Alibaba Cloud ApsaraDB for Redis

Product Introduction

Issue: 20191120

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, consequent**

ial, 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 contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, 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 this document 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.**

Document conventions

Style	Description	Example
	A danger notice 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.
	A warning notice 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 restart an instance.
	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	 **Notice:** If the weight is set to 0, the server no longer receives new requests.
	A note indicates supplemental instructions, best practices, tips, and other content.	 **Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click Settings > Network > Set network type.
Bold	Bold formatting is used for buttons, menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands.	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *`Instance_ID`*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`

Style	Description	Example
{} or {a|b}	**This format is used for a required value, where only one item can be selected.**	`switch {active|stand}`

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 transactions, 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: standalone edition, master-replica edition, standalone cluster edition, master-replica cluster edition, read/write splitting edition and read/write splitting cluster edition. These instances are suitable for different scenarios.

- **Standalone edition: This architecture is suitable for cache-only scenarios. Based on this architecture, you can perform flexible upgrading or downgrading for a standalone cluster to meet high queries per second (QPS) requirements in a cost-efficient manner.**
- **Master-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.**
- **Standalone cluster edition: A single-replica cluster instance is deployed in a cluster architecture. Each shard server runs in the single-replica mode. This instance is applicable to cache-only or high-QPS scenarios.**

- **Master-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.**
- **Read/write splitting edition: Similar to the standalone edition, the read/write splitting edition uses a master-replica architecture to ensure high availability. You can attach one or more read-only replica nodes to the master node to synchronize data and improve read performance on a linear scale. This can effectively solve the performance issues caused by hotkeys. Therefore, this instance is suitable for business scenarios that require high read/write ratios. The read/write splitting edition provides proxy servers to automatically assign read and write requests. This edition can provide one, three, and five read-only replica nodes.**
- **Read/write splitting cluster edition: In a read/write splitting cluster instance, you can attach a read-only replica node to each shard server. In this way, each shard server can automatically process read requests. This instance is suitable for ultra-large-scale business scenarios that require high read/write ratios. Each shard server only provides one read-only replica node.**

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 *Specifications and performance*.

2 Release notes

This topic describes the feature updates of ApsaraDB for Redis and the related topics.

June 2019

Feature	Description	Release date	Suppor region	Topic
Port modification	You can modify the port for the public endpoint of an ApsaraDB for Redis instance.	2019-06-30	All	*#unique_6*
Public endpoint	You can apply for a public endpoint for an ApsaraDB for Redis instance. You can use the public endpoint to access the instance over the public network.	2019-06-30	All	*#unique_7*

May 2019

Feature	Description	Release date	Suppor region	Topic
Tag management	You can bind tags to ApsaraDB for Redis instances to classify instances by tag.	2019-05-14	All	*#unique_8*

April 2019

Feature	Description	Release date	Suppor region	Topic
Support for adding internal IP addresses of ECS instances to a whitelist	You can add internal IP addresses of Elastic Compute Service (ECS) instances to a whitelist on the whitelist configuration page.	2019-04-27	All	*#unique_9*

February 2019

Feature	Description	Release date	Suppor region	Topic
Cross-zone migration	You can migrate an ApsaraDB for Redis instance from one zone to another zone in the same region.	2019-02-28	All	*#unique_10*

January 2019

Feature	Description	Release date	Suppor region	Topic
Indonesia (Jakarta) zone A+B	The zone A+B in the Indonesia (Jakarta) region is available.	2019-01-17	All	None
China (Hangzhou) zone H+I	The zone H+I in the China (Hangzhou) region is available.	2019-01-10	All	None

3 Product series

3.1 Overview

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

Type	Description	Scenario
Standalone instances	**A standalone instance runs in a single-node structure.**	· **Cache-only applications.** **Note:** **If your application is sensitive to data reliability, we recommend that you do not use the single-replica edition.** · **Compatibility with Redis protocols.** · **Limited queries per seconds (QPS) performance.** · **Redis commands are simple and contain few sorting and compute commands.**
Master-replica instances	**A master-replica instance runs in a master-replica replication 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.**

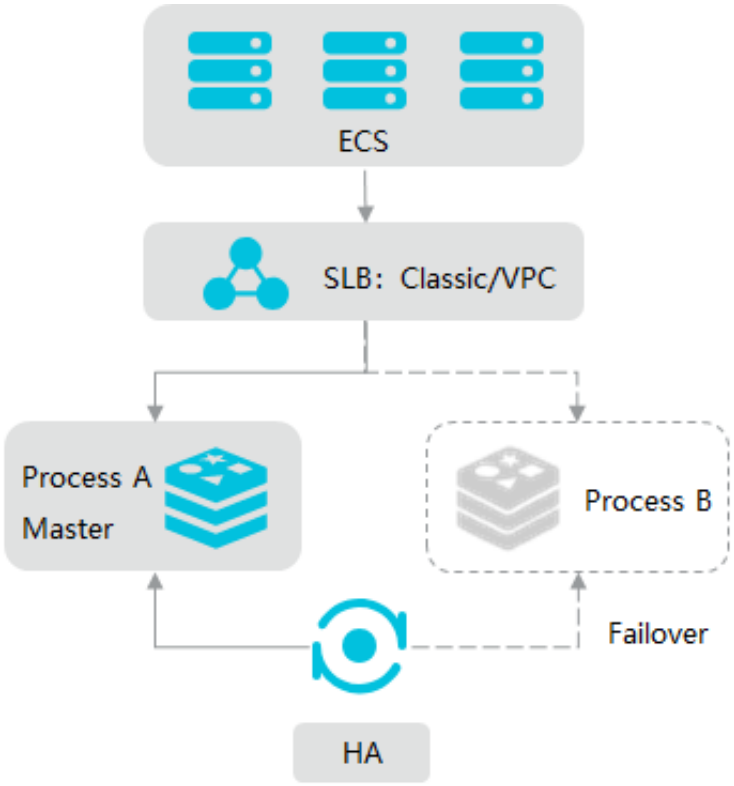
Type	Description	Scenario
Standalone Cluster	A standalone cluster instance is deployed in a cluster architecture. Each shard server runs in the single-node mode.	· Large data volumes. · Cache-only applications. **Note:** If your application is sensitive to data reliability, we recommend that you do not use the single-replica cluster edition. · High-QPS applications. · Throughput-intensive applications. · Cache-only applications that do not require data reliability.
Master-replica cluster	A master-replica cluster instance is deployed in a cluster architecture. Each shard server runs in the master-replica mode.	· Large data volumes. · High-QPS applications. · Throughput-intensive applications.
Read/write splitting edition	A read/write splitting instance consists of the proxy servers, master and replica nodes, and read-only nodes.	· High-QPS read requests. · Compatibility with Redis protocols. · Persistent data storage based on ApsaraDB for Redis.

3.2 Standalone instances

ApsaraDB for Redis standalone instances have no replica, but act great in caching scenarios that do not require high data reliability.

Overview

The standalone instance is a new type of ApsaraDB for Redis instance. This instance runs in a master-only structure. Compared with master-replica instances, this structure contains no replica for synchronizing data in real time, and does not support data persistence and replication policies. This edition is applicable to cache-only scenarios that do not require data reliability.



Features

The standalone instance is deployed in a master-only structure. The proprietary high-availability (HA) system detects service status on each node. If the service is not available, the HA system starts a new Redis process to take over the service and ensure high availability of the service. The standalone instance provides more cost-effective benefits.

Scenarios

- **Cache-only applications**

  The standalone instance runs in a master-only structure. If the master fails, the system automatically starts a new Redis process that contains no data to take over the service. After the failover operation, your application has to warm up data to prevent traffic bursts on the backend database.

  Note:
  The standalone edition cannot ensure data reliability. Your application has to warm up data after the master fails. If your application is sensitive to data reliability, we recommend that you do not use the standalone instance. Instead, you can select a high-availability master-replica instance.

- **Compatibility with Redis protocols**

  The standalone instance is fully compatible with Redis protocols, so you can smoothly migrate your business.
- **Limited QPS performance**

  The original Redis database runs in a single-thread mechanism based on a single -core CPU. In scenarios where QPS is 80,000, 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**

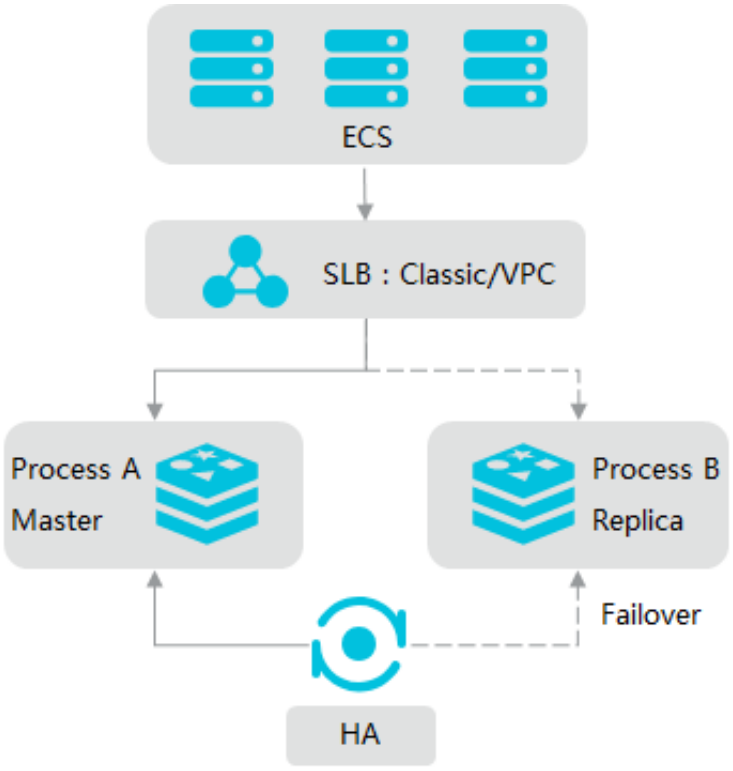
  CPU may be bottlenecked due to Redis single-thread mechanism. If you require plenty of sorting and compute operations, we recommend that you select a cluster instance.

3.3 Master-replica instances

ApsaraDB for Redis master-replica instances provide not only high-performance caching service, but also high data reliability.

