

Tencent Cloud Distributed Cache (Redis OSS-Compatible) Troubleshooting Product Documentation



Copyright Notice

©2013–2026 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by the Tencent corporate group, including its parent, subsidiaries and affiliated companies, as the case may be. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

Troubleshooting

Connection Exception

- One-Click Connectivity Checker

- Private Network Unable to Connect Location Guide

Exception Analysis and Solution of Redisson Client Timeout Reconnection

Performance Troubleshooting and Fine-Tuning

- High CPU Utilization

- High Outbound Traffic

- High Memory Utilization

- High Number of Total Requests

- High Connection Utilization

- High Execution Latency

- Execution Error

Troubleshooting

Connection Exception

One-Click Connectivity Checker

Last updated: 2026-03-18 10:49:56

If the connection to Tencent Cloud Distributed Cache fails, it is recommended that you use the One-Click Connectivity Checker to locate the cause. This article introduces the potential causes and solutions for connection failures using the connectivity checker.

- When a private network connection fails, see [Private Network One-Click Connectivity Check](#) to quickly locate the issue. If the problem persists, further analyze the cause of the inability to connect by following the [Guide to Troubleshooting Private Network Connectivity Issues](#).
- For public network connection failures, see [Public Network One-Click Connectivity Check](#).

Private Network One-Click Connectivity Check

The Private Network One-Click Connectivity Check will inspect the following:

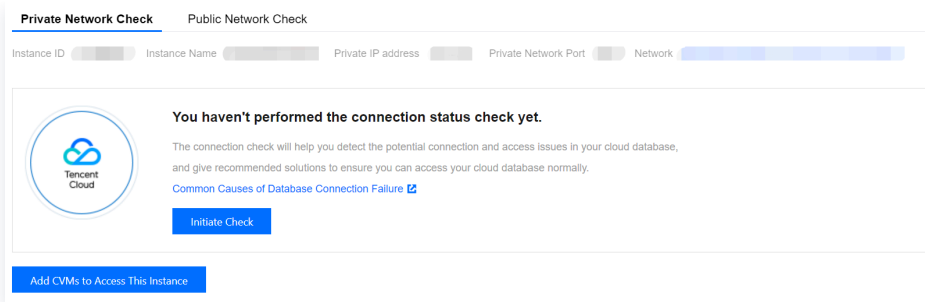
- Whether the Distributed Cache and CVM instances are normal.
- Whether CVM and Distributed Cache belong to the same VPC.
- Whether the inbound rules of the Distributed Cache security group allow the IP address of CVM, and whether the outbound rules of the CVM security group allow the private IP address of Distributed Cache.

Note:

In the **Classic Network**, the security group bound to the CVM **cannot filter** data packets to or from No SQL Database (Redis, Memcached). Use the One-Click Connectivity Checker for the Classic Network security group's configuration. If the check results are abnormal, please perform manual inspection and verification.

Directions

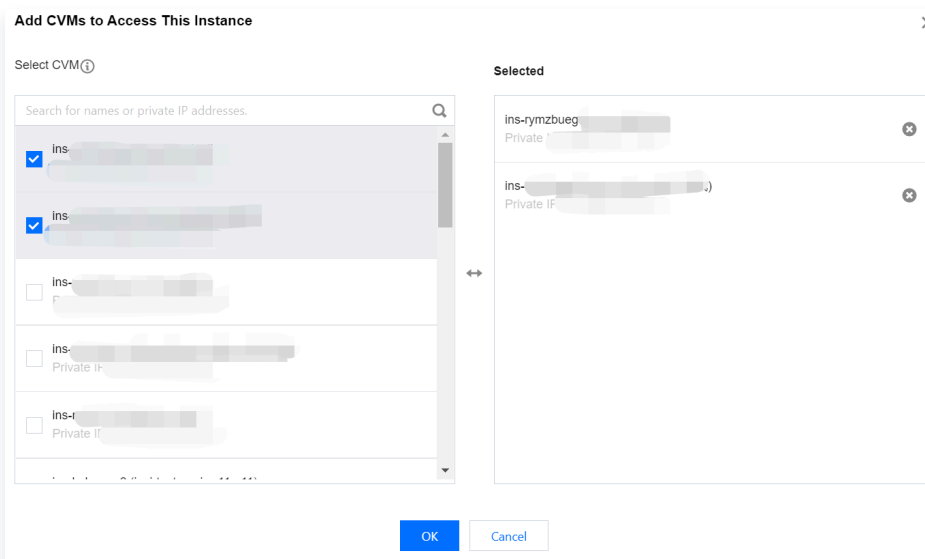
1. Log in to the [Tencent Cloud Distributed Cache console](#).
2. Select the instance you want to troubleshoot, click **Instance ID** to open the instance details page.
3. On the instance details page, go to the **Connection Check > Private Network Check** page.



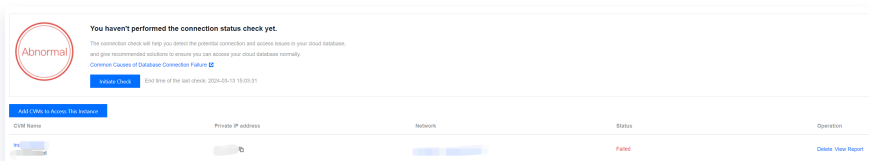
4. Click **Add CVMs to Access This Instance**, in the following displayed window, all CVMs in the same region as the instance will automatically be displayed. Select the CVM used to access this Redis instance and click **OK**.

Note:

By default, only cloud services in the same region are provided. Cloud services across regions do not support configuration checks.



5. Click **Initiate Check**, wait for the check to complete. Then a check report will be generated, as shown in the following figure.



6. In the **Operation** column, click **Peek Report**. Based on the check report, locate the issues and make adjustments according to the recommended solutions before attempting to reconnect to Distributed Cache.

Check Report Details ✕

CVM Name: ins-n1gmmhfo

Check Item	Status	Impact	Processing Suggestion
Redis Instance status	Passed	--	
CVM Instance Status	Passed	--	
CVM and Redis are in the same VPC	Abnormal	CVM cannot access the Redis instance.	It has been detected that your server and Redis are not within the same VPC IP range. CVM needs to be in the same VPC of the same region as Redis. Please refer here for processing.
CVM Security Group Policy	Passed	--	
Redis security group policy	Passed	--	

Total Items: 5

Exception Analysis and Measures

Check Items	Status	Impact	Suggestion	Solution
Redis instance status	Abnormal	Redis instance unreachable	<p>Isolated: Your Redis instance has been isolated. If you need to continue using this Redis instance, please go to the Redis Recycle Bin to restore the isolated instance.</p> <p>It is detected that your instance has been isolated. If you need to continue using this instance, please go to the recycle bin to restore the isolated instance.</p>	For specific operations, see Restoring Isolated Instance .
CVM instance status	Abnormal	CVM instance unreachable	<ul style="list-style-type: none"> ● Isolated: Your CVM instance has been isolated. If you need to continue using this CVM instance, please go to the CVM Recycle Bin to restore the instance. ● Shutdown: Your CVM instance has been shut down. If you need to continue using this CVM instance, please go to 	<ul style="list-style-type: none"> ● Isolated: For specific operations, see Instance Repossession or Recovering. ● Shutdown: For specific operations, see Starting Up Instances.

			the CVM console to start the CVM instance.	
CVM and Redis are in the same VPC	Abnormal	Redis instance unreachable from the CVM	Your server and the Redis instance are detected to be in different VPC IP ranges. The CVM instance needs to be in the same region and the same VPC as the Redis instance.	<ul style="list-style-type: none"> To modify the network information of Distributed Cache, see Configuring Network. To modify the network information of CVM, see Switching to VPC.
Redis security group policy	Abnormal	Redis instance unreachable from the CVM	It has been detected that the inbound rules of the security group bound with your Redis have not allowed access to the CVM's internal network IP address and ports.	<ol style="list-style-type: none"> Log in to the CVM console, in the instance list under the Master IPv4 Address column, verify the internal network IP address of the CVM connecting to Distributed Cache. For specific operations, see Viewing Instance Information. Please go to the Tencent Cloud Distributed Cache console, under the Security Group tab to the Rule Preview section, check Inbound Rules. For specific operations, see Configuring Security Group.
CVM security group policy	Abnormal	Redis instance unreachable from the CVM	It has been detected that the outbound rules of the security group affiliated with your CVM have not allowed access to Redis's internal network IP address and ports.	<ol style="list-style-type: none"> Log in to the CVM console, on the Security Group management page, verify the Outbound Rules. For specific

operations, see [Viewing Security Groups](#).

