Cloud Load Balance Operations Manual

Contents

Operations Manual
Solution to Excessive Clients in TIME_WAIT Status

Last updated: 2023-09-05 18:51:18

Background

When performing stress testing on CLB, you may encounter connection failures caused by too many client TIME-WAIT (all ports are occupied in short time). Below are reasons and solutions:

Linux Parameter Introduction

tcp_timestamps: Determines whether the TCP timestamps option is enabled. Timestamps are negotiated during the TCP three-way handshake. If either party does not support it, the connection will not use the timestamps option.
tcp_tw_recycle: Determines whether the TCP TIME_WAIT state recycling is enabled.
tcp_tw_reuse: When enabled, connections in the TIME_WAIT state for more than 1 second can be directly recycled.

Cause Analysis

An excessive number of client TIME-WAIT states occur when the client actively disconnects. Each disconnected connection enters the TIME-WAIT state, with a default 60-second timeout for recycling. In general, clients encountering this scenario will enable the `tcp_tw_recycle` and `tcp_tw_reuse` parameters to facilitate recycling connections in the TIME-WAIT state. However, the current CLB does not enable the `tcp_timestamps` option, rendering the client's `tcp_tw_recycle` and `tcp_tw_reuse` ineffective and unable to quickly recycle connections in the TIME-WAIT state. The following sections will explain the meanings of several Linux parameters and the reason why CLB cannot enable `tcp_timestamps`.

1. `tcp_tw_recycle` and `tcp_tw_resuse` only take effect when `tcp_timestamps` is enabled.
2. In FullNAT scenarios, tcp_timestamps and tcp_tw_recycle cannot be enabled simultaneously, as public network clients accessing the server through the NAT gateway may encounter issues. The reasons are as follows:
   When both tcp_tw_recycle and tcp_timestamps are enabled, the timestamp in the socket connect request from the same source IP host within 60 seconds must be incremental. Taking the 2.6.32 kernel as an example, the specific implementation is as follows:

```
if(tmp_opt.saw_tstamp && tcp_death_row.sysctl_tw_recycle &&
(dst = inet_csk_route_req(sk,req))!= NULL &&
 (peer = rt_get_peer((struct rtable *)dst))!= NULL &&
 peer->v4daddr == saddr){
 if(get_seconds()< peer->tcp_ts_stamp + TCP_PAWS_MSL &&
    (s32)(peer->tcp_ts - req->ts_recent) > TCP_PAWS_WINDOW){
       NET_INC_STATS_BH(sock_net(sk),LINUX_MIB_PAWSPASSIVEREJECTED);
       goto ↓drop_and_release;
 }
}
```

Note:
tmp_opt.saw_tstamp: This socket supports tcp_timestamp.
sysctl_tw_recycle: The local system has the tcp_tw_recycle option enabled.
TCP_PAWS_MSL: 60s, this condition indicates that the last TCP communication from this source IP occurred within 60 seconds.
TCP_PAWS_WINDOW: 1, this condition indicates that the timestamp of the last TCP communication from this source IP is greater than the current TCP timestamp.

3. On CLB (Layer-7), tcp_timestamps is disabled because the public network client may fail to access the server through the NAT gateway, as shown in the example below:
   3.1 A quintuple is still in the TIME_WAIT state. The NAT gateway's port allocation policy reuses the same quintuple within 2MSL and sends a SYN packet.
   3.2 When tcp_timestamps is enabled, the SYN packet will be discarded if the following two conditions are met (since the timestamp option is enabled, it is considered an old packet):
      3.2.1 Last timestamp > Current timestamp.
      3.2.2 Within 24 days, packets have been received (the timestamp field is 32-bit, and Linux updates the timestamp every 1ms by default. Timestamp wraparound occurs after 24 days).

Note:
This issue is more prominent on mobile devices, as clients share limited public IP addresses under the carrier's NAT gateway. The five-tuple may be reused within the 2MSL period, and timestamps from different clients cannot be guaranteed to be incremental.

Taking 2.6.32 kernel as an example, the details are as follows:

```
static inline int tcp_paws_check(const struct tcp_options_received *rx_opt,int paws
{
 if((s32)(rx_opt->ts_recent - rx_opt->rcv_tsval)<= paws_win)
 return 1;
 if(unlikely(get_seconds()>=rx_opt->ts_recent_stamp + TCP_PAWS_24DAYS))
 return 1;
 return 0;
}
```

Note:
rx_opt->ts_recent: The timestamp of the previous instance.
rx_opt->rcv_tsval: The timestamp received this time.
get_seconds(): The current time.
rx_opt->ts_recent_stamp: The time when the last packet was received.

