ApsaraDB for Redis

Product Introduction

Product Introduction

ApsaraDB for Redis is compatible with open-source Redis protocol standards and provides persistent memory database services. Based on its high-reliability dual-machine hot standby architecture and seamlessly scalable cluster architecture, this service can meet the needs of businesses that require high read/write performance and flexible capacity adjustment.

ApsaraDB for Redis supports many data types including String, List, Set, SortedSet, and Hash, and provides advanced functions such as Transactions and Pub/Sub.

Using memory + hard disk storage, ApsaraDB for Redis can meet your persistence requirements, while providing high-speed data read/write capability.

ApsaraDB for Redis supports flexible architecture deployment. The single-copy, dual-copy, and cluster version architectures suit different business scenarios.

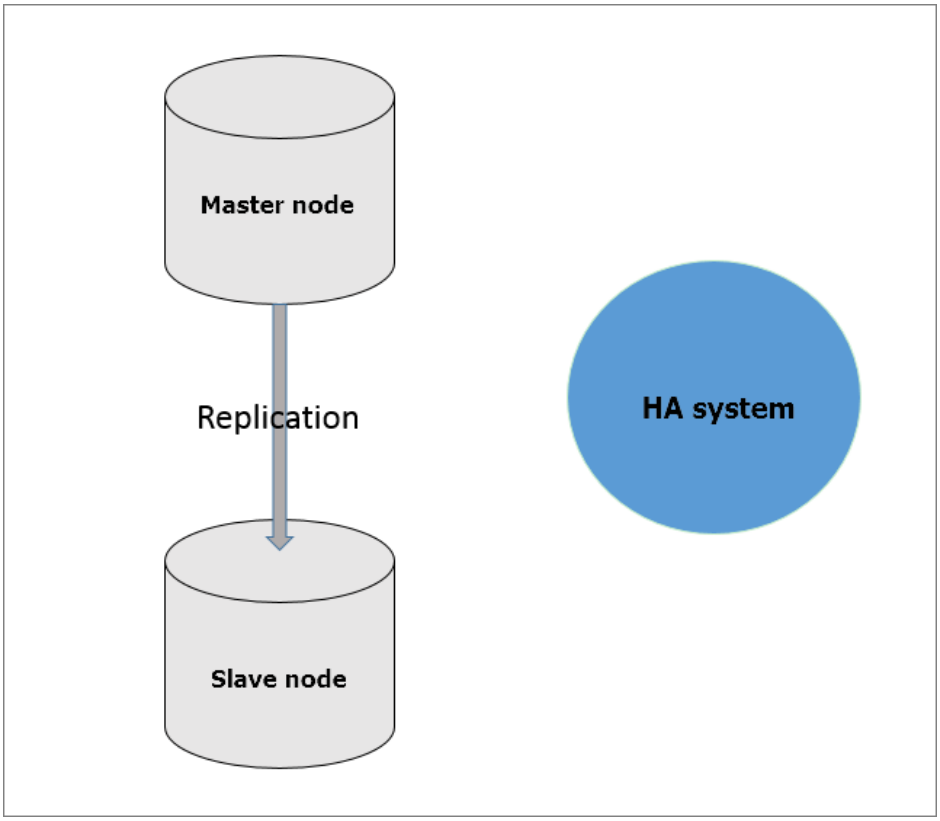
> **Single-node architecture**: This is suitable for cache-only scenarios. It supports flexible configuration changes for single-node clusters and provides cost-effective performance that suits high QPS scenarios.
>
> **Dual-node hot-standby architecture**: During system operation, data is synchronized between the master and slave nodes. If the master node fails, the system automatically switches over to the slave node in a matter of seconds. The entire process is automatic without affecting your business. The master/slave architecture guarantees the high availability of system services.
>
> **Cluster architecture**: Cluster instances adopt a distributed architecture, with each node working in master/slave mode. This supports automatic disaster recovery switchover and failover.You can choose from multiple cluster specifications based on your business needs and the service allows you to scale database performance without limit.