2. Modify the outbound rules of the security group to allow access to Redis's internal network IP address and ports. For specific operations, see [Modifying Security Group Rules](#).

Public Network One-Click Connectivity Check

The Public Network One-Click Connectivity Checker will inspect the following:

- Whether the status of the Distributed Cache instance is normal.
- Whether the instance is enabled with the public network access feature.
- Whether the Distributed Cache security group allows access to external network addresses and ports.

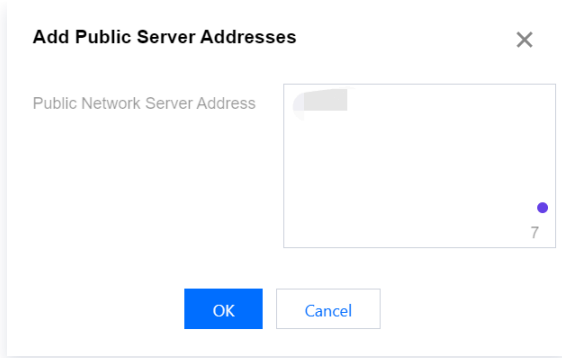
Directions

1. Log in to the [Tencent Cloud Distributed Cache console](#).
2. Select the instance you want to troubleshoot, click on the **Instance ID** to enter the instance details page.
3. On the instance details page, go to the **Connection Check > Public Network Check** page.
4. Click **Add the Public server Addresses accessing this instance**. In the **Public Server Addresses** input field, enter the required external network server address according to the instructions, and then click **OK**.

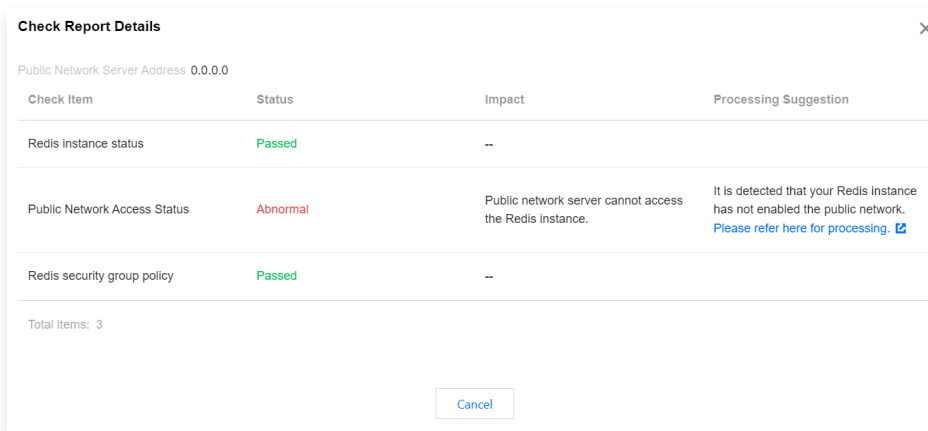
Note:

The required format for the added external network server are as follows:

1. IP address format: xx.xx.xx.xx. Other formats will result in an error prompt.
2. Multiple IP address separators: Please use line break, comma, semicolon, or space.
3. Quantity limit: Up to 20 external network server addresses can be added for each check.



5. Click **Initiate Check**, and wait for the check to complete. Then a check report will be generated.
6. In the **Operation** column, click **Peek Report**. Based on the check report, locate the issues and make adjustment according to the recommended solutions before attempting to reconnect to Redis.



Exception Analysis and Measures

Check Items	Status	Impact	Suggestion	Solution
Redis instance status	Abnormal	Redis instance unreachable	Isolated: Your Redis instance has been isolated. If you need to continue using this Redis instance, please go to the Redis Recycle Bin to restore the isolated instance.	For specific operations, see Restoring Isolated Instance .
Public network access status	Abnormal	Public network servers are unable to access the Redis	It has been detected that your Redis instance's public network is not enabled. To enable it, see Configuring Public Network Address .	To enable the public network, please refer to Configuring Public Network Address . Note:

		instance.		<p>Currently, only the Chengdu, Beijing, Shanghai, and Guangzhou regions support public network addresses.</p>
Redis security group policy	Abnormal	Public network servers are unable to access the Redis instance.	<p>It has been detected that the inbound rules of the security group bound to your Redis instance have not allowed access to 1.1.1.1 and TCP protocol port 6379. If you have not configured it purposely, it may prevent your public network servers from accessing the Redis instance properly.</p>	<p>For detailed operations, see Configuring Security Group.</p> <p>Note: After public network access is enabled, it will be governed by the security group network access policy. Please configure the source information for database access in the security group's inbound rules and enable the protocol ports (both private and public network ports need to be enabled, and the default private network port is 6379).</p>

Private Network Unable to Connect Location Guide

Last updated: 2026-03-17 18:23:48

Issue Description

- Issue 1: Use the CVM to connect to a Tencent Cloud Distributed Cache instance at the private network address automatically assigned by the system as instructed in [Connecting to Tencent Cloud Distributed Cache Instance](#). However, the connection fails.
- Issue 2: Log in to the [Tencent Cloud Distributed Cache console](#), go to the instance list, click **Log In** in the **Operation** column of the target instance to redirect to DMC, connect to the instance. However, the connection fails, as shown in the following figure.

Possible Causes

- For database connection failures for the first time, the possible causes are as follows:
 - Port error.
 - Network configuration error or incorrect security group configuration.
 - Password error.
- For sudden connection failures during instance running, the possible causes are as follows:
 - The maximum number of connections has been reached.
 - Memory or shards have been used up.
 - A high-availability (HA) switchover occurs, the database service becomes unavailable, a read-only replica switchover occurs, the read-only replica service becomes unavailable, or more.
- For client errors, the possible causes are as follows:
 - Connection pool parameter setting is not proper.
 - Connections leak.

Troubleshooting

Step 1. Run `telnet` to Check Whether the Distributed Cache Port Can Be Accessed Normally

For most connection failure issues, run `telnet` in the command line tool to narrow down the cause of the error:

```
[root@VM-4-10-centos ~]# telnet 10.x.x.34 6379
Trying 10.x.x.34...
```

```
Connected to 10.x.x.34.  
Escape character is '^]'.
```

As shown above, if there is successful connection prompt, the port of the Distributed Cache instance can be accessed normally. If there is an exception, go to [Step 2](#) to troubleshoot network issues.

Step 2. Check Whether It Is Caused by Network Configuration

To connect over the private network, the CVM and Tencent Cloud Distributed Cache must be under the same account and in the same VPC, or both in the classic network. The connection will fail in the following conditions:

- If the CVM instance is in a VPC, while the Redis instance in the classic network, you are advised to switch the network type of the Distributed Cache instance from the classic network to VPC.
- If the Distributed Cache instance is in a VPC, while the CVM instance in the classic network, you are advised to switch the network type of the CVM instance from classic network to VPC. For more information, see [Switching to VPC](#).
- If the CVM and Distributed Cache are in different VPCs in the same region, you are advised to migrate the Redis instance to the VPC of the CVM instance.
- If the CVM and Distributed Cache are in different VPCs in different regions, you are advised to create a [CCN](#) between the two VPCs.
- If the CVM and Distributed Cache are in different VPCs under different accounts, you are advised to create a [CCN](#) between the two VPCs.