Solution

If the client has too many TIME_WAIT, see below for solutions:

- With HTTP short connections (Connection: close), the CLB instance actively closes the connection, preventing the client from generating TIME_WAIT.
- If the scenario requires using persistent connections, enable the SO_LINGER option on the socket and close the connection using RST to avoid entering the TIME_WAIT state, achieving the purpose of quickly recycling ports.

Load Balancer HTTPS Service Performance Testing

Last updated: 2023-09-05 18:52:36

1. CLB Cloud Load Balancer HTTPS Capability Overview

Tencent Cloud CLB Cloud Load Balancer achieves a significant improvement in HTTPS performance by deeply optimizing the protocol stack and server-side. Additionally, through international collaboration on certificates, we have substantially reduced certificate costs. Tencent Cloud CLB can bring remarkable benefits to your business in the following aspects:

1. The use of HTTPS does not affect the access speed of the client.
2. SSL encryption and decryption performance of a single server in a cluster can sustain full handshakes of up to 65,000 connections per second (CPS), which is at least 3.5 times higher than that of a high-performance CPU. This reduces server costs, greatly improves service capability during business peaks and traffic surges, and strengthens the computation-based anti-attack capability.
3. Offloading and conversion of multiple protocols are supported, which reduces the business stress in adapting to various client protocols. The business backend only needs to support HTTP/1.1 to use different protocols such as HTTP/2, SPDY, SSL 3.0, and TLS 1.2.
4. One-stop SSL certificate application, monitoring, and replacement services are provided. Tencent Cloud cooperates with Comodo and Symantec, two leading global certificate authorities, to simplify the certificate application process and reduce application costs.
5. Anti-CC and WAF features are provided to effectively defend against various attacks at the application layer, such as slow HTTP attacks, high-traffic DDoS attacks, SQL injections, and website trojans.

2. Test Objectives

HTTPS service offers advantages such as identity verification, information encryption, and integrity validation. However, implementing secure communication through the SSL protocol inevitably results in some performance loss, mainly in terms of increased latency and CPU resource consumption for encryption and decryption. This article presents the performance data of Tencent Cloud HTTPS service under SSL encryption and decryption scenarios, providing a reference for users to compare with traditional HTTPS performance data.

3. Test Environment

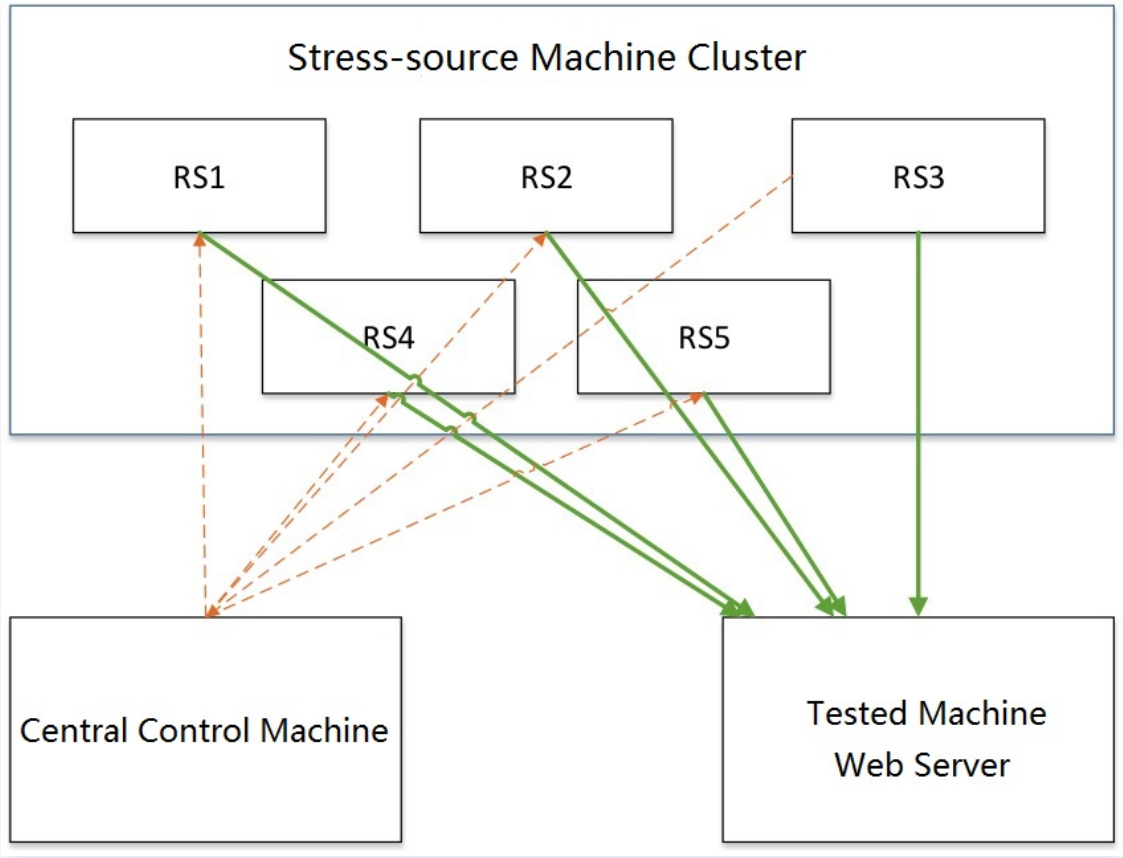
- Stress Test Tool: wrk 4.0.2
- Tencent Cloud underlying service environment: Nginx 1.1.6_1.9.9 + OpenSSL 1.0.2h