Overview

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



Benefits

- **Reliability**
  - **Reliable services**

    In the master-replica architecture, the master the and replica run on different physical servers. The master 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 fails, the proprietary HA system performs the failover operation to ensure stability of services.
  - **Reliable data**

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

  The master-replica instance is developed on the basis of open-source Redis and 100% compatible with Redis commands. You can smoothly migrate an on-premises Redis database to an ApsaraDB for Redis master-replica instance. 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 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 and the replica . 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 runs the Partial Resynchronization (PSYNC) command to resynchronize partial data. During this process, the resynchronization fails. The master has to fully synchronize RDB files to the replica.
    - To fully synchronize RDB files, the master has to perform full replication first due to the single-thread model. As a result, the master 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. This may run out of memory and cause the application to exit abnormally.
    - The replica files that the master 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 master-replica instance is fully compatible with Redis protocols, so you can smoothly migrate your business.
- **Persistent data storage based on ApsaraDB for Redis**

  The master-replica instance 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 instance.
- **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 instance.

3.4 Standalone Cluster

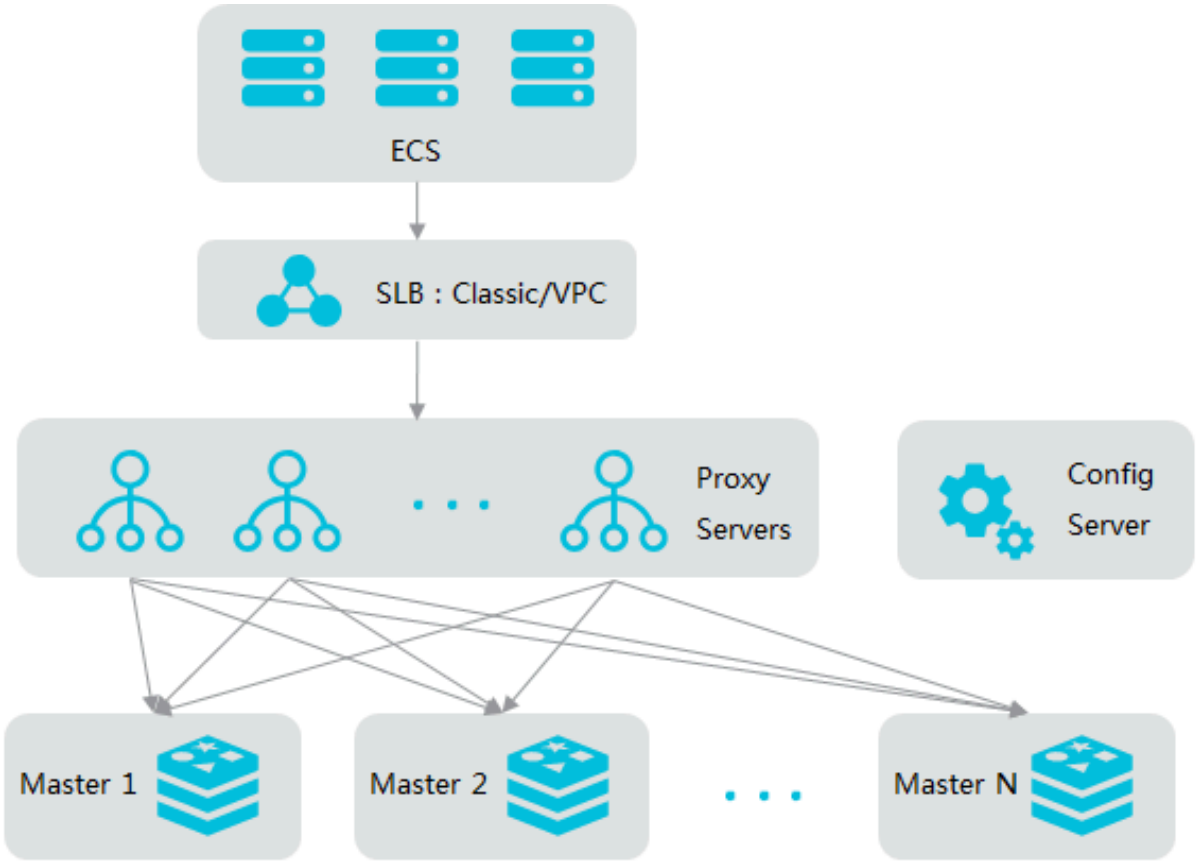
ApsaraDB for Redis standalone clusters are proxy based clusters. Each shard in a cluster has one node and no replica.

Overview

In cache-only or high-queries per second (QPS) scenarios, ApsaraDB for Redis provides standalone cluster instances to eliminate the bottleneck of Redis single-thread model and easily process large-capacity or high-performance services. Compared with the master-replica cluster edition, standalone cluster instances provide services in a cost-effective way.

Components

An ApsaraDB for Redis cluster 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 single-node structure. This structure contains no replica node for synchronizing data in real time, and does not support data persistence and replication policies. If a shard fails, the system automatica lly starts a new Redis process within 30 seconds to ensure high availability of services.

- **Configuration server**

  The configuration server is used to store cluster configurations and partitioni ng policies. The server runs in a dual-node 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 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 standalone cluster instance, or upgrade a standalone instance to a standalone cluster.

Scenarios

- Large data volumes

  The cluster edition of ApsaraDB for Redis supports data capacity scaling. Compared with the master-replica 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.
- Cache-only applications

  A shard server of the standalone cluster runs in a single-node structure. If a shard fails, the system automatically starts a new Redis process that contains no data to take over the service. Data on the faulty shard is lost after the failover. You have to connect to the background database to obtain data and cause traffic bursts on the database. Therefore, you must prepare warm-up and protection mechanisms for your application.

   **Notice:**
  The standalone cluster edition cannot ensure data reliability. Your application has to warm up data after a shard fails. If your application is sensitive to data reliability, we recommend that you do not use the standalone cluster edition.
- High-QPS applications

  A master-replica 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 multiple types of cluster configurations, for more information, see *Specifications and performance*.

- **Throughput-intensive applications**

  Compared with the master-replica edition, the cluster edition provides higher throughput over the Alibaba Cloud intranet. You can easily read hot data and provide high-throughput services by using the cluster edition.

- **Applications insensitive to Redis protocols**

  The cluster edition contains multiple components in the service architecture, and has more restrictions on the use of Redis protocols. For more information, see *#unique_18*.

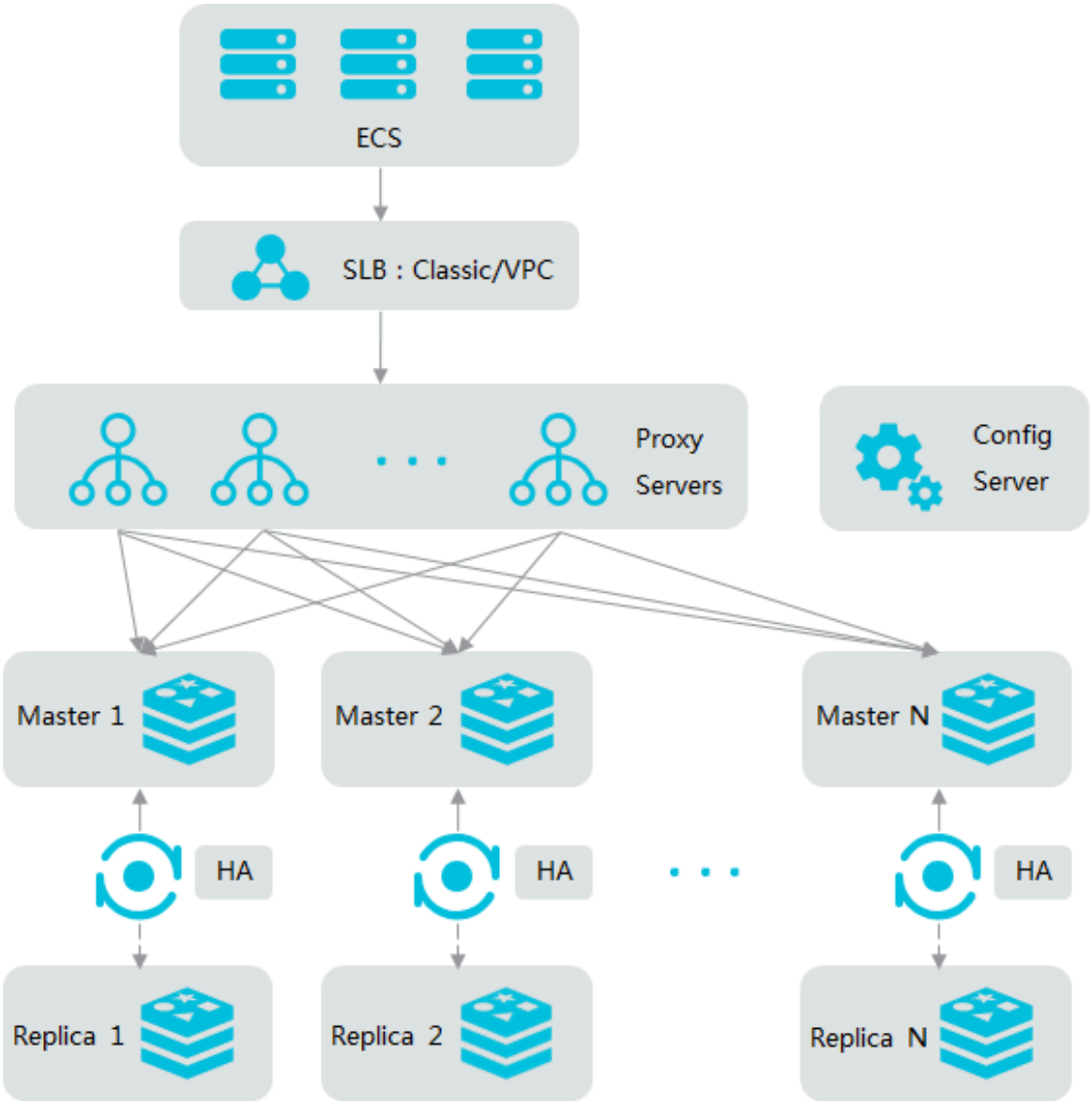
3.5 Master-replica cluster

ApsaraDB for Redis master-replica clusters are proxy based clusters. Each shard in a cluster has one master node and one replica node.