Step 3. Check Whether It Is a Security Group Issue

The CVM instance cannot connect to the Distributed Cache if their security group configuration is incorrect.

Incorrect CVM Security Group Configuration


To use the CVM to connect the Distributed Cache, you need to configure an outbound rule in the security group of the CVM instance. **If the target of the outbound rule is not "0.0.0.0/0" and the protocol port is not "ALL"**, the IP address and port of the Distributed Cache instance need to be added to the rule.

1. Go to the [Security Group](#) page in the CVM console and click the name of the CVM-bound security group to open its details page.
2. On the **Outbound rule** tab, click **Add Rule**.
 - **Type:** Select **Custom**.
 - **Target:** Enter the IP address or IP range of your Distributed Cache.
 - **Protocol Port:** Enter the private network port of the Distributed Cache.
 - **Policy:** Select **Allow**.

Incorrect Distributed Cache Security Group Configuration

To use the CVM to connect the Distributed Cache, you need to configure an inbound rule in the security group of the Redis instance. **If the source of the inbound rule is not "0.0.0.0/0" and the protocol port is not "ALL",** the IP address and port of the CVM instance need to be added to the rule.

1. Go to the [Security Group](#) page in the CVM console and click the name of the Distributed Cache security group to open its details page.
2. On the **Inbound rule** tab, click **Add Rule**.
Note that you also need to enable the IP address and port of the Redis in the inbound rule.
3. Enter the IP address (or IP range) and port information (Distributed Cache private network port) you wish to allow connections from, and select Allow.
 - **Type:** Select **Custom**.
 - **Source:** Enter the IP address or IP range of your CVM.
 - **Protocol Port:** Enter the private network port of the Distributed Cache instance.
 - **Policy:** Select **Allow**.

 **Note:**

- The default private network port of Redis is 6379, and the port can be customized. If the default port is changed, the new port need be enabled in the inbound rule of the Redis security group.
- If the default port 6379 of the Redis instance is used, it need to be enabled in the inbound rule of the Redis security group.

Step 4. Check Whether the Issue Is Caused by the Password

Run the `info` command. If the following information is displayed, the password of the Distributed Cache is correct.

```
[root@SNG-Qcloud /data/home/rickyu]# redis-cli -h 10.x.x.34 -p 6379 -a
password
10.x.x.2:6379> info cpu
# CPU
used_cpu_sys:1623.176000
used_cpu_user:4649.572000
used_cpu_sys_children:0.000000
used_cpu_user_children:0.000000
```

If `NOAUTH Authentication required.` is displayed, the password is incorrect.

```
10.x.x.31:6379> info memory
```

```
NOAUTH Authentication required.  
10.x.x.31:6379>
```

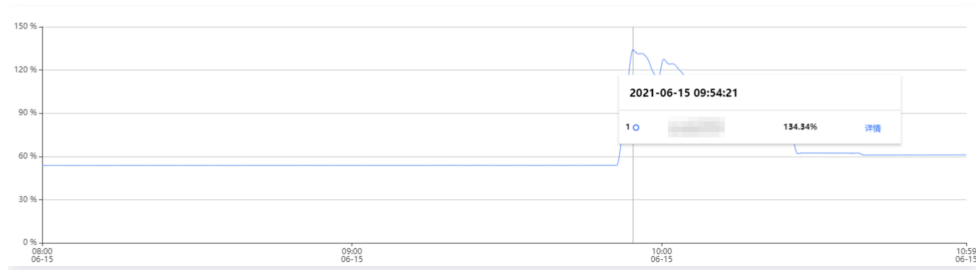
Log in to the [Tencent Cloud Distributed Cache console](#) and click an instance ID in the instance list to open the instance details page, where you can reset the password. For more information, see [Managing Account](#).

Step 5. Check Whether the Memory or Shards Are Full

If the following error message is displayed for the business:

```
"-readonly You can't write against a read only slave.\r\n"
```

Log in to the [Tencent Cloud Distributed Cache console](#), find the target instance in the instance list, click the instance ID to enter the **System Monitoring** page, and select **Memory Utilization** in the **Metric** drop-down list to view the memory usage of the instance.



If memory is full, writes will fail. In this case, perform the following operations:

- Expand the capacity immediately as instructed in [Changing Instance Specification](#).
- Modify the database eviction policy as instructed in [Managing Instance Parameter](#). In **Parameter Settings**, set the `maxmemory-policy` parameter to `allkeys-lru` or `volatile-lru`. For details, see [Setting Instance Parameters](#).

Note:

Instance data may be lost if the `allkeys-lru` eviction policy is adopted. Assess the impact before doing so.

Step 6. Check the Connection Quota

The **Connection Utilization** metric refers to the ratio of the number of TCP connections from the client to the instance to the maximum number of connections to the instance. If this metric is continuously high, the current database connection quota is insufficient, and the maximum number of connections needs to be adjusted.

Issue

The error message is as follows:

```
ERR max number of clients reached
```

Solution

1. Log in to the [Tencent Cloud Distributed Cache console](#). Above the instance list on the right, select the region. In the instance list, find the target instance. Click the instance ID in blue to enter the **Instance Details** page, click the **System Monitoring** tab, and then select the **Monitoring Metrics** tab to view the monitoring data. In the **View** drop-down list, select **Instance Monitoring** and **Connection Utilization** as the metric, and check whether it is continuously high in the monitoring view.
2. If the connection usage is continuously high, adjust the maximum number of connections and modify the specifications for the number of connections. For details, see [Adjusting the Number of Connections](#).

Step 7. Check Whether HA Switchover, Unavailable Database Service, Read-only Replica Switchover, or Unavailable Read-only Replica Service Occurs

If you find abnormal connections or a large number of access errors and slow queries at a certain point in time, and TCOP event alarms for those abnormal events are received, [Contact Us](#) for help.

For the configuration method of TCOP event alarms, see [Creating Event Rule](#).

Step 8. Confirm Whether the Configuration of the Jedis Connection Pool Is Correct If You Use It

Issue

If the number of available connections in the connection pool is used up and the old connections are not released in time, the newly created connection will fail, and the following error message will be displayed on the client.

```
JedisConnectionException: Could not get a resource from the pool
```

Solution

1. Use the following command on the client to confirm the number of connections currently accessing port 6379 of the instance. If this number is close to the `maxTotal` value configured in the connection pool, a connection failure will occur.

```
netstat -an | grep 6379 | grep ESTABLISHED | wc -l
```
2. See [Java Connection Sample](#), and check whether to call `jedis.close()` to release old connections to avoid connection leaks.
3. If all old connections have been released and the concurrent business volume is large, the `maxTotal` parameter value needs to be increased.

 **Note:**

The maxTotal value of each client connection pool * The number of clients = The maximum number of connections for TencentDB for Redis®

References

Viewing the Network Type and VPC Information

To enable connection between CVM and Distributed Cache over private network, they must be under the same account and in the same VPC, or both in the classic network.

Note:

- If the **Network** fields in the instance lists both show **Classic Network** or **VPC**, the networks of the CVM and Redis are of the same type.
- If the **Network** fields in the instance lists both show the same **VPC** (in the same region), it means that the CVM and TencentDB for Redis® instances are in the same VPC.

Viewing the CVM Network Type

Log in to the [CVM console](#) and view the **Network** in the **Instance List**.

ID Name	Monitoring	Status	Availability zone	Instance type	Network configuration	Primary IP	Primary IPv6	Instance billing mode	Network billing mode	Project	Operation
...	...	Running	Guangzhou Zone G	Compatible C5	Classic Network	...	-	Monthly subscription	Prepaid by bandwidth	Default Project	Log In, Renew, More

Viewing the Distributed Cache Network Type

Log in to the [Tencent Cloud Distributed Cache console](#) and view the **Network** in the **Instance List**.