- Nginx installation machine operating system information: Linux TENCENT64.site 3.10.94-1-tlinux2-0036.tl2 #1 SMP Thu Jan 21 03:40:59 CST 2016 x86_64 x86_64 x86_64 GNU/Linux
- Other stress test machine operating system: Linux TENCENT64.site 2.6.32.43-tlinux-1.0.17-default #1 SMP Tue Nov 17 18:03:12 CST 2015 x86_64 x86_64 x86_64 GNU/Linux

4. WebServer Cluster Testing Scheme

The stress from a single server is not enough to test the performance limit of Tencent Cloud's HTTPS service. Multiple stress servers are needed. The test includes three parts:

1. Stress testing cluster: Used for sending HTTP/HTTPS stress and outputting the stress test results of individual machines.
2. Central control server, which synchronously controls the start and end of the stress testing cluster, obtains testing data from each stress server, aggregates and outputs the data.
3. Web server, which is the CVM instance hosting Tencent Cloud's HTTPS service. When WebServer performance is tested, a page can be returned directly without upstream connection.
   The connection is as follows:



5. HTTPS Web Server Performance Test Data

Connection Type	Session cache	Payload (bytes)	Cipher suite	Performance (QPS)
Long	Open	230	ECDHE-RSA-AES128-GCM-SHA256	296241
Short	None	230	ECDHE-RSA-AES128-GCM-SHA256	65630

6. CLB HTTPS Capability Test Conclusion

According to the table above, Tencent Cloud's HTTPS service supports SSL encryption and decryption. It has multiple server clusters on the backend, and a single server in a cluster can achieve a performance of up to 65,000 QPS during full handshake and of about 300,000 QPS during persistent connection.
Under normal circumstances, the HTTPS protocol, due to the use of SSL, adds at least one full handshake process, resulting in an increased latency of 2 × RTT. Furthermore, SSL symmetric/asymmetric encryption consumes a significant amount of CPU resources, and RSA decryption capability is the primary challenge for HTTPS access.
By utilizing Tencent Cloud's Cloud Load Balancer HTTPS service, users do not need to deploy separate services for SSL encryption and decryption, and Tencent Cloud does not charge any additional fees. This allows users to effortlessly possess robust business carrying capacity and anti-attack capabilities.

Stress Testing FAQ

Last updated: 2023-09-05 18:53:58

Based on customer experiences in stress testing, this document summarizes common performance issues in stress testing, and provides troubleshooting solutions as well as suggestions.

Stress Testing FAQs

The public network access is not enabled on real server

If public network access is not enabled when you purchase CVM, forwarding may fail when a public network CLB is mounted to the CVM instance.

The bandwidth of a real server is insufficient

If the real server has a low bandwidth, it cannot return packets to the CLB when the threshold is exceeded. CLB will return a 504 or 502 error to the client.

