Distributed Cache Service

Troubleshooting Guide

Issue 01

Date 2024-10-30

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Huawei Cloud Computing Technologies Co., Ltd.

Address:	Huawei Cloud Data Center Jiaoxinggong Road Qianzhong Avenue Gui'an New District Gui Zhou 550029 People's Republic of China
Website:	https://www.huaweicloud.com/intl/en-us/

Contents

1 Troubleshooting Redis Connection Failures

Overview

This topic describes why Redis connection problems occur and how to solve the problems.

Problem Classification

To troubleshoot abnormal connections to a Redis instance, check the following items:

- **Connection Between the Redis Instance and the ECS**
- **Public Access to Redis 3.0**
- **Password**
- **Instance Configuration**
- **Client Connections**
- **Bandwidth**
- **Redis Performance**

Connection Between the Redis Instance and the ECS

The ECS where the client is located must be in the same VPC as the Redis instance and be able to communicate with the Redis instance.

- For a Redis 3.0 or professional edition instance, check the security group rules of the instance and the ECS.

  Correctly configure security group rules for the ECS and the Redis instance to allow the Redis instance to be accessed. For details, see **How Do I Configure a Security Group?**
- For a DCS Redis 4.0/5.0/6.0 basic instance, check the whitelist of the instance.

  If the instance has a whitelist, **ensure that the client IP address is included in the whitelist**. Otherwise, the connection will fail. For details, see **Managing IP Address Whitelist**. If the client IP address changes, add the new IP address to the whitelist.

- Check the regions of the Redis instance and the ECS.

  If the Redis instance and the ECS are not in the same region, create another Redis instance in the same region as the ECS and migrate data from the old instance to the new instance by referring to **Migration Solution Notes**.
- Check the VPCs of the Redis instance and the ECS.

  Different VPCs cannot communicate with each other. An ECS cannot access a Redis instance if they are in different VPCs. You can establish VPC peering connections to allow the ECS to access the Redis instance across VPCs.

  For more information on how to create and use VPC peering connections, see **VPC Peering Connection**.

Public Access to Redis 3.0

Before accessing a Redis instance through a public network, ensure that the instance supports public access. For details, see the **public access explanation**.

- **Symptom**: "Error: Connection reset by peer" is displayed or a message is displayed indicating that the remote host forcibly closes an existing connection.
  - **Possible cause 1**: The security group is incorrectly configured.

    Solution: Correctly configure the Redis instance and access the instance by following the **public access instructions**.
  - **Possible cause 2**: Check whether the VPC subnet where Redis resides is associated with a network ACL and whether the network ACL denies outbound traffic. If yes, remove the ACL restriction.
  - **Possible cause 3**: SSL encryption has been enabled, but Stunnel is not configured during connection. Instead, the IP address displayed on the console was used for connection.

    Solution: When enabling SSL encryption, install and configure the Stunnel client. For details, see **Connecting to Redis with SSL Encryption**. In the command for connecting to the Redis instance, the address must be set to the IP address and port number of the Stunnel client. Do not use the public access address and port displayed on the console.
- **Symptom**: Public access has been automatically disabled.

  Cause: The EIP bound to the DCS Redis instance is unbound. As a result, public access is automatically disabled.

  Solution: Enable public access for the instance and bind an EIP to the instance on the management console. Then, try again.

Password

If the instance password is incorrect, the port can still be accessed but the authentication will fail. If you forget the password, you can reset the password. For details, see **Resetting Instance Passwords**.

Instance Configuration

If a connection to Redis is rejected, log in to the DCS console, go to the instance details page, and modify the **maxclients** parameter. For details, see **Modifying Configuration Parameters of an Instance**.

Client Connections

- The connection fails when you use redis-cli to connect to a Redis Cluster instance.

  Solution: Check whether **-c** is added to the connection command. Ensure that the correct connection command is used when connecting to the cluster nodes.

  - Run the following command to connect to a Redis Cluster instance:

    ./redis-cli -h *{dcs_instance_address}* -p 6379 -a *{password}* -c

  - Run the following command to connect to a single-node, master/standby, or Proxy Cluster instance:

    ./redis-cli -h *{dcs_instance_address}* -p 6379 -a *{password}*

  For details, see **Access Using redis-cli**.