Instance ID Name	Monitoring/Status/Tag	AZ	Network Type	Billing Mode	Architecture	Instance Edition	Used/Total	Creation Time	Project	Tag	Operation
...	Running	Guangzhou Zone G	Classic Network	Pay as You Go	Redis 6.0 Standard Architecture	Memory Edition	10GB/10GB(100.0%)	2024-03-08 15:16:07	Default Project		Log In, Configure, More

Enabling Public Network Access

Tencent Cloud Distributed Cache now allows you to manually enable public network access in the console, to access Redis instance from the public network. For detailed directions, see [Configuring the Public Network Address](#).

Exception Analysis and Solution of Redisson Client Timeout Reconnection

Last updated: 2026-03-17 18:23:49

This article aims to briefly discuss the business blocking issues encountered by the Redisson client during network fluctuations and service failover, and proposes a simple and effective business layer solution to inspire solutions to similar problems.

Overview

When the Redisson client is used connect to a Tencent Cloud Distributed Cache instance, the VPC gateway fails and the failure lasts nearly 10 seconds. To restore the network connection, the faulty network device is isolated and the network link is recovered. Although the number of connections is restored to the normal business level, error reports for businesses affected by the network failure are still reported continuously. Only after the client is restarted and the system re-establishes the connection with the Redis database, business requests return to normal.

Exception Analysis

Based on the process of the exception, the scenario is replicated to access error logs and deeply analyze the internal working mechanism of Redisson connections, to locate potential issues.

Replicating the Issue

For the complex scenario of continuous network jitter, to facilitate testing and clear presentation, iptables rules are configured on the CVM to block access to Redis services, thereby simulating the effect of network jitter. Then, the block is lifted, the network link is restored, and error logs from business requests are collected.

1. To ensure that code clarity and ease of understanding, design a simple Redisson client example to connect to a Tencent Cloud Distributed Cache server, and execute a simple while loop upon successful connection to simulate a continuous task.

Note:

- In the following code, set the Redis server's address, port, password, and timeout (3000ms) information in the `config-file.yaml` configuration file. Based on the configuration information, establish a connection to Redis.
- This example is based on Redisson 3.23.4, and different versions of Redisson may have some variations. You are advised to consult or inquire with Redisson's official for analysis.

```
package org.example;
import org.redisson.Redisson;
import org.redisson.api.RAtomicLong;
import org.redisson.api.RedissonClient;
import org.redisson.config.Config;
import org.redisson.connection.balancer.RoundRobinLoadBalancer;
import java.io.File;
import java.io.IOException;

public class Main {
    public static void main(String[] args) throws IOException {
        boolean connected = false;
        RedissonClient redisson = null;
        while (!connected) {
            try {
                Config config = Config.fromYAML(new
File("/data/home/sharmaxia/redissontest/src/main/resources/config-
file.yaml"));
                redisson = Redisson.create(config);
                connected = true;
            } catch (Exception e) {
                e.printStackTrace();
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException ex) {
                    Thread.currentThread().interrupt(); // Actively
interrupt the current thread to retrieve a connection from the
connection pool again
                }
            }
        }
        int i = 0;
        while (i++ < 1000000) {
            try {
                RAtomicLong atomicLong =
redisson.getAtomicLong(Integer.toString(i));
                atomicLong.getAndDecrement();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

```
        try {
            Thread.sleep(1000);
        } catch (InterruptedException ex) {
            Thread.currentThread().interrupt(); // Actively
interrupt the current thread to retrieve a connection from the
connection pool again
        }
        i--;
    }
}
redisson.shutdown();
}
}

// Import Redisson-related packages
import org.redisson.Redisson;
import org.redisson.api.RAtomicLong;
import org.redisson.api.RedissonClient;
import org.redisson.config.Config;
import org.redisson.connection.balancer.RoundRobinLoadBalancer;

// Import Java IO-related packages
import java.io.File;
import java.io.IOException;

public class Main {
    public static void main(String[] args) throws IOException {
        // Read Redisson's configuration information from a YAML file
        Config config = Config.fromYAML(new
File("/data/home/test*/redisson-test/src/main/resources/config-
file.yaml"));

        RedissonClient redisson = Redisson.create(config);

        // Initialize a counter variable
        int i = 0;

        // Loop 1,000,000 times
        while(i++ < 1000000){
```

```
// Access an atomic long object, where the key is the
current counter's value
RAtomicLong atomicLong =
redisson.getAtomicLong(Integer.toString(i));

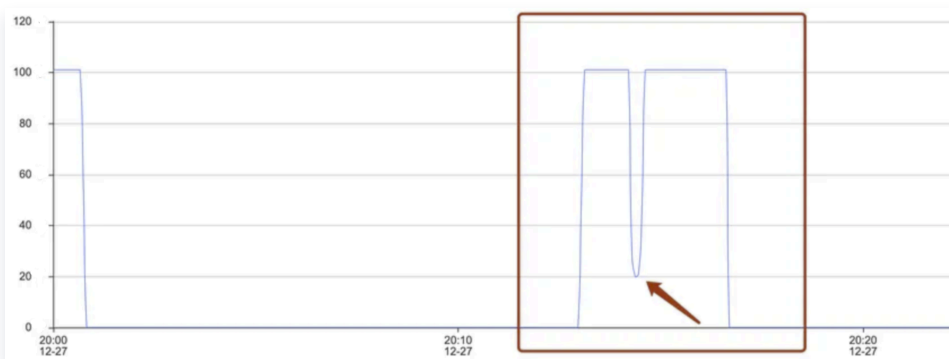
// Perform an atomic decrement by 1 on the value of the
atomic long object
atomicLong.getAndDecrement();
}

// Close the Redisson client
redisson.shutdown();
}
}
```

2. After the client starts normally and runs for a short period, modify the iptables configuration to block connections to Redis.

```
sudo iptables -A INPUT -s 10.0.16.6 -p tcp --sport 6379 -m conntrack -
-ctstate NEW,ESTABLISHED -j DROP
```

3. On the [Tencent Cloud Distributed Cache console](#), in the **System Monitoring** page, check the **Connections** metric. This metric shows a straight-line downward trend, as shown in the following figure.



4. Execute the following command to modify the iptables configuration, remove the connection block, and restore the network.

```
sudo iptables -D INPUT -s 10.0.16.6 -p tcp --sport 6379 -m conntrack -
-ctstate NEW,ESTABLISHED -j DROP
```



```
    }

    log.debug("channel: {} closed due to PING response timeout set
in {} ms", ctx.channel(), config.getPingConnectionInterval());
    ctx.channel().close();

connection.getRedisClient().getConfig().getFailedNodeDetector().onPing
Failed();
    } else {

connection.getRedisClient().getConfig().getFailedNodeDetector().onPing
Successful();
        sendPing(ctx);
    }
} else {

connection.getRedisClient().getConfig().getFailedNodeDetector().onPing
Successful();
    sendPing(ctx);
}

Throwable cause = cause(future);
if (!(cause instanceof RedisRetryException)) {
    if (!future.isCancelled()) {
        log.error("Unable to send PING command over channel: {}",
ctx.channel(), cause);
    }
    log.debug("channel: {} closed due to PING response timeout set
in {} ms", ctx.channel(), config.getPingConnectionInterval());
    ctx.channel().close();

connection.getRedisClient().getConfig().getFailedNodeDetector().onPing
Failed();
    } else {

connection.getRedisClient().getConfig().getFailedNodeDetector().onPing
Successful();
        sendPing(ctx);
    }
} else {

connection.getRedisClient().getConfig().getFailedNodeDetector().onPing
```

```
Successful();
    sendPing(ctx);
}
```

2. Further analyze the Connection source `RedisConnection connection = RedisConnection.getFrom(ctx.channel());` and check the constructor for creating the connection `RedisConnection()`, as shown in the following figure. This function updates the channel attributes and records the last use time of the connection but does not switch to a new channel for connection attempts.