ApsaraDB for Redis is used as a cloud computing service, with hardware and data deployed on Alibaba Cloud, supported by comprehensive infrastructure planning, network security protection, and system maintenance services. This service enables you to focus fully on business innovation.

Product series

Introduction

The ApsaraDB for Redis standard version adopts a dual-copy structure and works in master-slave replication mode. The master node provides routine service access, while the slave node guarantees high service availability. If the master node fails, the system will automatically switch over to the slave node within 30 seconds, to guarantee smooth business operation.



Features

Reliability

Reliable service

The service adopts a dual-host master/slave architecture, with the master and slave nodes located on different physical machines. The master node provides external access, allowing you to use the Redis command line and universal clients to add, delete, modify, and query data. If the master node fails, the self-developed HA system will automatically switch over to the slave node, to guarantee smooth business operation.

Data reliability

The data persistence function is enabled by default, so all data is written to the disk. The service supports the data backup feature that allows you to clone or roll instances back to a backup set. This provides an effective solution to the problems such as incorrect data operations.

Compatibility

Developed on Redis 2.8, the ApsaraDB for Redis standard version is fully compatible with Redis protocol commands. Self-built Redis databases can be smoothly migrated to Redis standard version. In addition, we provide a data transmission tool (DTS) for incremental Redis migration, to guarantee the stable transition of your business.

Developed by Alibaba Cloud

Failover system (HA)

Alibaba Cloud's Redis service includes the HA failover system. Whenever an exception is detected on the master node, this system immediately switches over to the slave node to guarantee high service availability. This is an effective solution for disk I/O faults, CPU faults, and other problems that lead to service exceptions.

Master/Slave replication mechanism

Alibaba Cloud has customized Redis' master/slave replication mechanism by using an incremental log format to replicate and transmit data. Under this mechanism, interrupted master/slave replication has limited impact on system performance and stability, thereby avoiding the disadvantages of Redis' native replication mechanism.

Problems with Redis' native replication mechanism:

When Redis replication is interrupted, the Slave will immediately initiate psync. If psync's attempt to deploy synchronization fails, all RDB data will be synchronized and sent to the Slave node.

Full Redis synchronization may cause the master node to run full backup, and the Fork process may cause a lag of milliseconds or seconds on the master node.

The Redis Fork process causes Copy-On-Write, which consumes the master node's thread memory. In extreme cases, this can cause the

master node’ s out-of-memory program to exit abnormally.

The backup files generated by the Redis master node consume the server’ s disk I/O and CPU (compression) resources.

Sending backup files of several GBs may overload the server’ s network egress and drive up the disk’ s sequential I/O throughput. This in turn may delay the response to normal business requests (and result in other chain reactions).

Scenarios

Businesses that require strong compatibility with Redis protocol

The standard version is completely compatible with Redis protocol, so you can smoothly migrate your business.

Using Redis for persistent data storage

The standard version provides a persistence mechanism and backup & recovery mechanism to guarantee high data reliability.

Controlling individual Redis performance load

Running under a single thread mechanism, Redis is recommended for businesses that require a performance capacity of under 100,000 QPS. For a higher performance capacity, please select the cluster version.

Simple Redis commands and few sort and algorithm-type commands

Because of Redis’ single-thread system, CPU performance is the primary limiting factor. The cluster version is preferred for businesses involving many sort and algorithm-type commands.

