecture ensures high availability of system services.

Cluster architecture

Cluster instances run in a distributed architecture. Each node uses a master-replica high-availability structure to automatically perform failover and disaster recovery. Multiple types of cluster instances are applicable to various businesses. You can scale the database system to improve performance as needed.

Data security

Persistent data storage

ApsaraDB for Redis uses hybrid storage that consists of memory and hard disks. In this way, the system can perform high-speed read and write operations, and support data persistence.

Replication and easy recovery

The system automatically replicates data every day and provides powerful disaster recovery. You can easily restore data in case of accidental data operations to minimize your business losses.

Multi-layer network security protection

- A Virtual Private Cloud (VPC) isolates network transmission at the transport layer.
- The Anti-DDoS protection service monitors and protects against Distributed-Denial-of-Service (DDoS) attacks.
- The system has more than 1,000 IP whitelists configured to control access risks from sources.
- The system requires password verification to provide secure and reliable connections.

In-depth kernel optimization

The experts of Alibaba Cloud have performed in-depth kernel optimization for the Redis source code to effectively prevent running out of memory, fix security vulnerabilities, and protect your business.

High availability

Master-replica structure

Dual-replica instances of standard and cluster editions contain master-replica structures. They can prevent service interruptions caused by a single point of failure (SPOF).

Automatic failure detection and recovery

The system automatically detects hardware failures. In the case of failures, the system performs the failover operation and restores services within a few seconds.

Resource isolation

Instance-level resource isolation is a better measure to ensure stability of individual services.

Elastic scaling

Data capacity scaling

ApsaraDB for Redis supports multiple types of memory. You can upgrade the memory type according to service requirements. This upgrade neither interrupts the service nor affects your business.

Flexible service patterns

The single-node cache architecture and two-node storage architecture are applicable to various service scenarios. You can easily upgrade or downgrade the standard, cluster, and read/write splitting editions.

Performance scaling

The cluster architecture allows you to elastically scale the storage space and throughput performance of the database system. This can eliminate the performance bottlenecks caused by large amounts of data and high-QPS requirements. Therefore, you can easily handle millions of read and write requests per second.

Intelligent O&M

Monitoring platform

The platform provides real-time monitoring and alarms of instance information, such as CPU usage, connections, and disk utilization. You can check instance status anywhere and at any time.

Visual management platform

The ApsaraDB for Redis console, a visual management platform, allows you to easily perform frequent and risky operations, such as instance cloning, replication, and data restoration.

Engine version management

The system automatically upgrades engine versions and quickly fixes flaws so that you can easily manage database versions. This upgrade optimizes parameters of ApsaraDB for Redis to make full use of system resources.

6 Scenarios

Game industry applications

ApsaraDB for Redis can be an important part of the business architecture for deploying a game application.

Scenario 1: ApsaraDB for Redis works as a storage database

The architecture for deploying a game application is simple. You can deploy a main program on an ECS instance and all business data on an ApsaraDB for Redis instance . The ApsaraDB for Redis instance works as a persistent storage database. ApsaraDB for Redis supports data persistence, and stores redundant data on master and replica nodes.

Scenario 2: ApsaraDB for Redis works as a cache to accelerate connections to applications

ApsaraDB for Redis can work as a cache to accelerate connections to applications. You can store data in a Relational Database Service (RDS) database that works as a backend database.

Reliability of the ApsaraDB for Redis service is vital to your business. If the ApsaraDB for Redis service is unavailable, the backend database is overloaded when processing connections to your application. ApsaraDB for Redis provides a two-node hot standby architecture to ensure extremely high availability and reliability of services. The master node provides services for your business. If this node fails, the system automatically switches services to the replica node. The complete failover process is transparent.

E-commerce industry applications

In the e-commerce industry, the ApsaraDB for Redis service is widely used in the modules such as commodity display and shopping recommendation.

Scenario 1: rapid online sales promotion systems

During a large-scale rapid online sales promotion, a shopping system is overwhelme d by traffic. A common database cannot properly handle so many read operations. However, ApsaraDB for Redis supports data persistence, and can work as a database system.