```
public <C> RedisConnection(RedisClient redisClient, Channel channel,
CompletableFuture<C> connectionPromise) {
    this.redisClient = redisClient;
    this.connectionPromise = connectionPromise;

    updateChannel(channel);
    lastUsageTime = System.nanoTime();

    LOG.debug("Connection created {}", redisClient);
}

this.redisClient = redisClient;
this.connectionPromise = connectionPromise;

updateChannel(channel);
lastUsageTime = System.nanoTime();

LOG.debug("Connection created {}", redisClient);
}

// updateChannel updates the attributes of the Channel
public void updateChannel(Channel channel) {
    if (channel == null) {
        throw new NullPointerException();
    }
    this.channel = channel;
    channel.attr(CONNECTION).set(this);
}

if (channel == null) {
    throw new NullPointerException();
}
this.channel = channel;
```

```
channel.attr(CONNECTION).set(this);  
}
```

3. If cluster information is updated, the following error message will be displayed.

```
ERROR org.redisson.cluster.ClusterConnectionManager - Can't update  
cluster stateorg.redisson.client.RedisTimeoutException: Command  
execution timeout for command: (CLUSTER NODES), params: [], Redis  
client: [addr=redis://10.0.16.7:6379] at  
org.redisson.client.RedisConnection.lambda$async$0 (RedisConnection.java:  
256) at  
io.netty.util.HashedWheelTimer$HashedWheelTimeout.run (HashedWheelTimer.  
java:715) at  
io.netty.util.concurrent.ImmediateExecutor.execute (ImmediateExecutor.j  
ava:34) at  
io.netty.util.HashedWheelTimer$HashedWheelTimeout.expire (HashedWheelTi  
mer.java:703) at  
io.netty.util.HashedWheelTimer$HashedWheelBucket.expireTimeouts (Hashed  
WheelTimer.java:790) at  
io.netty.util.HashedWheelTimer$Worker.run (HashedWheelTimer.java:503)  
at  
io.netty.util.concurrent.FastThreadLocalRunnable.run (FastThreadLocalRu  
nnable.java:30) at  
java.base/java.lang.Thread.run (Thread.java:829)
```

4. In `ClusterConnectionManager.java`, you can locate the error message fragment related to updating the cluster. It shows that after the `Can't update cluster state` exception occurs, only the change in cluster status is checked, without retrieving a new connection from the connection pool for operation, and the operation of returning is directly performed.

```
private void checkClusterState(ClusterServersConfig cfg,  
Iterator<RedisURI> iterator, AtomicReference<Throwable> lastException)  
{  
    if (!iterator.hasNext()) {  
        if (lastException.get() != null) {  
            log.error("Can't update cluster state",  
lastException.get());  
        }  
        if (!iterator.hasNext()) {
```

```
        if (lastException.get() != null) {
            log.error("Can't update cluster state",
lastException.get());
        }
        // Check for changes in cluster status
        scheduleClusterChangeCheck(cfg);
        return;
    }
    .....
}

        scheduleClusterChangeCheck(cfg);
        return;
    }
    .....
}
```

Solutions

Based on the above issue, adjust the business-side code. In case of an exceptional connection, actively interrupt the current connection thread, perform the Channel and connection configuration again, and attempt a reconnection.

Code Adjustment

Based on the code, when the issue was reproduced, optimize the following code to include operations for exceptional reconnection.

```
package org.example;
import org.redisson.Redisson;
import org.redisson.api.RAtomicLong;
import org.redisson.api.RedissonClient;
import org.redisson.config.Config;
import org.redisson.connection.balancer.RoundRobinLoadBalancer;
import java.io.File;
import java.io.IOException;

public class Main {
    public static void main(String[] args) throws IOException {
        boolean connected = false;
        RedissonClient redisson = null;
```

```
while (!connected) {
    try {
        Config config = Config.fromYAML(new
File("/data/home/sharmaxia/redissontest/src/main/resources/config-
file.yaml"));

        redisson = Redisson.create(config);
        connected = true;
    } catch (Exception e) {
        e.printStackTrace();
        try {
            Thread.sleep(1000);
        } catch (InterruptedException ex) {
            Thread.currentThread().interrupt(); // Actively
interrupt the current thread to retrieve a connection from the
connection pool again
        }
    }
}

int i = 0;
while (i++ < 1000000) {
    try {
        RAtomicLong atomicLong =
redisson.getAtomicLong(Integer.toString(i));
        atomicLong.getAndDecrement();
    } catch (Exception e) {
        e.printStackTrace();
        try {
            Thread.sleep(1000);
        } catch (InterruptedException ex) {
            Thread.currentThread().interrupt(); // Actively
interrupt the current thread to retrieve a connection from the
connection pool again
        }
        i--;
    }
}

redisson.shutdown();
}

import org.redisson.Redisson;
```

```
import org.redisson.api.RAtomicLong;
import org.redisson.api.RedissonClient;
import org.redisson.config.Config;
import org.redisson.connection.balancer.RoundRobinLoadBalancer;
import java.io.File;
import java.io.IOException;

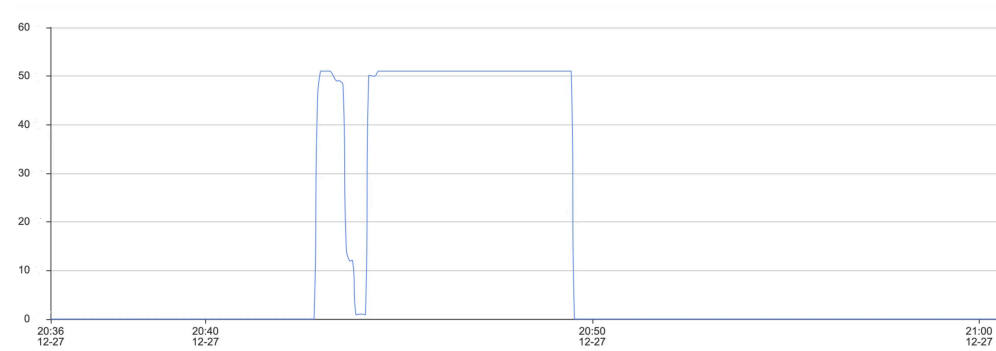
public class Main {
    public static void main(String[] args) throws IOException {
        boolean connected = false;
        RedissonClient redisson = null;
        while (!connected) {
            try {
                Config config = Config.fromYAML(new
File("/data/home/sharmaxia/redissontest/src/main/resources/config-
file.yaml"));
                redisson = Redisson.create(config);
                connected = true;
            } catch (Exception e) {
                e.printStackTrace();
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException ex) {
                    Thread.currentThread().interrupt(); // Actively
interrupt the current thread to re-obtain a connection from the
connection pool
                }
            }
        }
        int i = 0;
        while (i++ < 1000000) {
            try {
                RAtomicLong atomicLong =
redisson.getAtomicLong(Integer.toString(i));
                atomicLong.getAndDecrement();
            } catch (Exception e) {
                e.printStackTrace();
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException ex) {
```