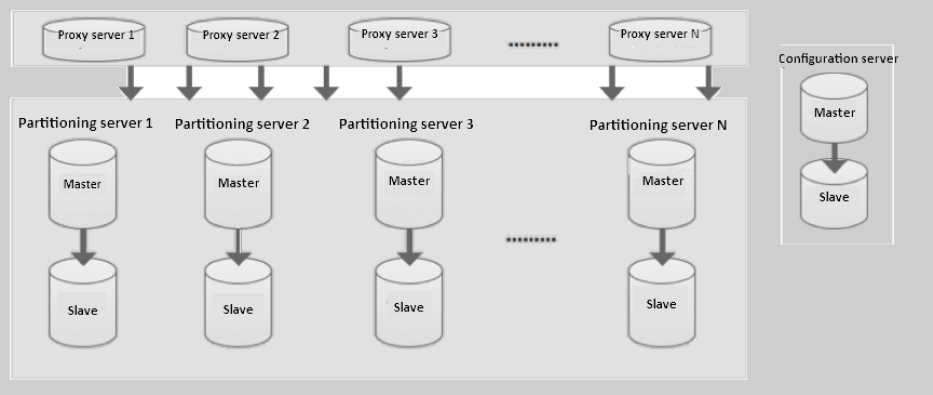
Introduction

ApsaraDB for Redis provides cluster version instances, allowing you to easily break through Redis’ single thread bottleneck. This is the preferred solution for businesses that require large Redis capacity or high performance. The ApsaraDB for Redis cluster version has built-in data partitions and reading

algorithms. The overall process is transparent to users, saving you from the headache of developing and operating Redis clusters.

Components

The ApsaraDB for Redis cluster version is composed of three components: proxy servers (service proxy), partition servers, and configuration servers.



Proxy servers

In single-node configuration, the cluster consists of multiple proxies. The system will automatically perform load balancing and failover for the cluster.

Partition servers

Each partition server is built on dual-copy highly available architecture. When the master node fails, the system will automatically switch to the standby node to guarantee the high availability of the service.

Configuration servers

Configuration servers are used to store cluster configuration information and partition policies. These servers currently adopt dual-copy high availability architecture, to guarantee high service availability.

Note:

- The quantities and configurations of these three components are specified by the system when you purchase the cluster version of the corresponding specification. Now, you cannot freely modify these options. The specifications are shown as follows:

Cluster version specification	Proxy count	Partition server count	Memory size of each partition server
16 GB cluster	8	8	2 GB

version			
32 GB cluster version	8	8	4 GB
64 GB cluster version	8	8	8 GB
128 GB cluster version	16	16	8 GB
256 GB cluster version	16	16	16 GB

Redis cluster versions expose a unified access domain name. You can visit this domain name for normal Redis access and data operations. Proxy servers, partition servers, and configuration servers do not support domain name access, so you cannot directly connect to them to perform operations.

You can purchase new cluster version instances or upgrade standard version (master/slave replication) instances to cluster version.

Scenarios

Large data volumes

Redis cluster version instances can be effectively scaled to accommodate different data volumes. Compared to the standard version, the cluster version can support larger capacities of 64, 128, and 256 GB to effectively meet your data scaling needs.

High QPS load

Standard version Redis instances cannot support high QPS and require you to use multi-node deployment to combat Redis' single thread performance bottleneck. Redis cluster version instances support 16, 32, 64, 128, and 256 GB configurations and 8-node and 16-node deployment modes. Cluster version instances can increase the QPS load by 8 or 16 times compared with standard version instances.

Throughput-intensive applications

Compared to standard version instances, Redis cluster version instances have looser restrictions on intranet throughput. This provides great support for businesses that read hotspot data and have high throughput requirements.

Apps insensitive to Redis protocol

Because of the multiple components, the cluster version architecture provides more limited support for Redis protocol than the standard version. For more information, see Supported Redis commands.

Specifications

Size (GB)	Max connections	Max intranet bandwidth (Mbps)	CPU core(s) (Relative)	Description
1	10000	10	1	Master-slave dual-node instance
2	10000	16	1	Master-slave dual-node instance
4	10000	24	1	Master-slave dual-node instance
8	10000	24	1	Master-slave dual-node instance
16	10000	32	1	Master-slave dual-node instance
32	10000	32	1	Master-slave dual-node instance
64	20000	48	1	Master-slave dual-node instance
128	160000	96	16	High-performance cluster instance
256	160000	96	16	High-performance cluster instance