- Error "Read timed out" or "Could not get a resource from the pool" occurs.

  Solution:

  - Check if the **KEYS** command has been used. This command consumes a lot of resources and can easily block Redis. Instead, use the **SCAN** command and do not execute the command frequently.
  - Check if the DCS instance is Redis 3.0. Redis 3.0 uses SATA disks. During AOF persistence, the disk performance may occasionally deteriorate and cause a connection failure. In this case, disable AOF persistence if data persistence is not required. Alternatively, you can use a DCS Redis 4.0 instance or later because they use SSD disks that offer higher performance.

- Error "unexpected end of stream" occurs and causes service exceptions.

  Solution:

  - Optimize the Jedis connection pool by referring to **Recommended Jedis Parameter Settings**.
  - Check whether there are many big keys. For details, see **How Do I Avoid Big Keys and Hot Keys?**

- The connection is interrupted.

  Solution:

  - Modify the application timeout duration.
  - Optimize the service to avoid slow queries.
  - Replace the **KEYS** command with the **SCAN** command.

- If an error occurs when you use the Jedis connection pool, see **Troubleshooting a Jedis Connection Pool Error**.

Bandwidth

If the bandwidth reaches the upper limit of the corresponding instance specifications, Redis connections may time out.

You can view the **Flow Control Times** metric to check whether the bandwidth has reached the upper limit.

Then, check whether the instance has big keys and hot keys. If a single key is too large or overloaded, operations on the key may occupy too many bandwidth resources. For details about big keys and hot keys, see **Analyzing Big Keys and Hot Keys**.

Redis Performance

Connections to an instance may become slow or time out if the CPU usage spikes due to resource-consuming commands such as **KEYS**, or too much memory is used because the expiration time is not set for the instance or expired keys remain in the memory. In these cases, do as follows:

- Use the **SCAN** command instead of the **KEYS** command, or disable the **KEYS** command.
- Check the monitoring data and configure alarm rules. For details, see **Setting Alarm Rules for Critical Metrics**.

  For example, you can view the **Memory Usage** and **Used Memory** metrics to keep track of the instance memory usage, and view the **Connected Clients** metric to determine whether the instance connections limit has been reached.
- Check whether the instance has big keys and hot keys.

  For details about the operations of big key and hot key analysis, see **Analyzing Big Keys and Hot Keys**.

2 Troubleshooting High CPU Usage of a DCS Redis Instance

Symptom

The CPU usage of a Redis instance increases dramatically within a short period of time. If the CPU usage is too high, connections may time out, and master/standby switchover may be triggered.

Possible Causes

1. The service QPS is high. In this case, refer to **Checking QPS**.
2. Resource-consuming commands, such as **KEYS**, were used. In this case, refer to **Locating and Disabling CPU-Intensive Commands**.
3. Redis rewrite was triggered. In this case, refer to **Checking Redis Rewrite**.

Checking QPS

On the **Cache Manager** page of the DCS console, click an instance to go to the instance details page. On the left menu, choose **Performance Monitoring** and then view the **Ops per Second** metric.

If the QPS is high, optimize customer services or **modifying instance specifications**. For details about the QPS supported by different instance specifications, see **DCS Instance Specifications**.