Scenario 2: counter-based inventory management systems

In this scenario, you can store inventory data in an RDS database and save count data to corresponding fields in the database. In this way, the ApsaraDB for Redis instance reads count data, and the RDS database stores count data. In this scenario, ApsaraDB for Redis is deployed on a physical server. Based on solid-state drive (SSD) high-performance storage, the system can provide an extremely high data reading capacity .

Live video applications

In live video services, ApsaraDB for Redis works as an important measure to store user data and relationship information.

Dual-node hot standby ensures high availability

ApsaraDB for Redis uses the two-node hot standby method to maximize service availability.

Cluster editions eliminate the performance bottleneck

ApsaraDB for Redis provides cluster instances to eliminate the performance bottleneck that is caused by Redis single-thread mechanism. Cluster instances can effectively handle traffic bursts during live video streaming and support high-performance requirements.

Easy scaling relieves pressure at peak hours

ApsaraDB for Redis allows you to easily perform scaling. The complete upgrade process is transparent. Therefore, you can easily handle traffic bursts at peak hours.

7 Terms

Term	Description
ApsaraDB for Redis	ApsaraDB for Redis is a high-performance key-value storage system that supports caching and storage. The system is developed on the basis of BSD open-source protocols.
Instance identifier (ID)	An instance corresponds to a user space, and serves as the basic unit of the ApsaraDB for Redis service. ApsaraDB for Redis has restrictions on instance configurations, such as connections, bandwidth, and CPU processing capacity. These restrictions vary according to different instance types. You can view the list of instance identifiers that you have purchased in the console.
Master-replica instance	This is an ApsaraDB for Redis instance that contains a master-replica structure. The master-replica instance provides limited capacity and performance.
High-performance cluster instance	This is an ApsaraDB for Redis instance that runs in a scalable cluster architecture. Cluster instances provide better scalability and performance, but they still have limited features.
Connection address	This is the host address for connecting to ApsaraDB for Redis. The connection address is displayed as a domain name. To obtain the connection address, choose Instance Information > Connection Information.
Connection password	This is the password used to connect to ApsaraDB for Redis. The password is in the format of Instance ID:custom password . For example, if you set the password as 1234 when you purchase an instance and the allocated instance ID is xxxx, the connection password is xxxx:1234.
Eviction policy	This is consistent with the Redis eviction policy. For more information, see *http://redis.io/topics/lru-cache*.
DB	This is the abbreviation of the word database to indicate a database in ApsaraDB for Redis. Each ApsaraDB for Redis instance supports 256 databases numbered DB 0 to DB 255. By default, ApsaraDB for Redis writes data to DB 0.

8 Features of engine version 4.0 of ApsaraDB for Redis

Alibaba Cloud has developed engine version 4.0 of ApsaraDB for Redis based on Redis 4.0 and fixed several bugs to provide you with excellent performance. Engine version 4.0 of ApsaraDB for Redis has all benefits of engine version 2.8 of ApsaraDB for Redis, and supports the following features.

Lazyfree

Engine version 4.0 supports the Lazyfree feature. This feature can avoid congestion on Redis-server caused by the `DEL`, `FLUSHDB`, `FLUSHALL` and `RENAME` commands and ensure service stability. This feature is described as follows.

UNLINK

For engine versions earlier than 4.0, when ApsaraDB for Redis runs the `DEL` command, the command returns `OK` only after releasing the memory of the target key. If the key contains large amounts of data, for example, 10 million items of data in a hash table, other connections have to wait a long time. To be compatible with the existing `DEL` syntax, engine version 4.0 uses the `UNLINK` command. The UNLINK command has the same effect and usage as the `DEL` command, but the background thread releases memory when engine version 4.0 runs the UNLINK command.

```
UNLINK   key  [ key  ...]
```

FLUSHDB/FLUSHALL

The `FLUSHDB` and FLUSHALL commands in engine version 4.0 allow you to specify whether to use the Lazyfree feature to clear all memory.

```
FLUSHALL  [ ASYNC ]
FLUSHDB  [ ASYNC ]
```

RENAME

When ApsaraDB for Redis runs the `RENAME OLDKEY NEWKEY` command, if the specified new key already exists, ApsaraDB for Redis deletes the existing new key first. If the key contains large amounts of data, other connections have to wait a long

time. To use the Lazyfree feature to delete the key in ApsaraDB for Redis, apply the following configuration in the console:

```
lazyfree - lazy - server - del   yes / no
```

Note:

This parameter is not available in the console.