```
        Thread.currentThread().interrupt(); // Actively
interrupt the current thread to re-obtain a connection from the
connection pool
    }
    i--;
}
}
redisson.shutdown();
}
}
```

Result Verification

Check the number of connections and the total number of requests and ensure that the business has returned to normal, to verify that adjustments to the code have resolved the issue of business disruptions caused by Redisson client timeout reconnection exceptions.

Number of Connections



Total Requests



Performance Troubleshooting and Fine-Tuning

High CPU Utilization

Last updated: 2024-11-05 10:22:22

Increasing CPU utilization will affect the throughput of the entire instance cluster, and may even cause application blocking and timeout interruptions. When the average CPU utilization exceeds 60% or the average peak CPU utilization stays higher than 90% for more than 5 minutes, you need to check the specific cause in time and solve the problem quickly to ensure business stability and availability.

Symptom

- Symptom 1: You received an alarm about high CPU utilization.
- Symptom 2: The value of the CPU utilization metric was high.
- Symptom 3: The overall throughput became lower and the response slower.

Troubleshooting

No.	Possible Causes	Cause Analysis	Troubleshooting Methods	Solution
1	Non-persistent connections are established frequently.	A large number of resources of the instance are consumed in processing frequent non-persistent connections, resulting in high CPU utilization and a lot of connections. However, the QPS (cluster accesses per second) did not meet expectations.	With DBbrain's Performance Optimization feature, you can analyze the real-time session statistics view and data of the database instance to check whether there is a sudden increase in the number of connections. For directions, see Real-Time Session .	Change the non-persistent connection to a persistent connection, for example, for example, by using the JedisPool connection pool. For specific code samples, see Jedis Client .

<p>2</p>	<p>There are commands with high time complexity, such as <code>sort</code>, <code>union</code>, and <code>zunionstore</code>.</p>	<p>As Redis executes commands in a single thread, the execution of complex commands may block other commands. The higher the time complexity of the command, the more resources will be consumed during execution and the CPU utilization will increase.</p>	<p>Executing high-complexity commands is usually time-consuming and will generate slow logs accordingly. You can use DBbrain's Performance Optimization feature to perform Slow Log Analysis, and check whether there are high-complexity commands in the slow log information list. For directions, see Slow Log Analysis.</p>	<p>Avoid using complex commands to get large amounts of data. Try to just use them on a limited amount of data instead, so Redis can process it quickly and return the results.</p>
<p>3</p>	<p>The read and write load is too high due to big keys or a large number of accesses to hot keys.</p>	<p>Hot keys refer to those key values that are accessed very frequently over a period of time, such as trending news, popular live streaming, and flash sales. In these cases, a large amount of access traffic is concentrated in one instance, reaching the processing capacity of a single instance and causing an increase in CPU utilization.</p>	<p>With DBbrain's Performance Optimization feature, you can perform Hot Key Analysis to quickly find hot keys with high access frequency. For directions, see Hot Key Analysis.</p>	<p>You can split hot keys of complex data structures into several new keys and distribute them across Redis nodes to reduce the pressure. For example, if a two-level hash hot key has a lot of hash elements, you can split it.</p>
<p>4</p>		<p>A big key is one that has big</p>	<p>With DBbrain's Performance</p>	<p>If the value of a big key is too large, you can</p>

		value and takes up a large space. When read or deletion operations related to big keys are performed, bandwidth and CPU will be seriously occupied.	Optimization feature, you can perform Big Key Analysis and monitor and analyze the memory usage of big keys in the database. For directions, see Memory Analysis .	split the big key into multiple key-values so that multiple Redis nodes will share the pressure. If there are too many keys, you can store them in a hash structure.
5		The read load is too high and the resource limit has been reached.	With DBbrain's Performance Optimization feature, you can check the Performance Trends view of the database instance, analyze QPS and read/write request metrics, and check whether the high CPU utilization is caused by excessive read load or write load. For directions, see Performance Trends .	If the read load is too high , you can share the read load by increasing the number of replicas as instructed in Changing Instance Specification . You need to enable read-only replicas and transfer the read requests of the current instance to the read-only replica node, so as to realize the elastic expansion of the read capability and improve the read and write performance. For directions, see Enabling/Disabling Read/Write Separation .
6		The write load is too large, and the memory specification is insufficient.		If the write load is too high , you can share the write load by increasing the number of shards as instructed in Changing Instance Specification . If the instance is on a standard architecture, you need to upgrade the standard architecture to a cluster architecture to improve

				<p>CPU processing capabilities. For directions, see Upgrading Instance Architecture. You need to check compatibility before upgrading to the cluster architecture. For more information, see Check on Migration from Standard Architecture to Cluster Architecture.</p>
7	<p>A large number of keys expire at the same time.</p>	<p>The key expiration time is set at the same time point. When the expiration time point is reached, Redis will take up a lot of CPU resources to process the elimination thread of those expired keys, causing a momentary lagging.</p>	<p>On the System Monitoring page in the console, you can check the Expired Keys metric to confirm whether a large number of keys expire at a certain point in time. For directions, see Update Notes of Monitoring at Five-Second Granularity.</p>	<p>You can disperse the expiration time of those keys in the business logic to avoid them all expiring at the same time.</p>

8	Databases are frequently switched by executing the <code>select</code> command.	Frequent database switch leads to excessive resource overhead.	With DBbrain's Latency Analysis feature, you can check the monitoring data of the <code>select</code> command in Command Word Analysis , and confirm whether there are many <code>select</code> requests. For directions, see Command Word Analysis .	<ul style="list-style-type: none">• For the storage of different businesses, if some of them require frequent database switch, we recommend that you store them separately.• For the storage of the same businesses, if the key names do not conflict, you can consider storing them in the same database to reduce the number of <code>select</code> request operations.
---	---	--	---	--

High Outbound Traffic

Last updated: 2026-03-17 18:23:49

Symptom

- Symptom 1: You received an alarm about the outbound traffic being restricted because the outbound traffic metric reached the maximum value allowed.
- Symptom 2: The response latency increased.

Possible Causes

- Big keys.

If the value of the request key is big, it is easy to cause the problem of high outbound traffic.

- Pipeline requests.