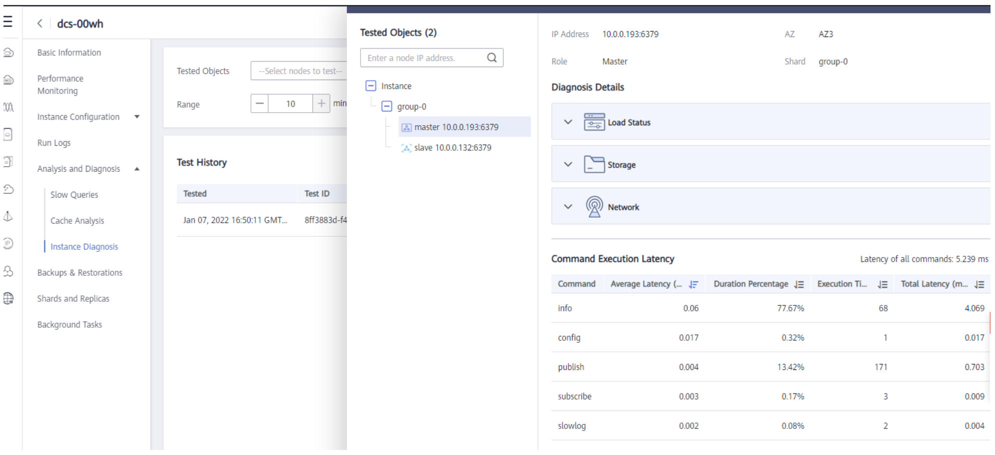
Locating and Disabling CPU-Intensive Commands

Resource-consuming commands (commands with time complexity O(N) or higher), such as **KEYS**, are used. Generally, the higher the time complexity, the more resources a command uses. As a result, the CPU usage is high, and a master/standby switchover can be easily triggered. For details about the time complexity of each command, visit the **Redis official website**. In this case, use the **SCAN** command instead or disable the **KEYS** command.

Step 1 On the **Performance Monitoring** page of the DCS console, locate the period when the CPU usage is high.

Step 2 Use the following methods to find the commands that consume a large number of resources.

- Redis logs queries that exceed a specified execution duration. You can find the commands that consume a large number of resources by analyzing the slow queries and their execution duration. For details, see **Viewing Redis Slow Queries**.
- Use the instance diagnosis function to analyze the execution duration percentage of different commands during the period when the CPU usage is high. For details, see **Diagnosing an Instance**.


dcs-00wh

Basic Information
Performance Monitoring
Instance Configuration
Run Logs
Analysis and Diagnosis
Slow Queries
Cache Analysis
Instance Diagnosis
Backups & Restorations
Shards and Replicas
Background Tasks

Tested Objects: --Select nodes to test--
Range: 10 min

Test History

Tested	Test ID
Jan 07, 2022 16:50:11 GMT...	8ff3883d-f4

Tested Objects (2)

Enter a node IP address.

Instance
group-0
master 10.0.0.193:6379
slave 10.0.0.132:6379

IP Address: 10.0.0.193:6379
AZ: AZ3
Role: Master
Shard: group-0

Diagnosis Details

Load Status
Storage
Network

Command Execution Latency

Latency of all commands: 5.239 ms

Command	Average Latency (...	Duration Percentage	Execution Ti...	Total Latency (m...
info	0.06	77.67%	68	4.069
config	0.017	0.32%	1	0.017
publish	0.004	13.42%	171	0.703
subscribe	0.003	0.17%	3	0.009
slowlog	0.002	0.08%	2	0.004

Step 3 Resolve the problem.

- Evaluate and disable high-risk and high-consumption commands, such as **FLUSHALL**, **KEYS**, and **HGETALL**.
- Optimize services. For example, avoid frequent data sorting operations.
- (Optional) Perform the following operations to adjust instances based on service requirements:
  – Change the instance type to read/write splitting to separate read and write requests from high-consumption commands or applications.
  – Scale up the instance.

----End

Checking Redis Rewrite

AOF persistence, which is enabled by default for DCS Redis instances excluding single-node and single-replica Redis Cluster ones, takes place in the following scenarios:

- If a small amount of data is written and the AOF file is not large, AOF rewrite is performed from 01:00 to 04:00 in the morning every day, and CPU usage may suddenly spike during this period.
- When a large amount of data is written and the AOF file size exceeds the threshold (three to five times the DCS instance capacity), AOF rewrite is automatically triggered in the background regardless of the current time.

Redis rewrite is performed by running the **BGSAVE** or **BGREWRITEAOF** command, which may consume many CPU resources (see **the discussion**). **BGSAVE** and **BGREWRITEAOF** commands need to fork(), resulting in CPU usage spikes within a short period of time.