Expire or evict data

You can specify data expiration time and allow ApsaraDB for Redis to delete expired data. However, when ApsaraDB for Redis deletes a large expired key, CPU jitter may occur. Engine version 4.0 allows you to specify whether to use the Lazyfree feature to expire or evict data.

```
lazyfree - lazy - eviction   yes / no
lazyfree - lazy - expire   yes / no
```

New commands

SWAPDB

The `SWAPDB` command is used to exchange data between two databases. After ApsaraDB for Redis runs the `SWAPDB` command, you can connect to the target database and check new data directly without running the `SELECT` command.

```
127 . 0 . 0 . 1 : 6379 >  select   0
OK
127 . 0 . 0 . 1 : 6379 >  set   key   value0
OK
127 . 0 . 0 . 1 : 6379 >  select   1
OK
127 . 0 . 0 . 1 : 6379 [ 1 ]>  set   key   value1
OK
127 . 0 . 0 . 1 : 6379 [ 1 ]>  swapdb   0   1
OK
127 . 0 . 0 . 1 : 6379 [ 1 ]>  get   key
" value0 "
127 . 0 . 0 . 1 : 6379 [ 1 ]>  select   0
OK
127 . 0 . 0 . 1 : 6379 >  get   key
" value1 "
```

ZLEXCOUNT

The `ZLEXCOUNT` command is used for sorted sets and similar to the `ZRANGEBYLE X` command. However, the `ZRANGEBYLE X` command returns the target members, and the `ZLEXCOUNT` command returns the number of target members.

MEMORY

Engine versions earlier than 4.0 support the `INFO MEMORY` command to provide limited memory information. Engine version 4.0 allows you to use the `MEMORY` command to obtain comprehensive memory status of ApsaraDB for Redis.

```
127 . 0 . 0 . 1 : 6379 >  memory   help
1 ) " MEMORY   DOCTOR                          -  Outputs   memory
problems   report "
2 ) " MEMORY   USAGE  < key > [ SAMPLES  < count >] -  Estimate
memory   usage   of   key "
3 ) " MEMORY   STATS                           -  Show   memory
usage   details "
4 ) " MEMORY   PURGE                           -  Ask   the
allocator   to   release   memory "
5 ) " MEMORY   MALLOC - STATS                    -  Show   allocator
internal   stats "
```

- `MEMORY USAGE`

  The `USAGE` child command is used to check the memory usage of a specified key in ApsaraDB for Redis.

  > **Notice:**
  >
  > - Key-value pairs in ApsaraDB for Redis use memory. ApsaraDB for Redis also uses memory when managing these key-value pairs.
  > - The memory usage of keys such as hash tables, lists, sets, and sorted sets is calculated based on sampling. The `SAMPLES` command controls the number of samples.

- `MEMORY STATS`

```
27 . 0 . 0 . 1 : 6379 >  memory   stats
       1 ) " peak . allocated "    //  The   maximum   memory
that   ApsaraDB   for   Redis   has   used   since   startup .
       2 ) ( integer )  423995952
       3 ) " total . allocated "    // The   current   memory
usage .
       4 ) ( integer )  11130320
       5 ) " startup . allocated "    // The   memory   that
ApsaraDB   for   Redis   uses   after   startup   and   initializa
tion .
       6 ) ( integer )  9942928
       7 ) " replicatio  n . backlog "    // The   memory   of
the   backlog   used   in   resuming   an   interrupte  d   master
- replica   replicatio  n .  Default   value :  10   MB .
       8 ) ( integer )  1048576
       9 ) " clients . slaves "    //  The   memory   used   in   a
  master - replica   replicatio  n .
      10 ) ( integer )  16858
      11 ) " clients . normal "    // The   memory   used   by
read   and   write   buffers   for   common   clients .
```

```
      12 ) ( integer )  49630
      13 ) " aof . buffer "    // The   sum   of   the   cache
used   for   append - only   file  ( AOF )  persistenc  e   and
the   cache   generated   during   the   AOF   rewrite   operation
.
      14 ) ( integer )  3253
      15 ) " db .  0 "    // The   memory   used   by   metadata
in   each   database .
      16 )  1 ) " overhead . hashtable . main "
          2 ) ( integer )  5808
          3 ) " overhead . hashtable . expires " // The   memory
used   for   managing   data   that   has   TTL   configured .
          4 ) ( integer )  104
      17 ) " overhead . total "    // The   total   memory   usage
  for   the   preceding   items .
      18 ) ( integer )  11063904
      19 ) " keys . count "    // The   total   number   of   keys
  in   the   current   storage .
      20 ) ( integer )  94
      21 ) " keys . bytes - per - key "    // The   average   size
  of   each   key   in   the   current   memory .
      22 ) ( integer )  12631
      23 ) " dataset . bytes "        // The   memory   used   by
user   data  (=  Total   memory  -  Memory   used   by   metadata
of   ApsaraDB   for   Redis ).
      24 ) ( integer )  66416
      25 ) " dataset . percentage "    // 100  *  dataset . bytes
/ ( total . allocated  -  startup . allocated )
      26 ) " 5 . 5934348106  384277 "
      27 ) " peak . percentage "    //  100  *  total . allocated
/  peak_alloc  ated
      28 ) " 2 . 6251003742  218018 "
      29 ) " fragmentat  ion "    // The   memory   fragmentat  ion
  ratio .
      30 ) " 1 . 1039986610  412598 "
```

