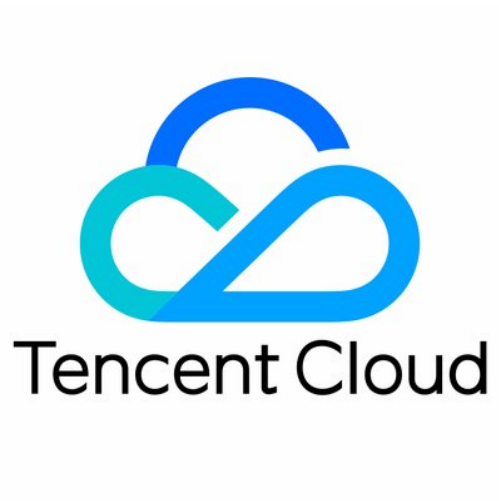
Tencent Smart Advisor-Chaotic Fault Generator

Fault Action Library

Product Documentation



Contents

Primary-secondary Stream Single Path Interruption

Fault Action Library
Compute
JVM Process CPU at Full Load

Last updated：2024-09-26 15:47:37

Experiment Execution

Step 1: Experiment Preparation

Prepare a CVM instance object for the experiment. The instance contains a Java process into which a fault can be injected.

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment.**
2. Click **Skip and create a blank experiment.**
3. Fill in the experiment information. The region can be switched to another region. After completing the basic information, click **Next**.
4. Input action information. Select **Compute - CVM** for Object Type, click **Add Instance**, and select the experiment objective CVM instance.
5. Add an experiment action. Click **Add Now**, select **JVM Fault**, click **JVM process CPU at full load**, and click **Next**.
6. Configure fault action parameters. Assign the JVM process name to be injected, and click **Confirm**.
7. After completing the parameter configuration, click **Next**.
8. Click **Add monitoring metrics**, select **CPU Usage** monitoring metrics, and click **Submit**. Then, an experiment is created.

Step 3: Execute the Experiment

1. Go to the Action Group, click **Execute** in the Fault Card, and inject a fault.
2. After a successful fault injection, target JVM process PID can be obtained from fault logs.
3. You can input a top command at the CVM terminal. You will find that Java process CPU utilization in target Pid is high, and overall CPU utilization is very high.
4. Go to the instance monitoring panel corresponding to the CVM Console to check CPU utilization. It can be seen that CPU utilization in the current instance is high.
5. Click **Execute** in the Recovery Action Card for recovery from a fault.

6. After a successful recovery, check CPU utilization in CVM once again.
7. Check CPU utilization at the instance monitoring panel in the CVM Console. It can be seen that CPU utilization has dropped to a level before the injection.

Cross-AZ Experiment in CVM

Last updated：2024-09-26 15:47:37

Background

To ensure the ability of your business to provide continuous service, CVM products allow cross-AZ deployment so that your applications can be protected from impact in the situation of regional or availability zone faults in some special scenes.
If you are not confident in your services or cloud products, and worry that the impact of an IDC fault on the production environment may result in inaccessibility to your business, you can execute fault simulation and experiment through Tencent Smart Advisor-Chaotic Fault Generator to allow timely avoidance of hidden dangers.

Experiment Objectives

Objective 1

Check whether the cross-AZ service architecture can provide normal services in the case of an availability zone instance down.

Objective 2

Check whether the service recovery time and recovery effects meet business requirements.

Experiment Implementation

Step 1: Preliminary Preparation

Prepare several CVM instances close to the production environment for tests in different availability zones in the same region, and provide identical services.
Prepare complete log recording tools.
Provide emergency measures against unexpected situations.
Count the visits to daily business, and write scripts for simulating user requests.

Step 2: Experimental Design

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment**, and fill in the experiment information.

3. Select the ready test instance object, and configure instance shutdown action for instances in the same availability zone to simulate instance down fault.
After a fault action is added, the 'start up' recovery action will be automatically added. For the experiment, the shell script custom action is added for simulating start up and self-start to start original services in the instance and facilitate observing the recovery situation of the instance.
4. Cloud monitoring metrics or guardrail policy can be configured to observe the operating status of CVM instances.

Step 3: Experiment Implementation

1. Go to Experiment Details, and click **Execute**.
2. Execute fault injection, shut down the instance, and monitor data forwarding by the load balancing traffic.
3. After completing the fault injection experiment, execute fault recovery, and click **Execute** of the recovery action 'start up' to recover instance status. The platform will automatically execute and perform recovery verification.

Note

If booting from startup is not configured, manually trigger the shell script for service recovery.
4. An experiment is completed if all experiment actions are completed. You can click **Record Drill Conclusion** at the top right-hand corner to record experiment results. Register the experiment and record issues in the experiment to allow subsequent replay.