The client ports are insufficient

If the number of clients is too small or the port range of the clients is set too low, there will be insufficient client ports, leading to connection failures. Additionally, if the keep_alive field is greater than 0 when establishing a long connection, the connection will continuously occupy the port, resulting in insufficient client ports.

Applications relied on by real servers have performance issues

After a request reaches a real server through CLB, the load on the real server is normal. However, because applications on real servers also rely on other applications such as database, performance issues in the database may also affect the stress testing performance.

A real server is unhealthy

The health status of real servers may be ignored in stress testing. If the real server has a health check failure or unstable health check status (sometimes good and sometimes bad with rapid changes), stress testing may have poor performance.

Session persistence enabled for CLB results in uneven traffic distribution among real servers

When session persistence is enabled in CLB, requests are more likely to be directed to a few specific real servers, resulting in an uneven distribution of traffic and affecting stress test

performance. It is recommended to disable session persistence during stress testing.

Stress Testing Recommendations

> **Note:**
> The following settings are for stress testing CLB capabilities only and do not imply that users' production environments should be configured in the same way.

- When stress testing the forwarding capabilities of CLB, it is recommended to use short connections. Generally, stress testing aims to verify the forwarding capabilities of CLB, apart from validating features like session persistence. Therefore, short connections can be used to test the processing capabilities of CLB and real servers.
- When stress testing CLB throughput, it is recommended to use persistent connections for testing bandwidth caps or long-lived connections. In this case, adjust the timeout value of the stress testing tool to a smaller threshold. A longer timeout may result in an increased average response time, which is not conducive to quickly determining whether the stress test level has been reached.
- We recommend that the real server provides a static web page for stress testing to avoid losses caused by the application's logic, such as I/O and database operations.
- Do not enable session persistence for the listener, as it may cause stress to concentrate on individual real servers. Additionally, when stress performance is suboptimal, you can check the monitoring data of real servers under the CLB instance to determine if the traffic distribution is even.
- Disabling health check for listeners reduces the number of health check requests to real servers.
- Conducting stress tests with multiple clients (> 5) and distributed source IPs can better simulate real-world online scenarios.

CLB Certificate Operation Permissions

Last updated: 2023-09-05 18:55:39

Scenario

As of March 23, 2020, all certificate-related operations for CLB have been integrated with Cloud Access Management (CAM) authentication. Therefore, when using a sub-user account for CLB certificate-related operations, if you receive a message stating "This operation requires authorization, please contact your developer to grant you permission," you can follow the steps below to grant certificate-related permissions to the sub-user account.

Preparations

The account used for login must be either a root account or a sub-user account with CAM-related permissions, specifically, a sub-user account associated with the QcloudCamFullAccess policy (full read and write access to Users and Permissions (CAM)).

Note:

- To check the CAM-related permissions of a sub-user account, you can go to the User List in the Cloud Access Management console, enter the details page of the corresponding sub-user, and check under "Permissions" whether the QcloudCamFullAccess policy is already associated.
- If you have already associated the QcloudCamFullAccess policy but still encounter the message "No API permissions available (message:GetReceiversOnAllType), please contact the developer for authorization" during the process of granting certificate-related permissions, please ignore the message and continue with the operation.

Instructions

Choose any of the following methods to grant certificate-related permissions.