Pipeline technology combines multiple requests into one request and send it to the database server for processing, receives all command execution results, and then returns to the upper-layer business. If there are many requests at a time, the outbound traffic will be too high. For more information, see [Redis pipelining documentation](#).

- Batch query, such as mget, hmget or hgetall, etc.

The number of keys that are queried in a single batch is large, causing the outbound traffic to be too high.

- The instance configuration specification is insufficient.

As business and data volume rise, the instance's outbound traffic has reached its upper limit and can't meet the current business traffic demands.

Solutions

Step 1. Troubleshoot big key issues

1. Log in to the [Tencent Cloud Distributed Cache console](#), use DBbrain's **Performance Optimization** feature to check the big key, and create a big key analysis task, and optimize the big key based on the analysis report. For details, see [Memory Analysis](#).
2. If there is a big key, you can reduce the access to the big key without affecting your business. If the value is too large, you can split the object into multiple key-values so that multiple Redis instances will share the pressure. If there are too many keys, you can store multiple keys in a hash structure.

Note:

In order to prevent the generation of big keys, we recommend that you refer to the following suggestions when designing value.

- The string big keys should be under 10 KB in size, and the number of hash, list, set, and zset elements should not exceed 5000.
- For non-string big keys, it is recommended to use hscan, sscan, and zscan to delete them progressively.

Step 2. Check whether the business uses pipeline

To check the business code logic, it is recommended not to use **pipeline** or reduce the number of command requests in each **pipeline**. If you are unsure whether your business has used the pipeline, [submit a ticket](#) for troubleshooting.

Step 3. Check the number of keys in a single query

- Log in to the [Tencent Cloud Distributed Cache console](#), use DBbrain's **Performance Optimization** feature to analyze the change trend of the performance metrics **Key Requests** and **Mget Executions**, and check whether there is a sudden increase in key requests. For directions, see [Performance Trends](#).
- Generally, it is recommended that the number of keys or elements in a single operation should not exceed 50. If the value is relatively large, you need to reduce it.

Step 4. Upgrade the instance specification

1. Increase the number of replicas as instructed in [Changing Instance Specification](#), enable read-only replicas to share the read load, and transfer the read requests of the current instance to the read-only replica nodes to achieve elastic expansion of read capabilities and improve network traffic performance. For specific operations, see [Enabling/Disabling Read/Write Separation](#).
2. Increase the number of shards as instructed in [Changing Instance Specification](#), and the system will allocate more standard bandwidth. If the instance is on a standard architecture, you need to upgrade the standard architecture to the cluster architecture to improve CPU processing capabilities. For directions, see [Upgrading Instance Architecture](#). It is necessary to check the compatibility before upgrading to the cluster architecture. For more information, see [Check on Migration from Standard Architecture to Cluster Architecture](#).

Step 5. Adjust the bandwidth

If the outbound traffic is still high after upgrading the instance specification, you can adjust the bandwidth to its maximum value. Increasing bandwidth is currently free of charge. For directions, see [Bandwidth Adjustment](#).

High Memory Utilization

Last updated: 2026-03-18 17:08:16

Symptom

- Symptom 1: You received an alarm about too high **memory utilization**.
- Symptom 2: On the **System Monitoring** page in the [Tencent Cloud Distributed Cache console](#), you can view that there are sudden rises in **Memory Utilization**, **Key Evictions**, and **P99 Response Latency**.
- Symptom 3: The error message `command not allowed when used memory > 'maxmemory'` was prompted when the program writes data.

Distributed CacheMemory Usage

The data that usually occupies memory in Tencent Cloud Distributed Cache includes the following:

- Object memory: User data area, that is, the actual stored value information.
- Buffer memory: Including client input and output buffers, and master-replica sync replication buffers.

Note:

When performing Range operations or big keys on the client side, the memory occupied by input buff and output buff will increase, affecting the buffer and even causing OOM.

- Memory fragmentation: A large number of update operations, such as append and setrange; a large number of expired keys are deleted, and the released space cannot be effectively used.
- Link memory: The memory consumed for creating subprocesses, which is generally relatively small.

Troubleshooting the Issue

No.	Possible Cause	Troubleshooting Method	Solution
1	<ul style="list-style-type: none"> ● The increase in the volume of written data leads to an increase in memory usage. ● The key is not set in the TTL policy. 	<ol style="list-style-type: none"> 1. Log in to the Tencent Cloud Distributed Cache console, click an instance ID, and enter the instance details page. 2. Select the System Monitoring tab to view the corresponding monitoring metrics of the instance, including 	<p>If the memory usage is proportional to the total number of keys, the possible causes are as follows.</p> <ul style="list-style-type: none"> ● The memory may be increased due to the normal data written by the business. Evaluate your business needs and expand the Distributed Cache instance in time. For directions, see Changing Instance Specification.

		<p>Memory Utilization, Total Keys, Expired Keys, and Evicted Keys. Then, you can analyze whether the memory utilization is consistent with the fluctuation trend of the number of keys.</p>	<ul style="list-style-type: none"> ● The memory may increase due to the accumulation of keys due to the failure to set a reasonable eviction policy and deletion frequency for expired keys. <ul style="list-style-type: none"> ○ The eviction policy setting is unreasonable, which may easily cause invalid keys to occupy too much space. To modify the parameters related to key expiration time, eviction policy, and deletion frequency, reconfigure the <code>maxmemory-policy</code> and <code>hz</code> parameters on the Parameter Settings page in the console. For directions, see Setting Instance Parameters. ○ If the deletion frequency for expired keys is set too high, that is, if the value of <code>hz</code> increases, more CPU resources will be occupied. The <code>hz</code> value should not be too large. You can adjust it as needed.
2	<ul style="list-style-type: none"> ● Input buffer overflow is caused by writing a big key caused the . ● Reading large keys, requesting a large number of command returns, or executing monitor commands causes output buffer overflow. 	<p>Analyze memory skew issues with the Performance Optimization feature of DBbrain. For more information, see Memory Analysis.</p>	<p>Perform value splitting and optimization for abnormally large keys. For more information, see Hot Key and Big Key.</p>

High Number of Total Requests

Last updated: 2026-03-18 10:09:12

Symptom

- Symptom 1: the QPS value was high.
- Symptom 2: the response latency increased.
- Symptom 3: connection timeout occurred.

Possible Causes

- The business needs to be optimized.
- The instance configuration needs to be upgraded.

Solutions

- Check the node load: for the cluster architecture, check the node load. If the QPS of only one or a few nodes exceeds the alarm threshold, there may be a hot key; if the QPS of most nodes is high, the overall load of the Distributed Cache instance is high, in which case the instance configuration needs to be upgraded.