Experiment Result Analysis

Monitoring Metrics via Platform Tools

For a target fault instance, fault injection during execution time will result in the instance down, and CLB monitor will detect that the instance is inaccessible.
Now, traffic will be forwarded to a CVM instance in another availability zone, resulting in a sudden increase in traffic at the time point.
When the fault instance is repaired, that is, after the start up and service restart are completed, CLB monitor will detect that the instance port is healthy and restored to steady status.
Objective Attainment:
In the case of an availability zone down, CLB will automatically forward traffic to another availability zone, making the entire zone available.
When the availability zone is recovered and service is restarted, the steady-status metrics before fault injection can be recovered, and requests can be received and processed normally.
Considering the two results, the overall performance of the cross-AZ fault experiment in CVM meets the expectations.

Theoretical Analysis

Qualitative Analysis: Compare the difference between system metrics and the steady-status metrics during fault injection.

Quantitative Analysis:

System Performance Metrics = Performance metrics in the experiment / Performance metrics at steady status

System Recovery Rate = Performance metrics after an experiment and recovery action / Performance metrics at steady status

Analysis of Causes of System Defects:

Analyze system weaknesses.

Analyze deficiencies in fault handling.

Analyze disturbance resistance of the system.

Analyze monitoring alarm effectiveness.

Analyze dependency relations between modules.

CVM DNS Unavailability Experiment

Last updated：2024-09-26 15:47:38

Background

Cloud Virtual Machine (CVM), launched by Tencent Cloud, has been increasingly widely used. When a CVM is used, the IP address of a server can be quickly located through a Domain Name System (DNS) to allow easy access by users. The complexity introduced by cloud computing architecture also brings forth new stability issues. However, the importance of DNS in the Internet world has long been ignored. Due to malicious DNS pollution, hijacking, and lack of high availability and scalability, DNS has become a popular object of attack. To ensure the continuity and security of businesses of users, the CVM DNS chaos engineering experiment emerges.

Experiment Execution

Step 1: Experiment Preparation

Prepare a CVM instance object for the experiment.

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment.**
2. Click **Skip and create a blank experiment.**
3. Fill in the experiment information, and switch the default region to another region as needed. After filling in the basic information, click **Next**.
4. Input action information, select **Compute -CVM** for Object Type, click **Add Instance**, and select a CVM instance for Experiment Objective.
5. Click **Add Now** to add an experiment action, select **Network Resources**, click **DNS unavailable**, and click **Next**.
6. Configure fault action parameters. Domain name allowlist parameters are supported, indicating a list of domain names available for parsing during a DNS fault. Multiple domain names are separated with English semicolons. Click **Confirm**.
7. After the configuration of parameters is completed, click **Next**.
8. Click **Submit** to create an experiment.

Step 3: Experiment Execution

1. Verify DNS parsing before a fault. Test domain name parsing through nslookup commands.
2. Go to the Action Group 1, click **Execute** in the Fault Card, and inject a fault.

3. After a successful fault injection, check CVM DNS parsing through nslookup.
4. Click **Execute** in the Recovery Action Card to recover DNS.
5. After a successful recovery, observe CVM DNS parsing again. You will find that DNS has returned to normal.

CVM Domain Name Parsing Tampering Experiment

Last updated：2024-09-26 15:47:37

Background

Chaos engineering experiments on CVM domain name parsing and tampering aim to test the CVM coping capability when CVM is facing the attack of DNS parsing and tempering. DNS parsing and tampering is a common network attack. Attackers tamper with DNS parsing results and redirect a website visited by users to a malicious website to steal sensitive information from users or perform other malicious acts. For this reason, CFG provides domain name parsing tampering experiments to test the coping capability and resilience of a business system in the situation of an attack so that business security and stability can be improved.

Experiment Execution

Step 1: Experiment Preparation

Prepare a CVM instance object for the experiment.

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment.**
2. Click **Skip and create a blank experiment.**
3. Fill in the experiment information, and switch the default region to another region as needed. After filling in the basic information, click **Next**.
4. Input action information, select **Compute - CVM** for Object Type, Click **Add Instance**, and select a CVM instance for the experiment objective.
5. Click **Add Now** to add an experiment action, select **Network Resources**, click **Domain name resolution tampering**, and then click **Next**.
6. Configure fault action parameters. Fill in the domain name and IP to be tampered with, separate multiple domain names with English semicolons, and then click **Confirm**.
7. After the configuration of parameters is completed, click **Next**.
8. Click **Submit** to create an experiment.

Step 3: Experiment Execution

1. Verify the parsing before a fault. Test domain name parsing through a ping qq.com command. The parsing is normal.
2. Go to the Action Group 1, click **Execute** in the Fault Card, and inject a fault.
3. After a successful fault injection, test the parsing through ping qq.com. It can be seen that qq.com domain name parsing has been tampered with.
4. Click **Execute** in the Recovery Action Card for recovery.
5. After a successful recovery, retest the parsing through ping qq.com, and it has recovered to normal.