QPS performance reference

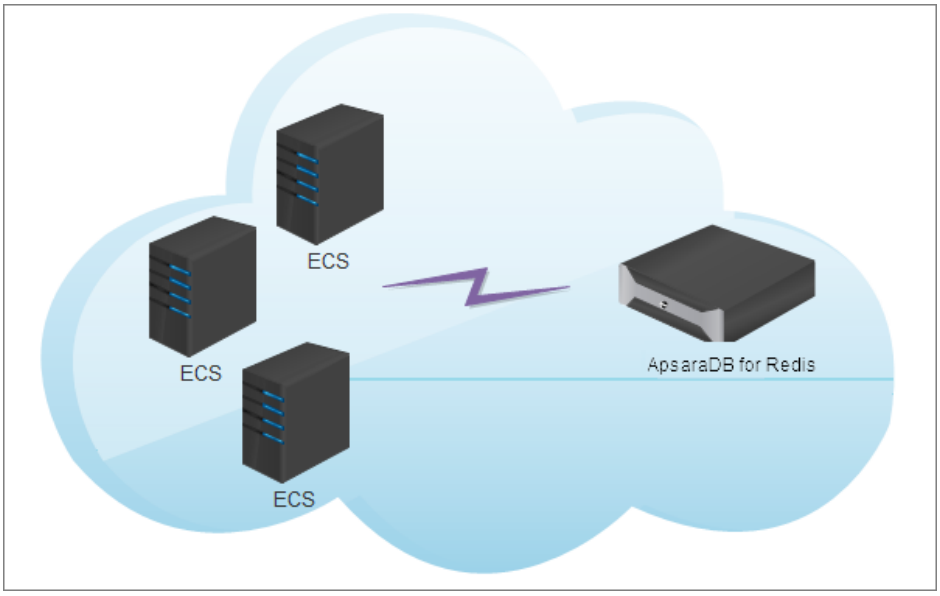
QPS performance

Size (GB)	Max	Max intranet	CPU core(s)	QPS reference

	connections	bandwidth (Mbps)		value
8	10000	24	1	80000

Test scenario

Network topology



Specification of cloud host

OS	CPU (number of cores)	Memory	Zone	Hosts
Ubuntu 14.04 64-bit	1	2048 MB	China South 1	3

Procedure

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

```
$ wget http://download.redis.io/releases/redis-2.8.19.tar.gz
$ tar xzf redis-2.8.19.tar.gz
$ cd redis-2.8.19
$ make
$ make install
```

Run the following command on the three ECS instances.

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

Summarize the testing data from the three ECS instances. The QPS is the total for the preceding three servers.

Flexible architecture

Single-node architecture

The single-node architecture is suitable for cache-only scenarios. It supports flexible configuration changes for single-node clusters and provides cost-effective performance that suits high QPS scenarios.

Dual-machine hot standby architecture

During system operations, data is synchronized between the master and slave nodes. If the master node is at fault, the system will automatically switch over to the slave node in a matter of seconds. The entire process is automatic without affecting your business. The master/slave architecture guarantees the high availability of system services.

Cluster architecture

Cluster instances adopt a distributed architecture, with each node working in master/slave mode. This provides automatic DR switchover and failover. You can choose from multiple cluster specifications based on your business needs, and the service allows you to scale your database performance without limit.

Data security

Backup and one-click recovery

The service automatically backs up data every day, providing strong disaster tolerance capabilities. Data can be recovered to any time point within seven days for free, to prevent incorrect data operations and minimize business loss.

Multi-layer cybersecurity protection

VPC provides network isolation protection at the TCP layer. The anti-DDoS feature can monitor and cleanse large-volume attacks in real time. Over 1,000 IP addresses can be added to the whitelist, to directly control risks at the access source.

Deep kernel optimization

Alibaba Cloud’ s expert team has performed in-depth kernel optimization on the Redis source code,

effectively preventing memory overflow and fixing security vulnerabilities to protect you throughout the service.

Elastic scaling

Data capacity scaling

ApsaraDB for Redis supports product configurations with multiple memory specifications. You can freely upgrade the memory specification to fit your business volume.