Overview

ApsaraDB for Redis provides master-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



- **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 master-replica high-availability structure. If the master 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 master-replica high-availability structure to ensure high availability of services.

> **Warning:**
>
> - **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 *Specifications 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 instance, or upgrade a master-replica instance to a cluster instance.**

Scenarios

- **Large data volumes**

  The cluster edition of ApsaraDB for Redis supports data capacity scaling. Compared with the master-replica instance, 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**

  The cluster edition contains multiple components in the service architecture, and has more restrictions on the use of Redis protocols. For more information, see *#unique_18*.

FAQ

- **For more information about memory usage exceptions on child nodes of the cluster edition, see** *#unique_19*.
- **For more information about data distribution in memory, see** *#unique_20*.

3.6 Read/write splitting edition

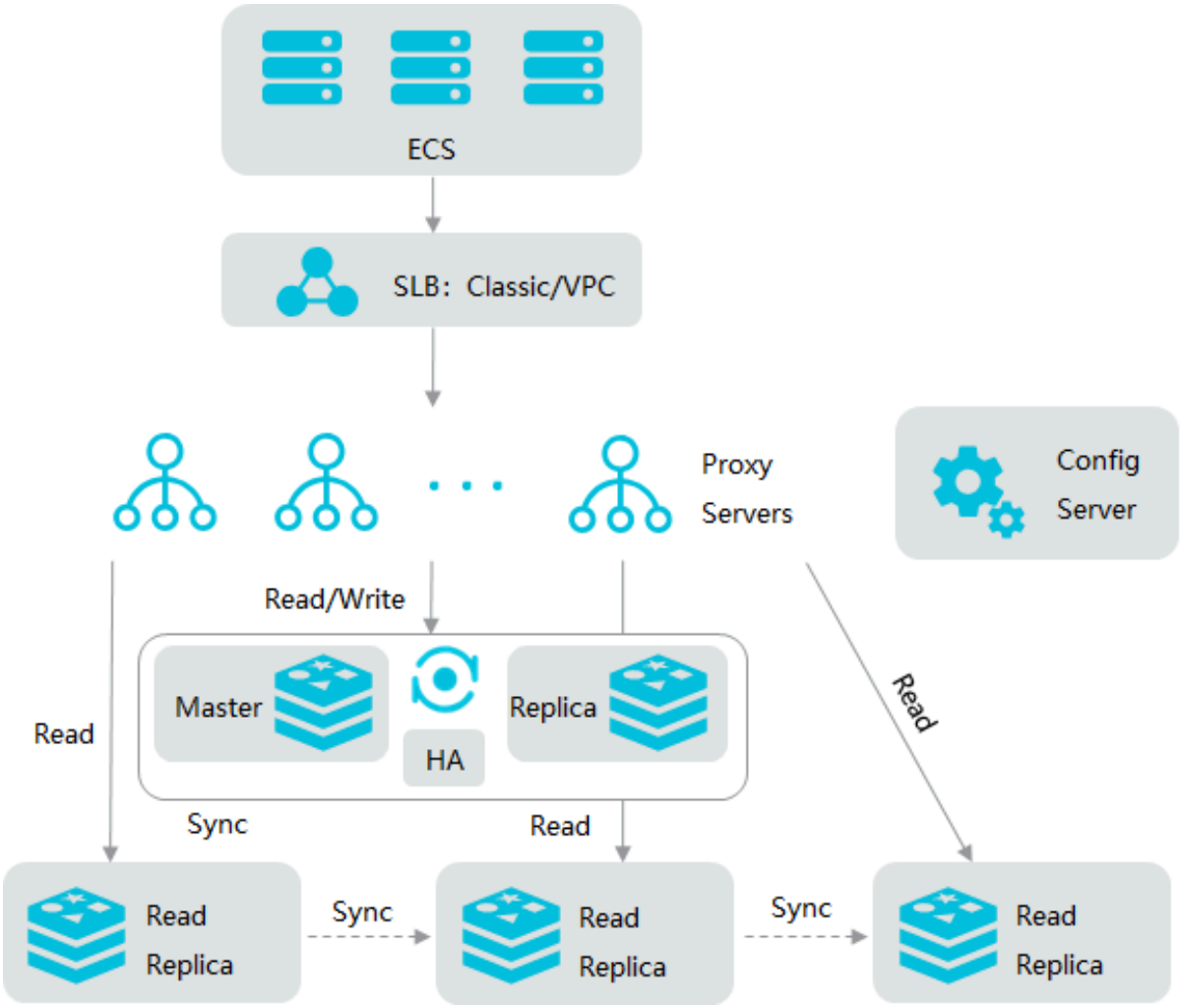
ApsaraDB for Redis read/write splitting edition can provide appropriate support in more-reading and less-writing scenarios.

Overview

A read/write splitting instance of ApsaraDB for Redis is applicable to scenarios where read operations are more than write operations. This instance provides high-availability, high-performance and flexible read/write splitting services. You can easily process centralized hot data and highly concurrent read operations, and minimize operations and maintenance (O&M) costs.

Components

A read/write splitting instance consists of the proxy servers, one or more primary nodes, and one or three or five read replicas.



The replica of the primary node works as a hot standby node and does not provide services for your business. Read replicas process read requests. Proxy servers forward read and write requests to the primary node or a read replica according to weight calculations. The weight assignment is automatic and cannot be customized .

Note:

The system evenly assigns read requests to the primary node and read replicas. For example, if you have purchased an instance with three read replicas, the weight of read requests is 25% for the primary node and 25% for each of the three read replicas.

The high-availability (HA) system automatically monitors the health of each node. If a node fails, the HA system performs the failover operation or reconstructs read replica, and updates the corresponding routing and weight information.

The read/write splitting edition supports the chained replication architecture. With the increase of read replicas, the processing performance of the overall system improves on a linear scale. Alibaba Cloud has optimized source code of replication processes of ApsaraDB for Redis. In this way, the system can maximize the stability of linear replications.

When an application connects to the read/write splitting edition, a proxy server automatically recognizes the types of read and write requests that the client initiates. Afterward, the proxy server performs load balancing based on weight calculations, and forwards write requests to the master node and read operations to the corresponding read-only nodes.

The read/writing splitting edition of ApsaraDB for Redis is developed on the basis of Redis open-source protocols and 100% compatible with Redis commands. You can easily and smoothly upgrade a master-replica instance of ApsaraDB for Redis to a read/write splitting instance. You can also migrate an on-premises Redis database to a read/write splitting instance.

Benefits

- **High availability**
  - **The read/write splitting edition of ApsaraDB for Redis uses a proprietary HA system to automatically monitor the health of all data nodes and ensure high availability of system services. If the master node is not available, the system automatically selects another master node and reconstructs the replication topology. If a read-only node fails, the HA system automatically detects the failure and starts another read-only node to synchronize data. The faulty node has to go offline.**
  - **The proxy server monitors real-time service status of each read-only instance . If a read-only instance fails, the proxy server module automatically reduces the service weight of this faulty node. If the read-only node fails consecutiv ely for certain times, the proxy server module terminates the service right of the faulty node. The proxy server module continues to monitor the faulty node , and when this node recovers from the failure, starts the service of the node again.**

- **High performance**

  The read/write splitting edition supports the chained replication architecture . With the increase of read-only instances, the processing performance of the overall system improves on a linear scale. In this way, the system can make full use of physical resources of each read-only node.

Scenarios

- **Read requests of high queries per second (QPS)**

  A master-replica edition of ApsaraDB for Redis cannot process high QPS. In the scenarios where read operations are more than write operations, you can deploy multiple read replicas to eliminate the performance bottleneck of Redis single -thread model. The read/write splitting edition of ApsaraDB for Redis supports one, three, or five read replicas. Therefore, the QPS performance can be five times that of the master-replica edition in an ideal scenario.
- **Compatibility with Redis protocols**

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

  The read/write splitting edition supports data persistence, replication and recovery to ensure data reliability.

3.7 Instructions of ApsaraDB for Redis read/write splitting edition

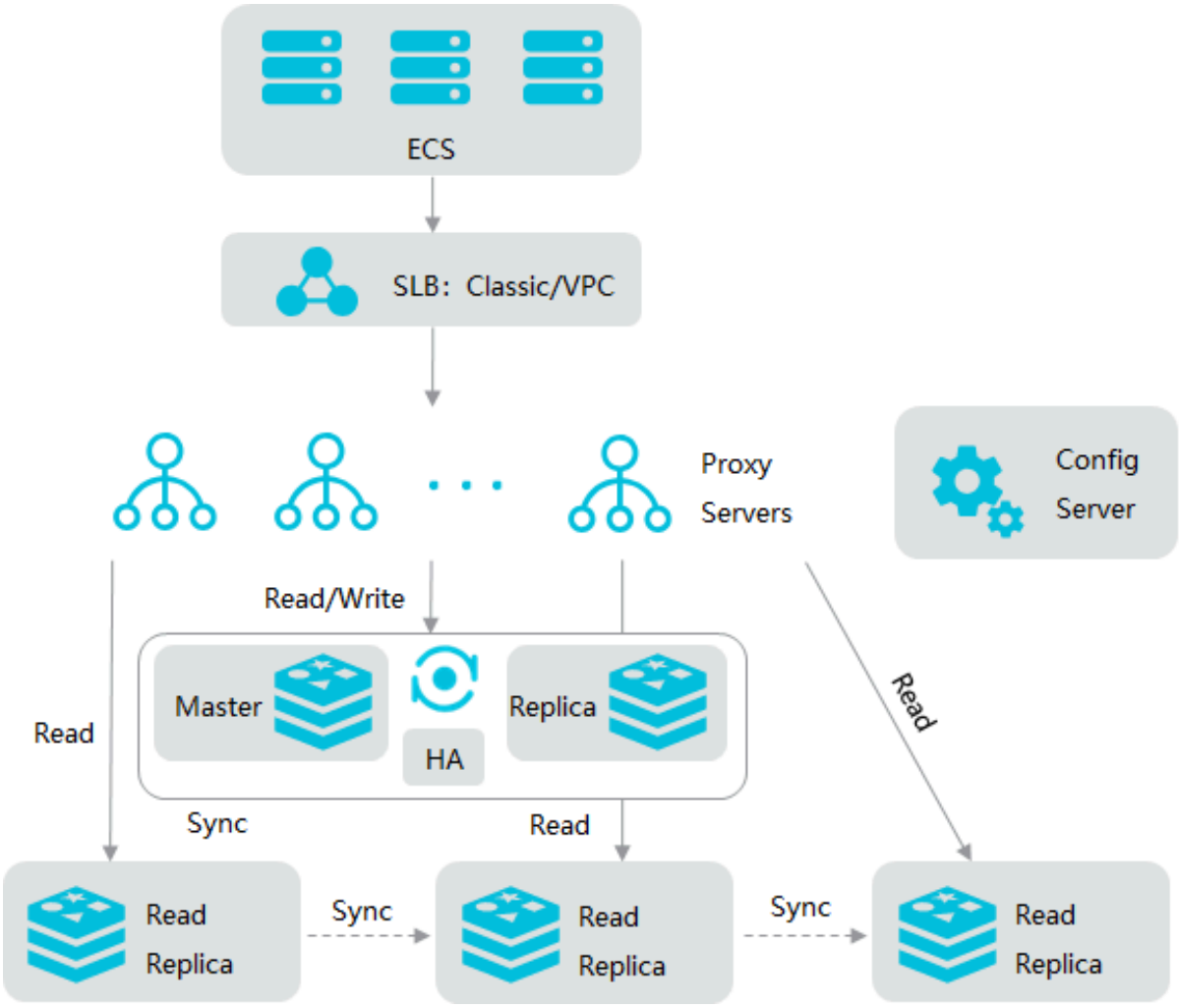
ApsaraDB for Redis read/write splitting edition is different from ApsaraDB for Redis standard edition and cluster edition in the architecture and data processing methods. This topic describes the working methods of each component in the read/write splitting architecture and the instructions of ApsaraDB for Redis read/write splitting edition.