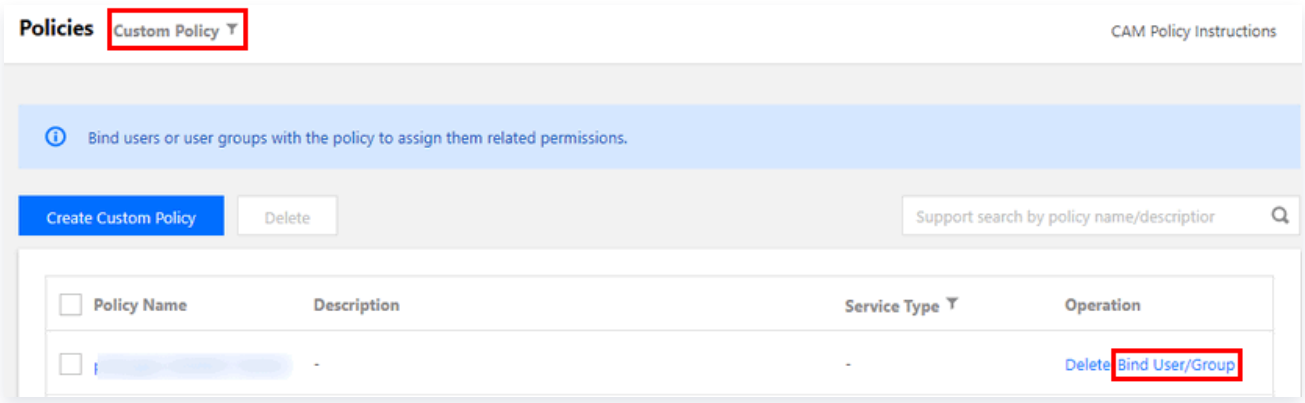
Method 1: Associate a custom policy

1. Log in to the Cloud Access Management console.
2. In the left sidebar, click **Policies** to navigate to the "Policies" list page.
3. Click **Create Custom Policy**. In the pop-up window, select **Create by Policy Syntax**.
4. In the "Select Template Policy" page, choose **Blank Template** and click **Next**.

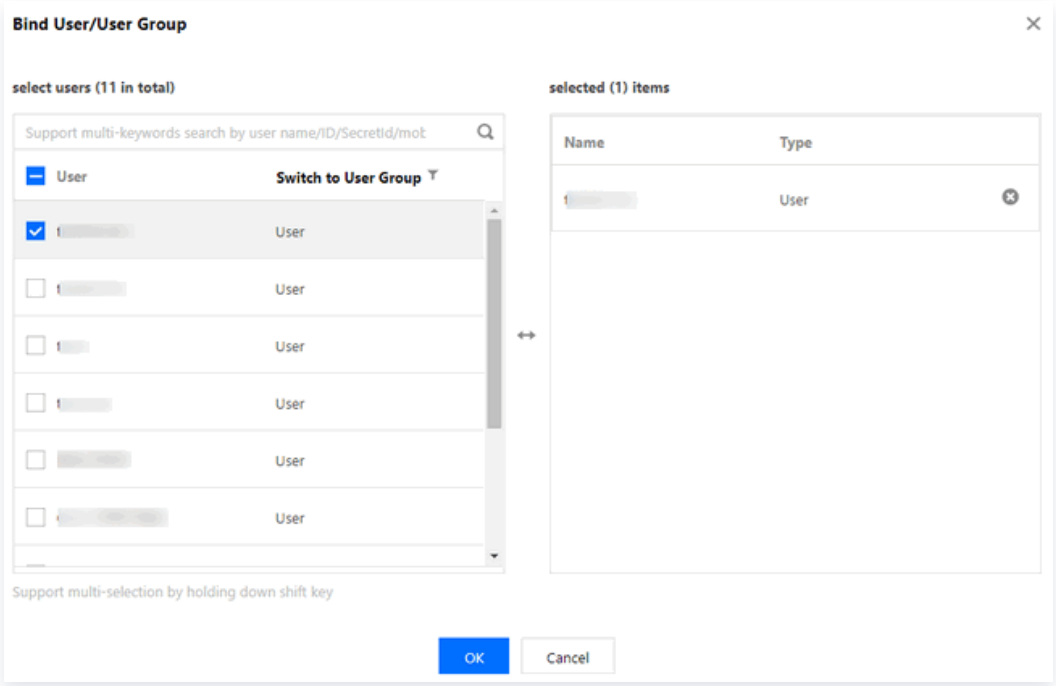
5. On the "Edit Policy" page, enter the policy name, and in the input box for "Edit Policy Content," enter the following policy content.

```
{
 "version": "2.0",
 "statement": [
    {
       "action": "name/ssl:*",
       "resource": "qcs::ssl:::*",
       "effect": "allow"
    }
 ]
}
```

6. Click **Complete** to return to the **Policy** list page.
7. In the "Policy" list page, locate the row containing the newly created policy and click **Associate User/Group/Role** under the operation column.



8. In the pop-up box, select the user to be authorized and click **Confirm** to proceed.



Method 2: Associate Preset Policy

1. Log in to the Cloud Access Management console.
2. In the left sidebar, select **Users** > **User List** to navigate to the "User List" page.
3. In the row of the sub-user requiring authorization, click **Authorize** under the Operation column.
4. In the pop-up box, select either QcloudSSLFullAccess (full read and write access to SSL certificates) or QcloudSSLReadOnlyAccess (read-only access to SSL certificates), and

click **OK** to confirm.