If persistence is not required, disable it by changing the value of **appendonly** to **no** on the **Parameters** page of the instance. However, if you disable persistence, data loss may occur due to a lack of data flushing to disk in extreme situations.

3 Troubleshooting High Memory Usage of a DCS Redis Instance

Symptom

Redis provides fast database services. If the memory is insufficient, keys may be frequently evicted, the response time may increase, and the QPS may be unstable, affecting service running. This is normal due to Redis functions (such as master/replica replication and lazyfree). When the memory becomes full, scale up the instance or remove unnecessary data. Generally, you need to be alerted when the memory usage exceeds 95%.

Fault Locating

1. Query the memory usage in a specified period. For details, see **Viewing Metrics**. Check whether the value of **Memory Usage** is close to 100% continuously.
2. If the values of **Evicted Keys** and **Maximum Command Latency** increase significantly during the period when the memory usage exceeds 95%, the memory is insufficient.

   In this case, log in to the console and analyze big keys and slow queries by referring to **Cache Analysis** and **Viewing Redis Slow Queries**. If no expiration is set for the instance, too much data will be stored in the instance, using up the memory.
3. If the memory of a Redis instance is full but there are not many keys, the output buffer may have occupied an excessive amount of memory.

   In this case, run the **redis-cli --bigkeys** command to scan for big keys after connecting to the instance using redis-cli. Then, run the **info** command to check the output buffer size.

Solution

1. Perform big key and hot key analysis on the DCS console, and take measures accordingly. For details, see **Analyzing Big Keys and Hot Keys**.

   NOTE

   You can **configure alarms** to detect hot keys for DCS Redis 3.0 instances, which do not support hot key analysis.

2. **Scan for expired keys** and release them, or manually delete unnecessary keys.
3. Other suggestions:
   - **The value of String must be less than or equal to 10 KB.**
   - **For data structures such as Hash, List, Set, and Zset, it is recommended that the number of elements in a single key be less than or equal to 5000.**
   - When naming keys, use the service name abbreviation as the prefix and do not use special characters such as spaces, line brakes, single or double quotation marks, and other escape characters.
   - Do not rely too much on Redis transactions.
   - The performance of short connections ("connect" in Redis terminology) is poor. Use clients with connection pools.
   - If the data is used only for data cache and data loss is tolerated, you are advised to disable the persistence function. (Change the value of **appendonly** to **no** in the instance parameter configuration to disable the AOF persistence function.)
   - Configure alarms to detect big keys and hot keys in advance.
     - Configure an alarm for node-level memory usage. For details, see **Configuring Alarm Rules for Critical Metrics**.

       If a node has a big key, the memory usage of the node is much higher than that of other nodes. In this case, an alarm is triggered to help you find the problematic key.
     - Configure alarms for node-level **Maximum Inbound Bandwidth**, **Maximum Outbound Bandwidth**, and **CPU Usage**. For details, see **Configuring Alarm Rules for Critical Metrics**.

       If a node has a hot key, the bandwidth and CPU usage of the node is much higher than that of other nodes. In this case, an alarm is triggered to help you find the problematic key.
4. If the instance memory usage remains high after you take the preceding measures, expand the instance specifications during off-peak hours. For details, see **Modifying Specifications**.

4 Troubleshooting High Bandwidth Usage of a DCS Redis Instance

Overview

Redis instances are close to application services, and therefore they process a large amount of data access requests and use network bandwidth. The maximum bandwidth varies depending on the instance specifications. When the maximum bandwidth is exceeded, flow control is triggered, and connections are discarded. This may increase the service latency and cause client connection exceptions. This section describes how to troubleshoot high bandwidth usage of a DCS Redis instance.