Architecture

- **ApsaraDB for Redis read/write splitting edition provides one or more read replicas.**
- **ApsaraDB for Redis read/write splitting edition supports one-way chained replication between the primary node and read replicas.**

- **The proxy automatically distributes requests to the primary node and read replicas as follows:**
  - **Distributes write requests to the primary node.**
  - **Distributes read requests to the primary node and read replicas based on the loads.**
- **Distributes requests from Lua scripts and transactions to the primary node.**
- **The high-availability (HA) module monitors the status of the primary node and read replicas and changes the shards upon failovers.**

Figure 3-1: Architecture of ApsaraDB for Redis Read/Write Splitting Edition



Instructions

- **If a read replica fails, requests are forwarded to other read replicas. If all read replicas are unavailable, requests are forwarded to the primary node. Exceptions of a read replica incur higher workloads on the primary node and more**

response time for requests. Therefore, we recommend that you use multiple read replicas for the businesses that generate a large number of read requests.

- If an exception occurs on a read replica, the HA module will suspend the service of the replica, and add a new read replica. The process of enabling the new read replica involves resource allocation, data synchronization, and system loading. The time consumed depends on system workloads and data size. ApsaraDB for Redis does not guarantee the time when the new read replica can start providing the service.
- Full synchronization between read replicas is triggered in certain scenarios, such as failovers. During full synchronization, read replicas do not provide services. If your query requests are forwarded to read replicas, the following error message is returned: `-LOADING Redis is loading the dataset in memory\r\n`.
- The primary node conforms to the *ApsaraDB for Redis Service Level Agreement* of ApsaraDB for Redis.

4 Specifications and performance

This topic describes the types and parameters of all ApsaraDB for Redis editions and the method of testing queries per second (QPS). When calling APIs related to instance types, you may need to query`InstanceClass`Parameters.

Note:

- **The value of intranet bandwidth applies to both upstream and downstream. If network resources are sufficient, the bandwidth is unlimited for ApsaraDB for Redis instances. However, if network resources are insufficient, the intranet bandwidth takes effect for the instances.**
- **The intranet bandwidth of cluster instances = the Number of shards × the intranet bandwidth of a single shard (96 MB/s). The maximum intranet bandwidth of an instance is 2048 MB/s.**
- **The maximum concorrent connections of cluster instances = the Number of shards × the maximum concorrent connections per shard (10000). The maximum number of maximum concorrent connections is 500000.**

Master-replica instances

Size (GB)	InstanceClass (for calling APIs)	New connects per second	Maximu concorr connects	Intranet bandwic (MB/s)	CPU processi capabili	QPS referenc value	Description
1 GB master-replica	redis.master.small.default	10,000	10,000	10	Single-core	80,000	Master-replica instance
2 GB master-replica	redis.master.mid.default	10,000	10,000	16	Single-core	80,000	Master-replica instance

Size (GB)	InstanceClass (for calling APIs)	New connects per second	Maximu concorr connects	Intranet bandwic (MB/s)	CPU processi capabili	QPS referenc value	Description
4 GB master-replica	redis.master.stand.default	10,000	10,000	24	Single-core	80,000	Master-replica instance
8 GB master-replica	redis.master.large.default	10,000	10,000	24	Single-core	80,000	Master-replica instance
16 GB master-replica	redis.master.2xlarge.default	10,000	10,000	32	Single-core	80,000	Master-replica instance
16 GB master-replica	redis.master.4xlarge.default	10,000	10,000	32	Single-core	80,000	Master-replica instance
64 GB master-replica	redis.master.8xlarge.default	10,000	10,000	48	Single core	80,000	Master-replica instance

Master-replica performance-enhanced edition

Type	InstanceClass (for calling APIs)	Number of I/O threads	New connections per second	Maximum concurrent connections	Intranet bandwidth (MB/s)	CPU processing capability	QPS reference value	Description
1 GB master-replica performance-enhanced	redis.amber.master.small.multithread	6	10,000	10,000	96	Single-core	240,000	Master-replica performance-enhanced instance
2 GB master-replica performance-enhanced	redis.amber.master.mid.multithread	6	10,000	10,000	96	Single-core	240,000	Master-replica performance-enhanced instance
4 GB master-replica performance-enhanced	redis.amber.master.stand.multithread	6	10,000	10,000	96	Single-core	240,000	Master-replica performance-enhanced instance
8 GB master-replica performance-enhanced	redis.amber.master.large.multithread	6	10,000	10,000	96	Single-core	240,000	Master-replica performance-enhanced instance
16 GB master-replica performance-enhanced	redis.amber.master.2xlarge.multithread	6	10,000	10,000	96	1	240,000	Master-replica performance-enhanced instance

Type	InstanceClass (for calling APIs)	Number of I/O threads	New connections per second	Maximum concurrent connections	Intranet bandwidth (MB/s)	CPU processing capability	QPS reference value	Description
32 GB master-replica performance-enhanced	redis.amber.master.4xlarge.multithread	6	10,000	10,000	96	1	240,000	Master-replica performance-enhanced instance
64 GB master-replica performance-enhanced	redis.amber.master.8xlarge.multithread	6	10,000	10,000	96	1	240,000	Master-replica performance-enhanced instance

Master-replica hybrid-storage edition

Type	InstanceClass (for calling APIs)	New connections per second	Maximum concurrent connections	Intranet bandwidth (MB/s)	QPS reference value	Description
16 GB memory 32 GB disk master-replica	redis.amber.master.16g.2x.ext4.default	10,000	10,000	96	100,000	Hybrid-storage master-relica instance
16 GB memory 64 GB disk master-replica	redis.amber.master.16g.4x.ext4.default	10,000	10,000	96	100,000	Hybrid-storage master-relica instance

Type	InstanceClass (for calling APIs)	New connections per second	Maximum concurrent connections	Intranet bandwidth (MB/s)	QPS reference value	Description
16 GB memory 128 GB disk master-replica	redis.amber.master.16g.8x.ext4.default	10,000	10,000	96	100,000	Hybrid-storage master-relica instance
16 GB memory 256 GB disk master-replica	redis.amber.master.16g.16x.ext4.default	10,000	10,000	96	100,000	Hybrid-storage master-relica instance
32 GB memory 64 GB disk master-replica	redis.amber.master.32g.2x.ext4.default	10,000	10,000	96	100,000	Hybrid-storage master-relica instance
32 GB memory 128 GB disk master-replica	redis.amber.master.32g.4x.ext4.default	10,000	10,000	96	100,000	Hybrid-storage master-relica instance
32 GB memory 256 GB disk master-replica	redis.amber.master.32g.8x.ext4.default	10,000	10,000	96	100,000	Hybrid-storage master-relica instance
32 GB memory 512 GB disk master-replica	redis.amber.master.32g.16x.ext4.default	10,000	10,000	96	100,000	Hybrid-storage master-relica instance

Type	InstanceClass (for calling APIs)	New connections per second	Maximum concurrent connections	Intranet bandwidth (MB/s)	QPS reference value	Description
64 GB memory 128 GB disk master-replica	redis.amber.master.64g.2x.ext4.default	10,000	10,000	96	100,000	Hybrid-storage master-replica instance
64 GB memory 256 GB disk master-replica	redis.amber.master.64g.4x.ext4.default	10,000	10,000	96	100,000	Hybrid-storage master-replica instance
64 GB memory 512 GB disk master-replica	redis.amber.master.64g.8x.ext4.default	10,000	10,000	96	100,000	Hybrid-storage master-replica instance
64 GB memory 1024 GB disk master-replica	redis.amber.master.64g.16x.ext4.default	10,000	10,000	96	100,000	Hybrid-storage master-replica instance

Master-replica zone-disaster recovery edition

Note:

- **We recommend that you try *Master-replica instances* before using the zone-disaster recovery edition.**
- **To create a zone-disaster recovery instance, you have to select the region and zone that support zone-disaster recovery, such asChina (Hangzhou) Zone (G + H).**