- Check the node load: for the cluster architecture, check the node load. If the QPS of only one or a few nodes exceeds the alarm threshold, there may be a hot key; if the QPS of most nodes is high, the overall load of the Distributed Cache instance is high, in which case the instance configuration needs to be upgraded.
- Check the CPU utilization: you can check whether the CPU utilization is too high, and if so, the machine resources may be insufficient, in which case the instance configuration needs to be upgraded.

If your business requires optimization, you can optimize it in terms of hot keys and big keys. If the instance configuration requires upgrade, you can enable read/write separation and add more shards to meet your current business needs.

Troubleshooting the Issue

1. Log in to the [Tencent Cloud Distributed Cache console](#).
2. In the instance list, locate the target instance and click the instance ID to go to the instance management page.
3. On the **System Monitoring** tab, check whether QPS is high or whether there are unexpected hot keys.
4. After troubleshooting abnormal access, optimize your business logic:
 - Hot keys: You can split hot keys of complex data structures into several new keys and distribute them across Redis nodes to reduce the pressure. For example, if a two-level hash hot key has a lot

of hash elements, you can split it.

- Big keys: If the value is too large, you can split the object into multiple key-values so that multiple Redis nodes will share the pressure. If there are too many keys, you can store multiple keys in a hash structure.

Instance upgrade

Heavy read load

Add replicas and enable [read/write separation](#) to share the read load.

Note:

Confirm that your business allows inconsistent data before enabling read/write separation, because after it is enabled, inconsistent data may be read from the replica node and the master node (the replica node lags behind the master node). For more information, see [Changing Instance Specification](#).

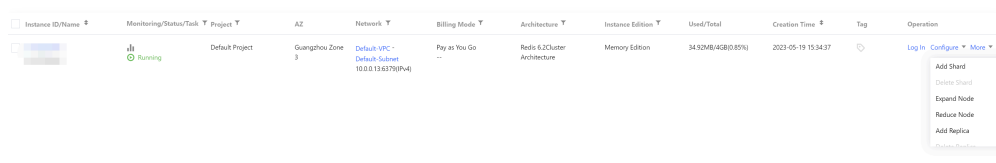
Heavy write load

Cluster Architecture

Log in to the [Tencent Cloud Distributed Cache console](#), find the target instance in the instance list, and select **Configure** > **Add Shard** in the **Operation** column.

Note:

- After the configuration is adjusted, the instance will be charged at the price of the new configuration.
- When shards are added, the system will automatically balance the slot configuration and migrate data, which may fail in rare cases. We recommend that you perform such operations during off-peak hours to avoid the impact of migration on business access.
- Add shards as needed: Each shard supports a QPS of 80,000 to 100,000.

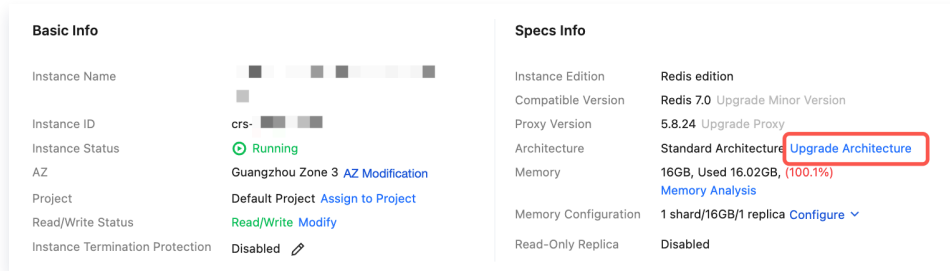


Standard Architecture

Upgrade the instance from standard architecture to cluster architecture to improve the processing power of CPU. Before the upgrade, you need to check the compatibility. For more information, see [Check on Migration from Standard Architecture to Cluster Architecture](#).

1. Log in to the [Tencent Cloud Distributed Cache console](#) and click an instance ID in the instance list to enter the instance details page.

2. In the **Specs Info** section, click **Upgrade Architecture**.



Basic Info		Specs Info	
Instance Name		Instance Edition	Redis edition
Instance ID	crs- [Progress Bar]	Compatible Version	Redis 7.0 Upgrade Minor Version
Instance Status	Running	Proxy Version	5.8.24 Upgrade Proxy
AZ	Guangzhou Zone 3 AZ Modification	Architecture	Standard Architecture Upgrade Architecture
Project	Default Project Assign to Project	Memory	16GB, Used 16.02GB, (100.1%) Memory Analysis
Read/Write Status	Read/Write Modify	Memory Configuration	1 shard/16GB/1 replica Configure
Instance Termination Protection	Disabled	Read-Only Replica	Disabled

3. After the upgrade is completed, go to the instance list and select **Configure > Add Shard** in the **Operation** column.

Note:

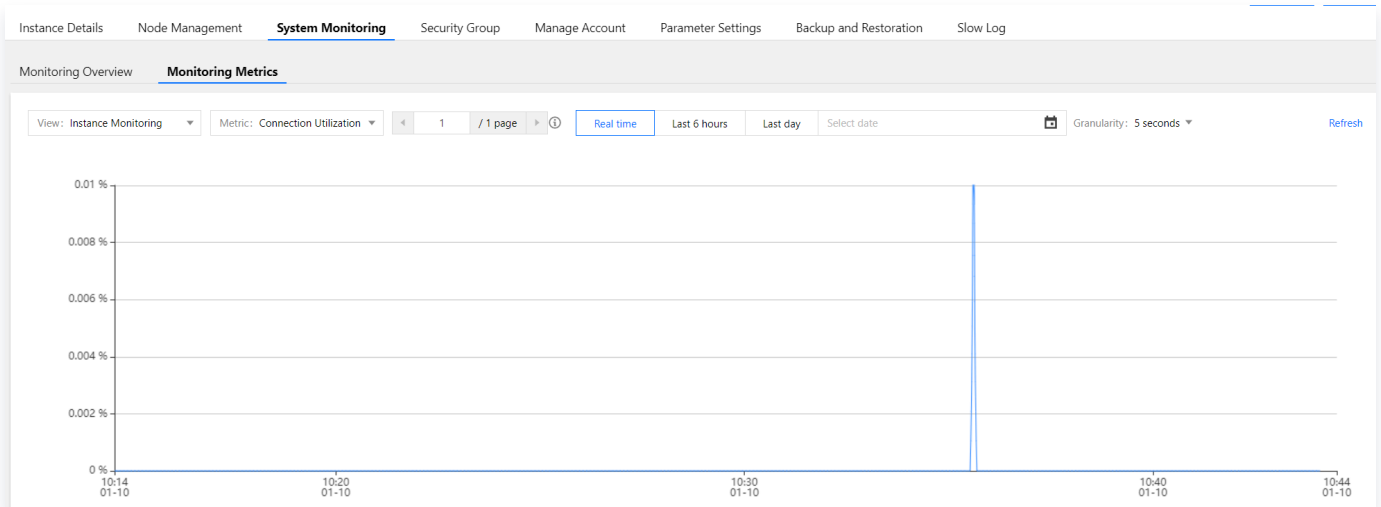
If the problem persists, [contact us](#) for assistance.

High Connection Utilization

Last updated: 2026-03-17 18:23:49

Error Description

- Symptom 1: the connection utilization was high.



- Symptom 2: connections couldn't be established.

Common Causes

- Connections leaked.
- The configured maximum number of connections is too low.

Solution

Check for connection leaks. For third-party network connections, there must be a finally block for execution; otherwise, non-standard code is prone to cause the problem where connections are not released properly.

Troubleshooting Procedure

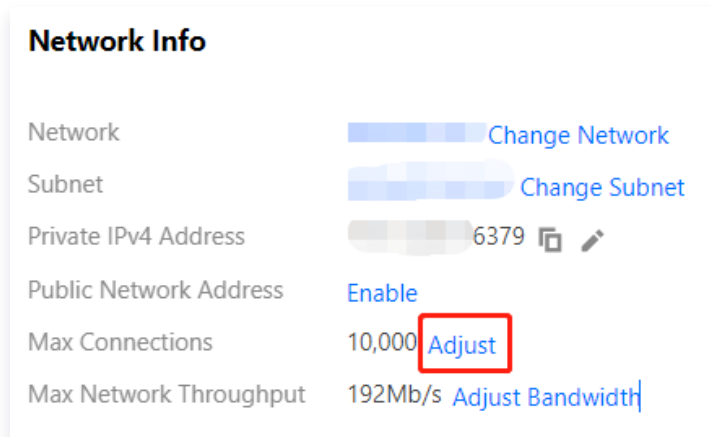
Modifying code logic

Check whether connections leaked, and if so, you can release invalid connections and fix non-standard code that leaked connections.






Modifying connection configuration

1. Log in to the [TencentDB for Redis® console](#), click an instance ID in the instance list, and enter the instance details page.

2. In the **Network Info** > **Max Connections** parameter on the instance details page, click **Adjust** to adjust the maximum number of connections.



Network Info

Network	 Change Network
Subnet	 Change Subnet
Private IPv4 Address	 6379  
Public Network Address	Enable
Max Connections	10,000 Adjust
Max Network Throughput	192Mb/s Adjust Bandwidth

3. In the pop-up window, adjust the maximum number of connections and click **OK**.

Note:

If the problem persists, [submit a ticket](#).

High Execution Latency

Last updated: 2026-03-17 22:12:52

Error Description

- Symptom 1: the execution latency was high.
- Symptom 2: the business was affected.

Common Causes

- The total number of requests was too high.
- The traffic and connections were restricted.
- There were complex commands such as KEYS * and MGET.
- There are big keys and hot keys.
- Cross-AZ or cross-Region access.
- Bandwidth throttling.
- Connection limit exceeded.
- There is network jitter.

Solution

You can check whether the above problems exist based on the monitoring data in the console and your own business logic and make targeted optimizations.

Troubleshooting Procedure

1. If the total number of requests is high, see [High Number of Total Requests](#).
2. If the traffic is restricted, see [High Outbound Traffic](#). If the number of connections is restricted, see [High Connection Utilization](#).
3. Optimize complex commands. If there is a KEYS *command, you can consider using the SCAN command to traverse in batches instead of using KEYS with a higher time complexity. If there is an MGET command, you can consider splitting it into n operations to get n keys.

Note:

If the problem persists, [contact us](#).

Execution Error

Last updated: 2026-03-17 18:23:49

Symptom

- Error 1: an execution error occurred.
- Error 2: the effect is not as expected.

Possible Causes

- The command is entered incorrectly.
- If the command is entered correctly, the cause may be that:
 - There is a deviation in business logic understanding.
 - The memory is full.

Solutions

- If the command is entered incorrectly, enter the correct command.
- If the command is entered correctly, locate the problem by referring to [Error Codes](#). Common problems include the following:
 - Lua script execution failed.
 - The cluster command is executed without the correct node ID.
 - The memory is full.

Troubleshooting the Issue

- For errors caused by the use of commands, you can fix them according to the corresponding descriptions and business logic.
- For execution errors caused by full memory, you can fix them as instructed in [High Memory Utilization](#).

Note:

If the problem persists, [contact us](#) for assistance.