Procedure

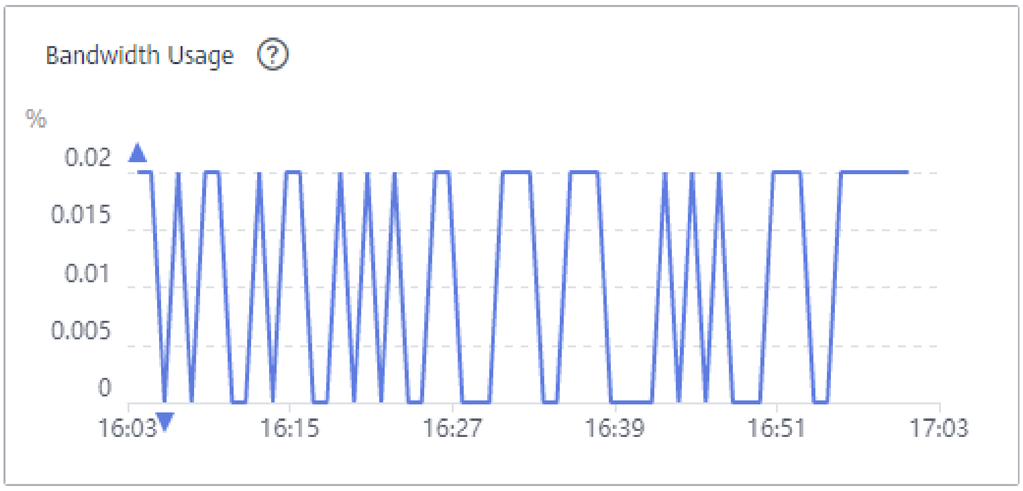
Step 1 Check the bandwidth usage.

Check the bandwidth usage of an instance in a specified period. For details, see **Viewing Metrics**.

Generally, if the input and output flows increase rapidly and remain above 80% of the instance's maximum bandwidth, the bandwidth may become insufficient.

The following figure shows the bandwidth usage. Bandwidth usage = (Input flow + Output flow)/(2 x Maximum bandwidth) x 100%

Figure 4-1 Bandwidth usage



Even if the bandwidth usage exceeds 100%, flow control may not necessarily be triggered and can be reflected on the **Flow Control Times** metric.

Even if the bandwidth usage is below 100%, flow control may still be triggered. The real-time bandwidth usage is reported once in every reporting period. Flow controls are checked every second. The traffic may surge within seconds and then fall back between reporting periods. By the time the bandwidth usage is reported, it may have already restored to the normal level.

Step 2 Optimize the bandwidth usage.

1. The service access traffic may not match the expected bandwidth consumption, for example, the bandwidth usage growth trend is inconsistent with the QPS growth trend. If this happens, analyze whether the traffic increase is from read services or write services by checking the input flow and output flow metrics. If the bandwidth usage on a single node increases, use the cache analysis function to detect big keys by referring to **Analyzing Big Keys and Hot Keys**. Optimize big keys (keys larger than 10 KB). For example, split big keys, access big keys less frequently, or delete unnecessary big keys.
2. If the bandwidth usage is still high, scale up the instance to a larger memory size to carry more network traffic. For details, see **Modifying Specifications**.

   NOTE

   Before the scale-up, you can buy a pay-per-use instance to test whether the desired specifications can meet the service load requirements. After the test is complete, you can release the instance by referring to **Deleting an Instance**.

----End

5 Troubleshooting a Jedis Connection Pool Error

The error message that will possibly be displayed when you use the Jedis connection pool is as follows:

```
redis.clients.jedis.exceptions.JedisConnectionException: Could not get a resource from the pool
```

If this error message is displayed, check whether your instance is running properly. If it is running properly, perform the following checks:

Step 1 Check the network.

1. Check the IP address configurations.

   Check whether the IP address configured on the Jedis client is the same as the subnet address configured for your DCS instance. If public access is enabled for your instance, check whether the IP address configured on the Jedis client is the same as the EIP bound to your instance. If they are inconsistent, modify the IP address configuration and then try again.

2. Test the network.

   Use the ping command and telnet on the client to test the network.

   – If the network cannot be pinged:

     ▪ For intra-VPC access to a DCS Redis 3.0 or professional edition instance, ensure that the client and your DCS instance belong to the same VPC and security group, or the **security group of your DCS instance allows access through port 6379**.