Type	InstanceClass (for calling APIs)	New connections per second	Maximum concurrent connections	Intranet bandwidth (MB/s)	CPU processing capability	QPS reference value	Description
1 GB zone-disaster recovery	redis.logic.sharding.drredissdb1g.1db.0rodb.4proxy.default	10,000	10,000	10	Single-core	80,000	Master-replica zone-disaster recovery instances
2 GB zone-disaster recovery	redis.logic.sharding.drredissdb2g.1db.0rodb.4proxy.default	10,000	10,000	16	Single-core	80,000	Master-replica zone-disaster recovery instances
4 GB zone-disaster recovery	redis.logic.sharding.drredissdb4g.1db.0rodb.4proxy.default	10,000	10,000	24	Single-core	80,000	Master-replica zone-disaster recovery instances
8 GB zone-disaster recovery	redis.logic.sharding.drredissdb8g.1db.0rodb.4proxy.default	10,000	10,000	24	Single-core	80,000	Master-replica zone-disaster recovery instances
16 GB zone-disaster recovery	redis.logic.sharding.drredissdb16g.1db.0rodb.4proxy.default	10,000	10.000	32	Single-core	80,000	Master-replica zone-disaster recovery instances

Type	InstanceClass (for calling APIs)	New connects per second	Maximu concorr connect s	Intrane bandwi (MB/s)	CPU process capabil	QPS referen value	Description
32 GB zone-disaster recovery	redis.logic.sharding.drredissdb32g.1db.0rodb.4proxy.default	10,000	10,000	32	Single-core	80,000	Master-replica zone-disaster recovery instances
64 GB zone-disaster recovery	redis.logic.sharding.drredissdb64g.1db.0rodb.4proxy.default	10,000	10,000	48	Single-core	80,000	Master-replica zone-disaster recovery instances

Standalone instances

Type	InstanceClass (for calling APIs)	New connects per second	Maximu concorr connect s	Intranet bandwic (MB/s)	CPU processi capabili	QPS referenc value	Description
1 GB standalone	redis.basic.small.default	10,000	10,000	10	Single-core	80,000	Master-only instances, with no replica
2 GB standalone	redis.basic.mid.default	10,000	10,000	16	Single-core	80,000	Master-only instances, with no replica

Type	InstanceClass (for calling APIs)	New connects per second	Maximum concurrent connects	Intranet bandwidth (MB/s)	CPU processing capability	QPS reference value	Description
4 GB standalone	redis.basic.stand.default	10,000	10,000	24	Single-core	80,000	Master-only instances, with no replica
8 GB standalone	redis.basic.large.default	10,000	10,000	24	Single-core	80,000	Master-only instances, with no replica
16 GB standalone	redis.basic.2xlarge.default	10,000	10,000	32	Single-core	80,000	Master-only instances, with no replica
32 GB standalone	redis.basic.4xlarge.default	10,000	10,000	32	Single-core	80,000	Master-only instances, with no replica

Master-replica cluster instances

Type	InstanceClass (for calling APIs)	Number of shards	New connections per second	Maximum concurrent connections	Intranet bandwidth (MB/s)	CPU processing capability	QPS reference value	Description
16 GB cluster	redis.logic.sharding.2g.8db.0rodb.8proxy.default	8	50,000	80,000	768	8-core	640,000	High-performance cluster instance
32 GB cluster	redis.logic.sharding.4g.8db.0rodb.8proxy.default	8	50,000	80,000	768	8-core	640,000	High-performance cluster instance
64 GB cluster	redis.logic.sharding.8g.8db.0rodb.8proxy.default	8	50,000	80,000	768	8-core	640,000	High-performance cluster instance
128 Gb cluster	redis.logic.sharding.8g.16db.0rodb.16proxy.default	16	50,000	160,000	1,536	16-core	1,280,000	High-performance cluster instance

Type	InstanceClass (for calling APIs)	Number of shards	New connections per second	Maximum concurrent connections	Intranet bandwidth (MB/s)	CPU processing capability	QPS reference value	Description
256 GB cluster	redis.logic.sharding.16g.16db.0rodb.16proxy.default	16	50,000	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	50,000	320,000	2,048	32-core	2,560,000	High-performance cluster instance

Note:
The number of shards in a master-replica cluster is the number of master shards.

Master-replica cluster performance-enhanced edition

Type	InstanceClass (for calling APIs)	Number of I/O thread	Number of shards	New connections per second	Maximum concurrent connections	Intranet bandwidth (MB/s)	CPU processing capability	QPS reference value	Description
16 GB performance-enhanced cluster	redis.amber.logic.sharding.2g.8db.0rodb.24proxy.multithread	6	8	50,000	80,000	768	8-core	1,920,000	Performance-enhanced cluster instance
16 GB performance-enhanced cluster	redis.amber.logic.sharding.4g.8db.0rodb.24proxy.multithread	6	8	50,000	80,000	768	8-core	1,920,000	Performance-enhanced cluster instance
64 GB performance-enhanced cluster	redis.amber.logic.sharding.8g.8db.0rodb.24proxy.multithread	6	8	50,000	80,000	768	8-core	1,920,000	Performance-enhanced cluster instance

Type	InstanceClass (for calling APIs)	Number of I/O threads	Number of shards	New connections per second	Maximum connections	Intranet bandwidth (MB/s)	CPU processing capability	QPS reference value	Description
128 GB performance-enhanced cluster	redis.amber.logic.sharding.8g.16db.0rodb.48proxy.multithread	6	16	50,000	160,000	1,536	16-core	3,840,000	Performance-enhanced cluster instance
256 GB performance-enhanced cluster	redis.amber.logic.sharding.16g.16db.0rodb.48proxy.multithread	6	16	50,000	160,000	1,536	16-core	3,840,000	Performance-enhanced cluster instance
512 GB performance-enhanced cluster	redis.amber.logic.sharding.16g.32db.0rodb.96proxy.multithread	6	32	50,000	320,000	2,048	32-core	7,680,000	Performance-enhanced cluster instance

Type	InstanceClass (for calling APIs)	Number of I/O thread	Number of shards	New connections per second	Maximum concurrent connections	Intranet bandwidth (MB/s)	CPU processing capability	QPS reference value	Description
1024 GB performance-enhanced cluster	redis.amber.logic.sharding.16g.64db.0rodb.192proxy.multithread	6	64	50,000	500,000	2,048	64-core	15,360,000	Performance-enhanced cluster instance
2048 GB performance-enhanced cluster	redis.amber.logic.sharding.16g.128db.0rodb.384proxy.multithread	6	128	50,000	500,000	2,048	128-core	30,480,000	Performance-enhanced cluster instance
4096 GB performance-enhanced cluster	redis.amber.logic.sharding.16g.256db.0rodb.768proxy.multithread	6	256	50,000	500,000	2,048	256-core	60,960,000	Performance-enhanced cluster instance

Master-replica cluster zone-disaster recovery edition

Note:

- **We recommend that you try *Master-replica cluster instances* before using the zone-disaster recovery edition.**
- **To create a zone-disaster recovery instance, you have to select the region and zone that support zone-disaster recovery, such asChina (Hangzhou) Zone (G + H).**

Type	InstanceClass (for calling APIs)	Number of shards	New connections per second	Maximum concurrent connections	Intranet bandwidth (MB/s)	CPU processing capability	QPS reference value	Description
16 GB zone-disaster recovery cluster	redis.logic.sharding.drredismdb16g.8db.0rodb.8proxy.default	8	50,000	80,000	768	8-core	800,000	Zone-disaster recovery cluster instance
32 GB zone-disaster recovery cluster	redis.logic.sharding.drredismdb32g.8db.0rodb.8proxy.default	8	50,000	80,000	768	8-core	800,000	Zone-disaster recovery cluster instance
64 GB zone-disaster recovery cluster	redis.logic.sharding.drredismdb64g.8db.0rodb.8proxy.default	8	50,000	80,000	768	8-core	800,000	Zone-disaster recovery cluster instance

Type	InstanceClass (for calling APIs)	Number of shards	New connections per second	Maximum concurrent connections	Intranet bandwidth (MB/s)	CPU processing capability	QPS reference value	Description
128 GB zone-disaster recovery cluster	redis.logic.sharding.drredismdb128g.16db.0rodb.16proxy.default	16	50,000	160,000	1,536	16-core	1,600,000	Zone-disaster recovery cluster instance
256 GB zone-disaster recovery cluster	redis.logic.sharding.drredismdb256g.16db.0rodb.16proxy.default	16	50,000	160,000	1,536	16-core	1,600,000	Zone-disaster recovery cluster instance
512 GB zone-disaster recovery cluster	redis.logic.sharding.drredismdb512g.32db.0rodb.32proxy.default	32	50,000	320,000	2,048	32-core	3,200,000	Zone-disaster recovery cluster instance

Standalone cluster

Type	InstanceClass (for calling APIs)	Number of shards	New connections per second	Maximum concurrent connections	Intranet bandwidth (MB/s)	CPU processing capability	QPS reference value	Description
16 GB standalone cluster	redis.sharding.basic.small.default	8	50,000	80,000	768	8-core	640,000	High-performance cluster instance
32 GB standalone cluster	redis.sharding.basic.mid.default	8	50,000	80,000	768	8-core	640,000	High-performance cluster instance
64 GB standalone cluster	redis.sharding.basic.large.default	8	50,000	80,000	768	8-core	640,000	High-performance cluster instance
128 GB standalone cluster	redis.sharding.basic.2xlarge.default	16	50,000	160,000	1,536	16-core	1,280,000	High-performance cluster instance
256 GB standalone cluster	redis.sharding.basic.4xlarge.default	16	50,000	160,000	1,536	16-core	1,280,000	High-performance cluster instance
512 GB standalone cluster	redis.sharding.basic.8xlarge.default	32	50,000	320,000	2,048	32-core	2,560,000	High-performance cluster instance

Note:

To create a standalone cluster, select the ApsaraDB for Redis engine version 2.8 only. This restriction is not applicable to a master-replica cluster.