Performance scaling

The cluster architecture allows you to elastically scale data system storage space and throughput performance without restriction, breaking through the data volume and QPS performance limitations. The service can easily handle tens of thousands of read and write requests per second.

Business format scaling

The service supports a single-node cache architecture and dual-node storage architecture to suit different business scenarios. You can flexibly switch between the standard version and dual-node version.

Smart O&M

Monitoring platform

This platform provides instance information, such as CPU utilization, connection counts, and disk space usage, for real-time monitoring and alarms, so that you can check your instance status at any time.

Visual management platform

The management platform allows you to perform frequent and risky operations, such as instance cloning, backup, and data recovery, with a single click.

Visual DMS platform

The specialized DMS data management platform supports visual data management, improving your R&D and O&M efficiency in an all-round way.

Database kernel version management

This feature proactively performs upgrades and quickly repairs bugs, which frees you from daily version management. It also optimizes Redis parameter configurations and maximizes the utilization of system resources.

Superior performance

The cluster function supports ultra-high capacity and ultra-high performance. This service supports cluster functions and cluster instances of 128 GB or higher to meet large capacity and high performance requirements.

Provides up to 64 GB master-slave dual-node instances, to meet the capacity and performance requirements of average users.

Elastic resizing

One-click storage resizing: You can use the console to adjust the storage capacity of your instances as needed.

Online resizing with no service interruption: You can adjust the instance capacity online without suspending your services or affecting your business.

Data security

Persistent data storage: With memory + hard disk storage, the feature provides high-speed data read/write capability and meets the data persistence requirements.

Master-slave dual-backup for data: All data is backed up both on master and slave nodes.

Supports password authentication to ensure secure and reliable access.

High availability

Dual-copy and cluster version instances have a master node and a slave node. This prevents service interruption caused by SPOF.

Automatic detection and recovery of hardware failure: The feature can automatically detect hardware failures and fail over to the slave node, restoring service in a matter of seconds.

Instance-level resource isolation provides enhanced stability for individual services.

Second-level monitoring

Real-time second-level monitoring, and minute-level historical monitoring.

Provides monitoring information for data structures and interfaces, allowing you to monitor access conditions at a glance. This helps you fully understand the usage of ApsaraDB for Redis.

Easy to use

Out-of-the-box service: This product requires no setup or installation and can be used right after purchase for quick and convenient business deployment.

Compatible with open-source Redis: This product is compatible with Redis commands, and any Redis client can easily establish a connection with ApsaraDB for Redis to perform data operations.

Visual management and monitoring panel: The console provides monitoring statistics for multiple metrics and allows you to conveniently manage Redis instances.

Gaming industry applications

Game companies can use ApsaraDB for Redis as an important part of their deployment architecture.

Scenario 1: Using Redis for data storage

Game deployment architecture is relatively simple. With the main program deployed on ECS, all business data is stored in Redis as a persistent database. ApsaraDB for Redis supports persistence functions, with primary-standby dual-machine redundant data storage.

Scenario 2: Using Redis as a cache to accelerate application access

Using Redis as a cache layer will accelerate application access. Data is stored in a backend database (RDS).

Reliability is critical to Redis services. If a Redis service becomes unavailable, business access may overload the backend database. ApsaraDB for Redis provides a hot standby highly available architecture that guarantees extremely high service reliability. The master node provides external services. If this node fails, the system automatically sets up the standby node to take over the

services. The entire failover process is completely transparent to users.

E-commerce industry applications

In the e-commerce industry, Redis is extensively used with large volumes of data, mostly for item display, shopping recommendations, and other modules.

Scenario: Seckill-type shopping systems

During large-scale seckill promotions, a shopping system is overwhelmed by traffic, which far exceeds the R/W capability of common databases.

The persistence function supported by ApsaraDB for Redis allows you to directly use Redis as a database system.

Scenario: Inventory system with a counter

In such a system, the underlying architecture usually keeps actual data in RDS and count information in database fields.