     ▪ For intra-VPC access to a DCS Redis 4.0/5.0/6.0 basic instance, ensure that the client and your DCS instance belong to the same VPC. If you have configured a whitelist for the instance, ensure that the client IP address is in the whitelist. For details, see **Managing IP Address Whitelist**.

     ▪ For public access to a DCS Redis 3.0 instance with SSL encryption, ensure that you have configured the security group of your DCS instance, **allowing access through port 36379**.

     ▪ For public access to a DCS Redis 3.0 instance without SSL encryption, ensure that you have configured the security group of your DCS instance, **allowing access through port 6379**.

– If the IP address can be pinged but telnet failed, restart your instance. If the problem persists after the restart, contact technical support.

Step 2 Check the number of connections.

Check whether the number of established network connections exceeds the upper limit configured for JedisPool. If the number of established connections approaches the configured upper limit, restart the DCS service and check whether the problem persists. If the number of established connections is far below the upper limit, continue with the following checks.

In Unix or Linux, run the following command to query the number of established network connections:

netstat -an | grep 6379 | grep ESTABLISHED | wc -l

In Windows, run the following command to query the number of established network connections:

netstat -an | find "6379" | find "ESTABLISHED" /C

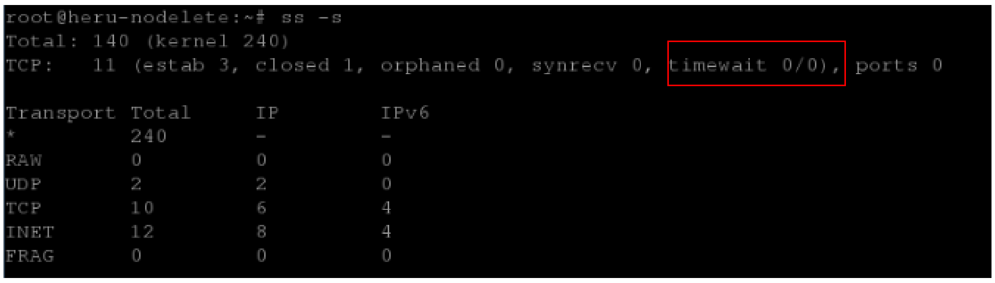
Step 3 Check the JedisPool code.

If the number of established connections approaches the upper limit, determine whether the problem is caused by service concurrency or incorrect usage of JedisPool.

When using JedisPool, you must call **jedisPool.returnResource()** or **jedis.close()** (recommended) to release the resources after you call **jedisPool.getResource()**.

Step 4 Check the number of TIME_WAIT connections.

Run the **ss -s** command to check whether there are too many **TIME_WAIT** connections on the client.



```
root@heru-nodelete:~# ss -s
Total: 140 (kernel 240)
TCP:   11 (estab 3, closed 1, orphaned 0, synrecv 0, timewait 0/0), ports 0

Transport Total     IP        IPv6
*         240       -         -
RAW       0         0         0
UDP       2         2         0
TCP       10        6         4
INET      12        8         4
FRAG      0         0         0
```

If there are too many **TIME_WAIT** connections, modify the kernel parameters by running the **/etc/sysctl.conf** command as follows:

```
##Uses cookies to prevent some SYN flood attacks when the SYN waiting queue overflows.
net.ipv4.tcp_syncookies = 1
##Reuses TIME_WAIT sockets for new TCP connections.
net.ipv4.tcp_tw_reuse = 1
##Enables quick reclamation of TIME_WAIT sockets in TCP connections.
net.ipv4.tcp_tw_recycle = 1
##Modifies the default timeout time of the system.
net.ipv4.tcp_fin_timeout = 30
```

After the modification, run the **/sbin/sysctl -p** command for the modification to take effect.

Step 5 If the problem persists after you perform the preceding checks, perform the following steps.

Capture packets and send packet files along with the time and description of the exception to technical support for analysis.

Run the following command to capture packets:

tcpdump -i eth0 tcp and port 6379 -n -nn -s 74 -w dump.pcap

In Windows, you can also install the Wireshark tool to capture packets.

NOTE

For public access, change the port number to **36379**.

Replace the NIC name to the actual one.

----End