Read/write splitting instances

Type	InstanceClass (for calling APIs)	Number of shards	Intranet bandwidth (MB/s)	New connections per second	Maximum concurrent connections	QPS reference value
1 GB read/write splitting(1 primary shard with 1 read replica)	redis.logic.splitrw.small.1db.1rodb.4proxy.default	1	192	20,000	20,000	200,000
1 GB read/write splitting(1 primary shard with 3 read replicas)	redis.logic.splitrw.small.1db.3rodb.4proxy.default	3	384	40,000	40,000	400,000
1 GB read/write splitting(1 primary shard with 5 read replicas)	redis.logic.splitrw.small.1db.5rodb.6proxy.default	5	576	50,000	60,000	600,000
2 GB read/write splitting(1 primary shard with 1 read replica)	redis.logic.splitrw.mid.1db.1rodb.4proxy.default	1	192	20,000	20,000	200,000
2 GB read/write splitting(1 primary shard with 3 read replicas)	redis.logic.splitrw.mid.1db.3rodb.4proxy.default	3	384	20,000	20,000	400,000
2 GB read/write splitting(1 primary shard with 5 read replicas)	redis.logic.splitrw.mid.1db.5rodb.6proxy.default	5	576	50,000	60,000	600,000

Type	InstanceClass (for calling APIs)	Number of shards	Intranet bandwidth (MB/s)	New connections per second	Maximum concurrent connections	QPS reference value
4 GB read/write splitting(1 primary shard with 1 read replica)	redis.logic.splitrw.stand.1db.1rodb.4proxy.default	1	192	20,000	20,000	200,000
4 GB read/write splitting(1 primary shard with 3 read replicas)	redis.logic.splitrw.stand.1db.3rodb.4proxy.default	3	384	40,000	40,000	400,000
4 GB read/write splitting(1 primary shard with 5 read replicas)	redis.logic.splitrw.stand.1db.5rodb.6proxy.default	5	576	50,000	60,000	600,000
8 GB read/write splitting(1 primary shard with 1 read replica)	redis.logic.splitrw.large.1db.1rodb.4proxy.default	1	192	20,000	20,000	200,000
8 GB read/write splitting(1 primary shard with 3 read replicas)	redis.logic.splitrw.large.1db.3rodb.4proxy.default	3	384	40,000	40,000	400,000
8 GB read/write splitting(1 primary shard with 5 read replicas)	redis.logic.splitrw.large.1db.5rodb.6proxy.default	5	576	50,000	60,000	600,000

Type	InstanceClass (for calling APIs)	Number of shards	Intranet bandwidth (MB/s)	New connections per second	Maximum concurrent connections	QPS reference value
16 GB read/write splitting(1 primary shard with 1 read replica)	redis.logic.splitrw.2xlarge.1db.1rodb.4proxy.default	1	192	20,000	20,000	200,000
16 GB read/write splitting(1 primary shard with 3 read replicas)	redis.logic.splitrw.2xlarge.1db.3rodb.4proxy.default	3	384	40,000	40,000	400,000
16 GB read/write splitting(1 primary shard with 5 read replicas)	redis.logic.splitrw.2xlarge.1db.5rodb.6proxy.default	5	576	50,000	60,000	600,000
32 GB read/write splitting(1 primary shard with 1 read replica)	redis.logic.splitrw.4xlarge.1db.1rodb.4proxy.default	1	192	20,000	20,000	200,000
32 GB read/write splitting(1 primary shard with 3 read replicas)	redis.logic.splitrw.4xlarge.1db.3rodb.4proxy.default	3	384	40,000	40,000	400,000

Type	InstanceClass (for calling APIs)	Number of shards	Intranet bandwidth (MB/s)	New connections per second	Maximum concorrent connections	QPS reference value
32 GB read/write splitting(1 primary shard with 5 read replicas)	redis.logic.splitrw.4xlarge.1db.5rodb.6proxy.default	5	576	50,000	60,000	600,000
64 GB read/write splitting(1 primary shard with 1 read replica)	redis.logic.splitrw.8xlarge.1db.1rodb.4proxy.default	1	192	20,000	20,000	200,000
64 GB read/write splitting(1 primary shard with 3 read replicas)	redis.logic.splitrw.8xlarge.1db.3rodb.4proxy.default	3	384	40,000	40,000	400,000
64 GB read/write splitting(1 primary shard with 5 read replicas)	redis.logic.splitrw.8xlarge.1db.5rodb.6proxy.default	5	576	50,000	60,000	600,000

Read/write splitting performance-enhanced edition

Type	InstanceClass (for calling APIs)	Number of I/O threads	Number of shards	Intranet bandwidth (MB/s)	New connects per second	Maximum concurrent connects	QPS reference value
1 GB performance-enhanced read/write splitting(1 primary shard with 1 read replica)	redis.amber.logic.splitrw.small.1db.1rodb.6proxy.multithread	6	1	192	20,000	20,000	600,000
1 GB performance-enhanced read/write splitting(1 primary shard with 3 read replicas)	redis.amber.logic.splitrw.small.1db.3rodb.12proxy.multithread	6	3	384	40,000	40,000	1,200,000
1 GB performance-enhanced read/write splitting(1 primary shard with 5 read replicas)	redis.amber.logic.splitrw.small.1db.5rodb.18proxy.multithread	6	5	576	50,000	60,000	1,800,000

Type	InstanceClass (for calling APIs)	Number of I/O threads	Number of shards	Intranet bandwid (MB/s)	New connects per second	Maximu concorr connects	QPS reference value
2 GB performance-enhanced read/write splitting(1 primary shard with 1 read replica)	redis.amber.logic.splitrw.mid.1db.1rodb.6proxy.multithread	6	1	192	20,000	20,000	600,000
2 GB performance-enhanced read/write splitting(1 primary shard with 3 read replicas)	redis.amber.logic.splitrw.mid.1db.3rodb.12proxy.multithread	6	3	384	20,000	20,000	1,200,000
2 GB performance-enhanced read/write splitting(1 primary shard with 5 read replicas)	redis.amber.logic.splitrw.mid.1db.5rodb.18proxy.multithread	6	5	576	50,000	60,000	1,800,000
4 GB performance-enhanced read/write splitting(1 primary shard with 1 read replica)	redis.amber.logic.splitrw.stand.1db.1rodb.6proxy.multithread	6	1	192	20,000	20,000	600,000

Type	InstanceClass (for calling APIs)	Number of I/O threads	Number of shards	Intranet bandwid (MB/s)	New connects per second	Maximu concorr connects	QPS reference value
4 GB performance-enhanced read/write splitting(1 primary shard with 3 read replicas)	redis.amber.logic.splitrw.stand.1db.3rodb.12proxy.multithread	6	3	384	40,000	40,000	1,200,000
4 GB performance-enhanced read/write splitting(1 primary shard with 5 read replicas)	redis.amber.logic.splitrw.stand.1db.5rodb.18proxy.multithread	6	5	576	50,000	60,000	1,800,000
8 GB performance-enhanced read/write splitting(1 primary shard with 1 read replica)	redis.amber.logic.splitrw.large.1db.1rodb.6proxy.multithread	6	1	192	20,000	20,000	600,000

Type	InstanceClass (for calling APIs)	Number of I/O threads	Number of shards	Intranet bandwid (MB/s)	New connects per second	Maximu concorr connects	QPS reference value
8 GB performance-enhanced read/write splitting(1 primary shard with 3 read replicas)	redis.amber.logic.splitrw.large.1db.3rodb.12proxy.multithread	6	3	384	40,000	40,000	1,200,000
8 GB performance-enhanced read/write splitting(1 primary shard with 5 read replicas)	redis.amber.logic.splitrw.large.1db.5rodb.18proxy.multithread	6	5	576	50,000	60,000	1,800,000
16 GB performance-enhanced read/write splitting(1 primary shard with 1 read replica)	redis.amber.logic.splitrw.2xlarge.1db.1rodb.6proxy.multithread	6	1	192	20,000	20,000	600,000

Type	InstanceClass (for calling APIs)	Number of I/O threads	Number of shards	Intranet bandwid (MB/s)	New connects per second	Maximu concorr connects	QPS reference value
16 GB performance-enhanced read/write splitting(1 primary shard with 3 read replicas)	redis.amber.logic.splitrw.2xlarge.1db.3rodb.12proxy.multithread	6	3	384	40,000	40,000	1,200,000
16 GB performance-enhanced read/write splitting(1 primary shard with 5 read replicas)	redis.amber.logic.splitrw.2xlarge.1db.5rodb.18proxy.multithread	6	5	576	50,000	60,000	1,800,000
32 GB performance-enhanced read/write splitting(1 primary shard with 1 read replica)	redis.amber.logic.splitrw.4xlarge.1db.1rodb.6proxy.multithread	6	1	192	20,000	20,000	600,000

Type	InstanceClass (for calling APIs)	Number of I/O threads	Number of shards	Intranet bandwid (MB/s)	New connects per second	Maximu concorr connects	QPS reference value
32 GB performance-enhanced read/write splitting(1 primary shard with 3 read replicas)	redis.amber.logic.splitrw.4xlarge.1db.3rodb.12proxy.multithread	6	3	384	40,000	40,000	1,200,000
32 GB performance-enhanced read/write splitting(1 primary shard with 5 read replicas)	redis.amber.logic.splitrw.4xlarge.1db.5rodb.18proxy.multithread	6	5	576	50,000	60,000	1,800,000
64 GB performance-enhanced read/write splitting(1 primary shard with 1 read replica)	redis.amber.logic.splitrw.8xlarge.1db.1rodb.6proxy.multithread	6	1	192	20,000	20,000	600,000

Type	InstanceClass (for calling APIs)	Number of I/O threads	Number of shards	Intranet bandwidth (MB/s)	New connects per second	Maximum concurrent connects	QPS reference value
64 GB performance-enhanced read/write splitting(1 primary shard with 3 read replicas)	redis.amber.logic.splitrw.8xlarge.1db.3rodb.12proxy.multithread	6	3	384	40,000	40,000	1,200,000
64 GB performance-enhanced read/write splitting(1 primary shard with 5 read replicas)	redis.amber.logic.splitrw.8xlarge.1db.5rodb.18proxy.multithread	6	5	576	50,000	60,000	1,800,000

Read/write splitting clusters

Type	InstanceClass (for calling APIs)	Number of shards	Number of read replicas	Intranet bandwidth (MB/s)	New connects per second	Maximum concurrent connects	QPS reference value
4 GB read/write splitting cluster (2 primary shards and 2 read replicas)	redis.logic.splitrw.sharding2g.2db.1rodb.4proxy.default	2	1	384	50,000	40,000	400,000