In contrast, ApsaraDB for Redis reads the counts, while RDS stores the count information.

In this scenario, ApsaraDB for Redis is deployed on a physical machine, with an underlying architecture based on SSD high-performance storage that can provide high-level data reading capabilities.

Live video applications

Live video services are often reliant on Redis to store user data and relationship information.

Dual-machine hot standby guarantees high availability

ApsaraDB for Redis provides the hot-standby mode to maximize service availability.

Cluster version solves the performance bottleneck

ApsaraDB for Redis provides cluster version instances to break through the performance bottleneck of Redis single-thread mechanism. This approach can effectively cope with spikes in live video broadcast traffic and meet high performance requirements.

Easy resizing helps cope with business peaks

ApsaraDB for Redis can support one-click resizing. The entire upgrade process is fully transparent to

you and helps you easily cope with traffic bursts.

Term	Description
Redis	ApsaraDB for Redis is a high-performance Key-Value storage system (cache and store) released in compliance with the BSD open-source protocol.
Instance ID	An instance corresponds to a user space, and serves as the basic unit of using ApsaraDB for Redis. ApsaraDB for Redis limits connection quantities, bandwidth, CPU specifications, and other parameters based on the capacity specifications of individual instances. On the console, you can view the list of IDs of the instances you have purchased.
Master-Slave dual-node instance	This is an ApsaraDB for Redis instance that adopts a master-slave architecture. Master-slave dual-node instances provide limited capacity and performance.
High-performance cluster instance	This is an ApsaraDB for Redis instance that adopts a scalable cluster architecture. Cluster instances have better scalability and performance, but are functionally limited to a certain extent.
Connection address	This is the host address used to connect to ApsaraDB for Redis. It is displayed as a domain name, and can be found at **Instance Information** > **Connection Information**.
Connection password	This is the password used to connect to ApsaraDB for Redis. The password format is Instance ID:custom password. For example, if you set the password as 1234 when you make the purchase and the allocated instance ID is xxxx, then the connection password is xxxx:1234.
Eviction policy	This is consistent with the Redis eviction policy. For more information, see **the official document**.
DB	This is the abbreviation of database in Redis. By default, each ApsaraDB for Redis instance supports 256 databases numbered DB 0 to DB 255. By default, data is written to DB 0.

ApsaraDB for Redis was created when senior Alibaba Cloud experts performed in-depth kernel optimization on the Redis service, fixed security vulnerabilities, and improved the service’ s stability. At the same time, due to the constantly evolving needs of customers, the ApsaraDB for Redis kernel

version has been optimized to provide more product functions and support for native Redis commands.

This document will introduce the product features and functions provided by the latest ApsaraDB for Redis version. If you like the new features, you can upgrade to the latest kernel version with a single click on the console. Upgrading the kernel version will interrupt your connection for 30 seconds, so do it during low business hours and make sure your applications have reconnect mechanisms.

IP Whitelist

The standard version dual- and single-node configurations, and the cluster version al support custom whitelists.

GEO function

The current version of ApsaraDB for Redis is 2.8. To keep pace with the development of the Redis open-source community, ApsaraDB for Redis already fully supports the GEO feature that comes with Redis community version 3.2.

Config Get command

Restrictions on the use of the Config Get command are removed.

Support for LUA

Restrictions on the use of LUA scripts are removed. You can directly call this command in standard version dual- and single-node configurations.

The cluster version provides conditional support:

> All keys must be transmitted in the KEYS array. When the redis command in redis.call/pcall is called, the key location must be KEYS array, or an error will be reported.
>
> ```
> `"-ERR bad lua script for redis cluster, all the keys that the script uses must be passed using the KEYS array\r\n"`
> ```
>
> All keys must be in a single slot, or an error will be reported.
>
> "-ERR eval/evalsha command keys must in same slot\r\n"

Client list command

Restrictions on the use of the Client list command are removed. You can call this command in standard version dual- and single-node configurations, but the cluster version does not support the command for the moment.