- `MEMORY DOCTOR`

  This command is used to provide diagnostics and indicate hidden issues.

```
Peak   memory :  peak . allocated / total . allocated  >  1 . 5 .
This   indicates   a   possibly   high   memory   fragmentat  ion
ratio .
  High   fragmentat  ion :  fragmentat  ion  >  1 . 4 .  This
indicates   a   high   memory   fragmentat  ion   ratio .
  Big   slave   buffers :  the   average   memory   for   each
replica   buffer   is   more   than   10   MB .  This   may   be
caused   by   high   traffic   of   write   operations   on   the
master   node .
  Big   client   buffers :  the   average   memory   for   a
common   client   buffer   is   more   than   200   KB .  This
may   be   caused   by   the   improper   use   of   pipelining
or   caused   by   Pub / Sub   clients   that   delay   processing
  messages .
```

- `MEMORY STATS` **and** `MALLOC PURGE`

  Both commands are valid only when you use jemalloc.

Least Frequently Used (LFU) mechanism and hotkeys

Engine version 4.0 supports allkey-lfu and volatile-lfu data eviction policies, and allows you to use the `OBJECT` command to obtain the frequency that a specified key is used.

```
object freq user_key
```

Based on the LFU mechanism, you can use a combination of the `SCAN` and OBJECT FREQ commands to find hotkeys. You can also use the Redis command line interface (redis-cli) program as follows:

```
$./ redis - cli -- hotkeys
# Scanning the entire keyspace to find hot keys as
well as
# average sizes per key type . You can use - i 0 . 1
to sleep 0 . 1 sec
# per 100 SCAN commands ( not usually needed ).
[ 00 . 00 %] Hot key ' counter : 0000000000 02 ' found so
far with counter 87
[ 00 . 00 %] Hot key ' key : 0000000000 01 ' found so far
with counter 254
[ 00 . 00 %] Hot key ' mylist ' found so far with counter
107
[ 00 . 00 %] Hot key ' key : 0000000000 00 ' found so far
with counter 254
[ 45 . 45 %] Hot key ' counter : 0000000000 01 ' found so
far with counter 87
[ 45 . 45 %] Hot key ' key : 0000000000 02 ' found so far
with counter 254
[ 45 . 45 %] Hot key ' myset ' found so far with counter
64
[ 45 . 45 %] Hot key ' counter : 0000000000 00 ' found so
far with counter 93
-------- summary -------
Sampled 22 keys in the keyspace !
hot key found with counter : 254 keyname : key :
0000000000 01
hot key found with counter : 254 keyname : key :
0000000000 00
hot key found with counter : 254 keyname : key :
0000000000 02
hot key found with counter : 107 keyname : mylist
hot key found with counter : 93 keyname : counter :
0000000000 00
hot key found with counter : 87 keyname : counter :
0000000000 02
hot key found with counter : 87 keyname : counter :
0000000000 01
hot key found with counter : 64 keyname : myset
```