Type	InstanceClass (for calling APIs)	Number of shards	Number of read replicas	Intranet bandwid (MB/s)	New connects per second	Maximu concorr connects	QPS reference value
8 GB read/write splitting cluster (2 primary shards and 2 read replicas)	redis.logic.splitrw.sharding4g.2db.1rodb.4proxy.default	2	1	384	50,000	40,000	400,000
16 GB read/write splitting cluster(2 primary shards and 2 read replicas)	redis.logic.splitrw.sharding8g.2db.1rodb.8proxy.default	2	1	384	50,000	40,000	400,000
32 GB read/write splitting cluster(8 primary shards and 8 read replicas)	redis.logic.splitrw.sharding4g.8db.1rodb.16proxy.default	8	1	1,536	50,000	160,000	1,600,000
64 GB read/write splitting cluster(16 primary shards and 16 read replicas)	redis.logic.splitrw.sharding4g.16db.1rodb.32proxy.default	16	1	2,048	50,000	320,000	3,200,000
128 GB read/write splitting cluster(16 primary shards and 16 read replicas)	redis.logic.splitrw.sharding8g.16db.1rodb.32proxy.default	16	1	2,048	50,000	320,000	3,200,000

Type	InstanceClass (for calling APIs)	Number of shards	Number of read replicas	Intranet bandwidth (MB/s)	New connects per second	Maximum concurrent connects	QPS reference value
256 GB read/write splitting cluster(32 primary shards and 32 read replicas)	redis.logic.splitrw.sharding8g.32db.1rodb.64proxy.default	32	1	2,048	50,000	500,000	6,400,000
512 GB read/write splitting cluster(32 primary shards and 32 read replicas)	redis.logic.splitrw.sharding16g.32db.1rodb.64proxy.default	32	1	2,048	50,000	500,000	6,400,000

QPS performance reference

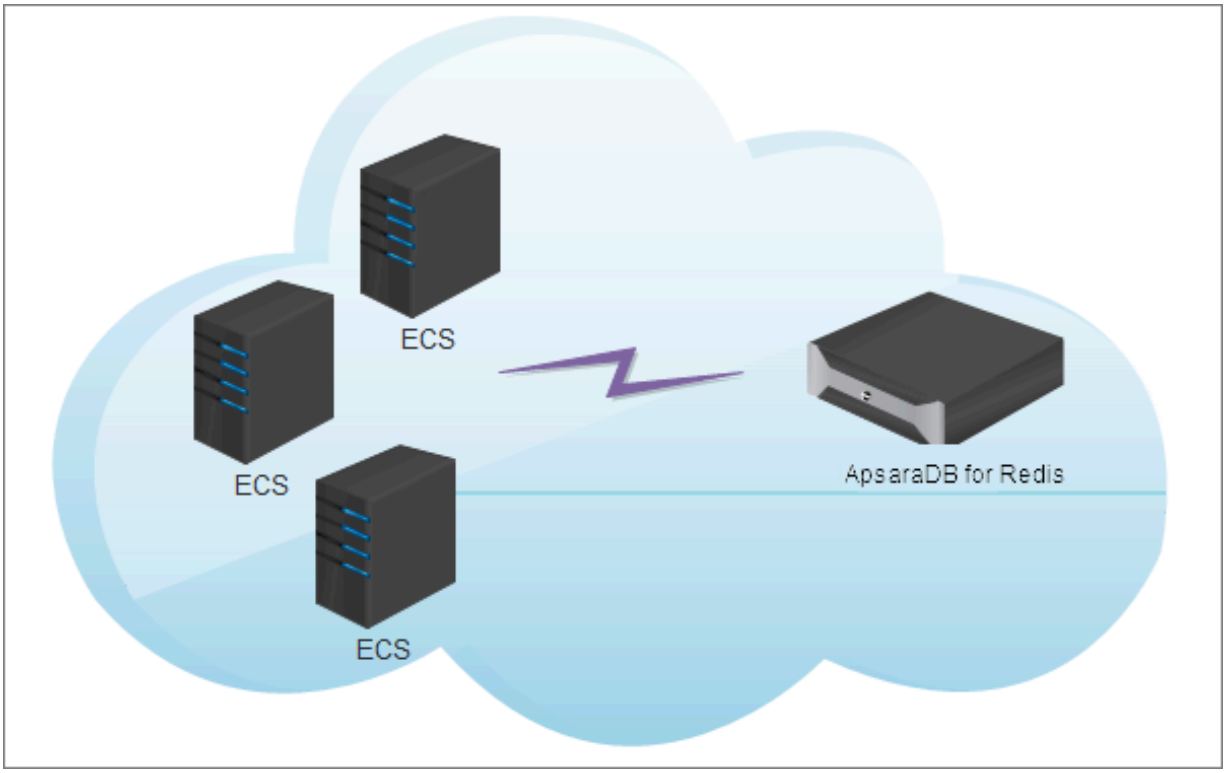
QPS performance

Type	Maximum concorrent connections	Intranet bandwidth(MB/s)	CPU processing capability	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 shards multiplied by the value from 80,000 to 100,000.

QPS testing method

Figure 4-1: Network topology



ECS instance type

Operating system	CPU processing capability	Memory	Region	Number of hosts
Ubuntu 14.04 64-bit	1-core	2048 MB	China (Shenzhen)	3

Steps

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

```
$ 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.**

5 Disaster recovery

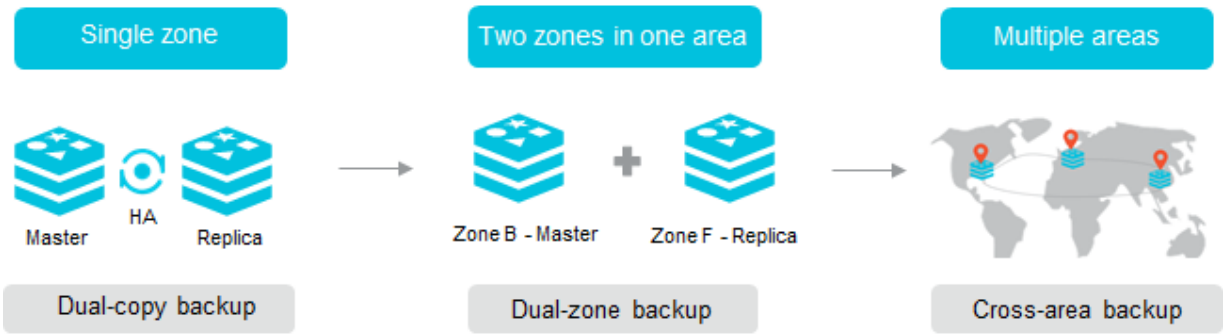
Data is the core element for most businesses. As a carrier of data, a database plays a decisive role in businesses. ApsaraDB for Redis is a high-performance key-value database system. The database stores large amounts of important data for businesses. This topic describes the mechanism of disaster recovery 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 5-1: 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.

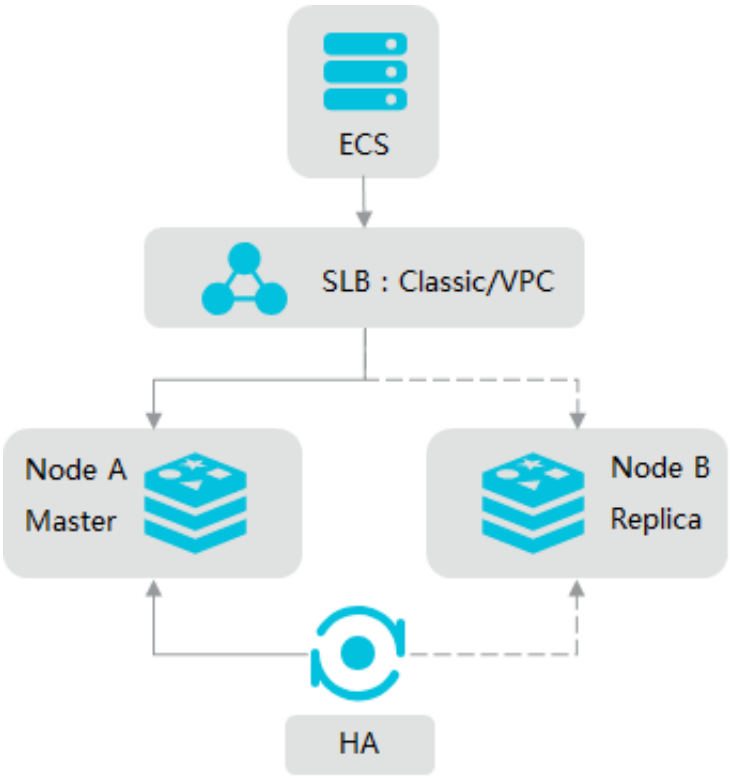
Single-zone high-availability mechanism

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 5-2: High-availability architecture of the master-replica instance



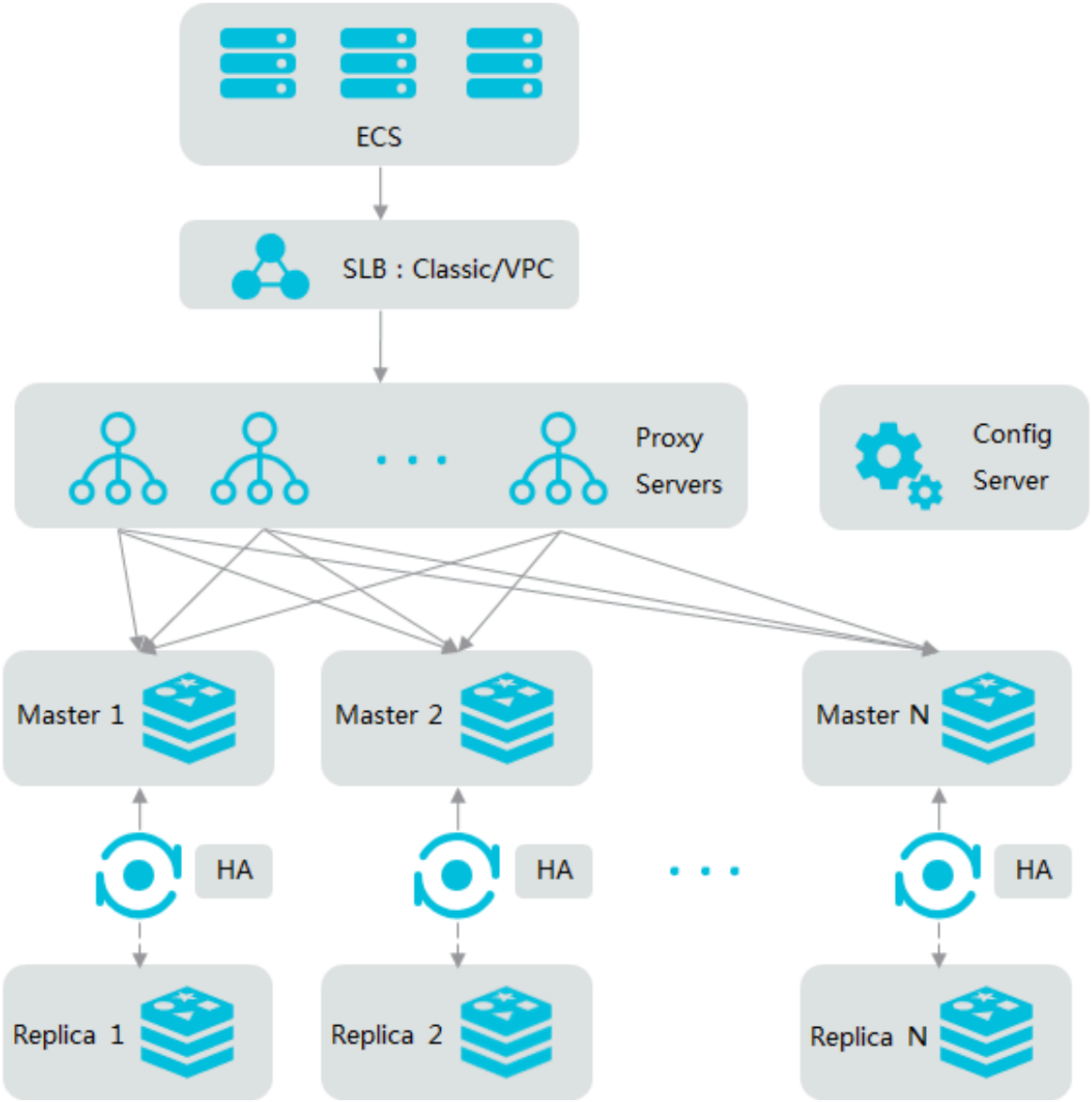
Master-replica instances

A *Master-replica 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 5-3: High-availability architecture of master-replica clusters



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 5-4: 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.

6 Features

ApsaraDB for Redis supports multiple architechture types, data persistence, high availability, elastic scaling and intelligent O&M.

Flexible architecture

Single-node architecture

This architecture is suitable for cache-only scenarios. Based on this architecture , you can perform flexible upgrading or downgrading for a single-node cluster to meet high queries per second (QPS) requirements in a cost-efficient manner.

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.

7 Scenarios

ApsaraDB for Redis provides great support for high QPS(queris per second) 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 overwhelmed 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.

Master-replica 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 .

8 Terms

You can find instructions for ApsaraDB for Redis related terms in this topic.

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.

9 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 `ZRANGEBYLEX` command. However, the `ZRANGEBYLEX` 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 initialization.
       6) (integer) 9942928
       7) "replication.backlog"    //The memory of the backlog used
in resuming an interrupted master-replica replication. Default value
: 10 MB.
       8) (integer) 1048576
       9) "clients.slaves"    // The memory used in a master-replica
 replication.
      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) persistence 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) "5.5934348106384277"
      27) "peak.percentage"    // 100 * total.allocated / peak_alloc
ated
      28) "2.6251003742218018"
      29) "fragmentation"    //The memory fragmentation ratio.
      30) "1.1039986610412598"
```

- `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 fragmentation ratio.
  High fragmentation: fragmentation > 1.4. This indicates a high
memory fragmentation 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:000000000002' found so far with counter 87
[00.00%] Hot key 'key:000000000001' found so far with counter 254
[00.00%] Hot key 'mylist' found so far with counter 107
[00.00%] Hot key 'key:000000000000' found so far with counter 254
[45.45%] Hot key 'counter:000000000001' found so far with counter 87
[45.45%] Hot key 'key:000000000002' found so far with counter 254
[45.45%] Hot key 'myset' found so far with counter 64
[45.45%] Hot key 'counter:000000000000' found so far with counter 93
-------- summary -------
Sampled 22 keys in the keyspace!
hot key found with counter: 254 keyname: key:000000000001
hot key found with counter: 254 keyname: key:000000000000
hot key found with counter: 254 keyname: key:000000000002
hot key found with counter: 107 keyname: mylist
hot key found with counter: 93  keyname: counter:000000000000
hot key found with counter: 87  keyname: counter:000000000002
hot key found with counter: 87  keyname: counter:000000000001
hot key found with counter: 64  keyname: myset
```

10 Version description

10.1 Feature updates of ApsaraDB for Redis 5.0

ApsaraDB for Redis 5.0 significantly optimizes kernel performance and system stability. It currently supports new features, such as the Redis Streams data type, account management, and audit logging, to meet the needs in diverse scenarios.

Update description

- **Supports a new data type: Redis Streams. For more information, see *Redis Streams*.**
- **Supports *#unique_29*.**
- **Supports the log management feature. You can manage *audit logs*, *operational logs*, and *slow logs*. You can query the records of commands used to manage databases, read/write operations, and sensitive operations such as KEYS and FLUSHALL operations. You can also monitor slow logs.**
- **Supports timer, cluster, and dictionary APIs.**
- **Supports the least frequently used (LFU) and least recently used (LRU) caching strategies for Redis database (RDB) files.**
- **Supports the Redis command line interface (redis-cli) instead of redis-trib.rb to manage clusters. You need to write command code by using C language instead of Ruby language.**
- **Supports the following commands for sorted sets: *ZPOPMIN*, *ZPOPMAX*, *BZPOPMIN*, and *BZPOPMAX*.**
- **Supports Active Defragmentation v2.**
- **Supports enhanced performance of HyperLogLog.**
- **Supports optimized statistical reports of the memory.**
- **Supports the HELP subcommand for various commands that can include subcommands.**
- **Supports performance stability when connections between databases and clients are frequently disabled and enabled.**
- **Supports Jemalloc 5.1.0.**
- **Supports the *CLIENT ID* and *CLIENT UNBLOCK* commands.**
- **Supports the *LOLWUT* command that is used to produce interesting outputs.**

- Discards the term "slave" in all scenarios unless you need to ensure backward compatibility of APIs.
- Optimizes the network layer.
- Improves Lua script-related configuration.
- Supports the dynamic-hz parameter to optimize CPU usage and response performance.
- Reconstructs and improves Redis core code.

 **Note:**
ApsaraDB for Redis 5.0 only supports the Standard Edition. We are working on support for other editions.

10.2 ApsaraDB for Redis 4.0 release notes

This topic describes the feature updates in ApsaraDB for Redis 4.0.

Updates in 2019

Date	Description
July 30, 2019	· Physical IP addresses are not restricted by the value of the maxclients parameter.
July 08, 2019	· The following issue is fixed: The memory used by audit logs is released repeatedly.
July 04, 2019	· The RPOPLPUSH coredump issue is fixed. · The latency record is added for event looping.
June 20, 2019	· The audit log feature is optimized to prevent the loss of operations logs and frequent startup of the bio_audit thread.
June 13, 2019	· Key expiration and eviction are optimized. · When password-free access is enabled, you still need to pass password verification if you access ApsaraDB for Redis instances over the public network. · The DEL command is replaced with the UNLINK command .

Date	Description
May 05, 2019	· The page cache is released automatically after the AOF file is written or the RDB or AOF file is loaded. · The master role is distinguished from the replica role, and management connections are distinguished from user connections in audit logs.
March 11, 2019	· The following issue is fixed: The connection counter for a database in a cluster does not work properly in some scenarios.
February 21, 2019	· The following issue is fixed: Slow logs are not collected in some scenarios.
January 24, 2019	· Jemalloc is upgraded from 4.0.3 to 5.0.
January 24, 2019	· The *audit log* feature is supported. This feature records the write operations and other sensitive operations in logs. · Multithreading protection is added for some resources, and resource isolation is optimized. · The exception handling feature of Redis is optimized to make sure that exception logs are generated properly. · The following issue is fixed: An exception occurs due to a null pointer when you run commands with long parameters.
January 07, 2019	· The issue of abnormal log rotation is fixed. · The exception of the slow log feature caused by an initialization error is fixed.

Updates in 2018

Date	Description
December 11, 2018	· The debug assert command is optimized. · The following issue is fixed: The master node is affected when a replica client runs commands. · The issue that may cause memory leakage is fixed.
December 11, 2018	· The ROLE command is available, which is used to view the role of the current node. · The SENTINEL command is available.

Date	Description
November 15, 2018	· Issues related to initializing and writing the AOF file are fixed.
October 17, 2018	· If the check-whitelist-always parameter is set to yes for an ApsaraDB for Redis instance, the whitelist is checked when other services in the same VPC connect to the instance even if password-free access is enabled. · The following issue is fixed: The output of the INFO command is abnormal.
October 10, 2018	· The configuration items related to server startup are optimized. · The system checks whether the AOF file list is NULL when obtaining the last AOF binlog during startup.
September 28, 2018	· The timeout issue for status management during cluster migration is fixed.
September 28, 2018	· The following issue is fixed: An exception occurs under special circumstances when the BRPOPLPUSH command in the AOF file is loaded during startup.

10.3 ApsaraDB for Redis 5.0 release notes

This topic describes the feature updates in ApsaraDB for Redis 5.0.

Updates in 2019

Date	Description
June 30, 2019	· Physical IP addresses are not restricted by the value of the maxclients parameter. · When password-free access is enabled, you still need to pass password verification if you access ApsaraDB for Redis instances over the public network.
June 20, 2019	· The adverse impacts on the expiration and eviction algorithms when the number of databases reaches 256 are reduced.

Date	Description
March 22, 2019	· Compatible with open-source Redis 5.0.3. · The features of ApsaraDB for Redis 4.0 are supported. · The following client connection types are supported: dbaas, user, and slave. · The memory leakage issue that occurs at an extremely low probability is fixed. · The print issue is fixed for keys that are larger than 128 bytes.