CVM System Time Skew

Last updated：2024-09-26 15:47:38

Background

In a real production environment, various uncertain factors (such as machine power failure, and network delay) may result in inconsistent system time on the nodes in a distributed system. If the nodes have inconsistent time, events generated by the system will be disordered, which may have catastrophic consequences on the system. For this reason, the platform provides a fault action for simulating a real CVM system time skew. Users can verify the resilience of the system in the event of a system time skew.

Experiment Implementation

Step 1: Experiment Preparation

Prepare several CVM instance objects for the experiment.

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment**, fill in the experiment information, and select a CVM instance for an experiment objective.
3. Click **Add Now** to add an experiment action, select **Shell Script**, click **CVM time skew**, and click **Next**.
4. Configure action parameters for fault actions, and click **Confirm**.
5. After action parameter configuration, click **Next**. After confirming all configurations, click **Submit** to complete experiment creation.

Step 3: Experiment Execution

1. Check CVM system time before the experiment.
2. Go to Experiment Details, select actions to be executed, and click **Execute** to start the experiment.
3. Click the Action Card to check details of an action.
4. Check of CVM instance system time, which has skewed.
5. Manually execute recovery actions. You can see that time has returned to normal.

CVM Disk IO Hang Fault Experiment

Last updated：2024-09-26 15:47:38

Background

In some scenes, a business system may cause frequent occupation of a CVM disk, making the disk unreadable and non-writable. The platform provides a CVM disk IO hang fault action to allow users to verify the capability of their systems for handling such abnormality.

Note:

This fault action only supports **Centos7.2, Debian8.2 and Ubuntu16.0.4** and later operating systems.

Experiment Implementation

Step 1: Experiment Preparation

Prepare several CVM instance objects for the experiment.

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment**. Fill in the experiment information, and select a CVM instance for the experiment objective.
3. Click **Add Now** to add an experiment action, select **IO Resources**, and select **IO Hang** fault.
4. Configure fault action parameters.
5. Submit to create an experiment.

Step 3: Experiment Execution

1. In the Experiment Action Group, click **Execute** to execute an experiment and start fault injection.
2. Click the Action Card to check results of the action execution.

Through inquiry of the IO monitoring metrics graph, you can find that IO has dropped and fault action has taken effect.

3. Execute recovery actions. We can see through disk monitoring that IO has recovered.

CVM Memory OOM and Disk IO Load

Last updated：2024-09-26 15:47:38

Background

Memory overflow and disk IO errors are common service errors that usually result in service response reduction or even service unavailability.

Experiment Implementation

Experiment Preparations

Prepare a CVM instance installed with TencentCloud Automation Tools (TAT).

Experiment Steps

Step 1: Create an Experiment

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment**. Fill in the experiment information, and select a CVM instance for the experiment objective.

Step 2: Add Actions

1. Add a **Memory OOM** action and fill in the corresponding parameter fields.
2. Add **Disk IO load** and fill in the corresponding parameter fields.

Step 3: Execute Experiment Actions

Click **Execute** Experiment, execute fault actions after identity verification, and check real-time instance metrics through Tencent Cloud Observability Platform (Instance Details > Monitoring) or monitoring commands.

Check Results

By checking instance memory metrics through the Tencent Cloud Observability Platform, it can be seen that data loss has occurred, which is a result of memory overflow and feedback failure. The result meets the expectations.
Check IO read-write pressure through a vmstat command.

Experiments on CVM Intra-host Network Disorder

Last updated：2024-09-26 15:47:38

Background

To simulate a scene in a production environment where the host receives out-of-order network packets, the network is unstable and requests are disordered. The platform provides a fault action for injecting intra-host network duplication into a CVM. Users can simulate a network disorder scene in a CVM host with this fault.

Experiment Implementation

Step 1: Experiment Preparation

Prepare several CVM instance objects for the experiment.

Step 2: Experiment Orchestration

1. Check network status before fault injection. Send icmp messages to another machine through ping commands. Make sure that the sequence of messages is normal.
2. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment**.
3. Click **Skip and create a blank experiment**, fill in the experiment information, and select a CVM instance for an experiment objective.
4. Click **Add Now** to add an experiment action, select **Network Resources**, click **Intra-host network disorder**, and click **Next**.
5. Configure fault action parameters, and click **Confirm**.
6. After configuring action parameters, click **Next**. After confirming all configurations, click **Submit** to complete experiment creation.

Step 3: Experiment Execution

1. Go to **Experiment Details**. In the Experiment Action Group, click **Execute** to start an experiment and start a fault action.
2. Click the Action Card, and check details of action execution.
3. Check the network status of the host after fault injection. It can be seen that icmp return network packets are out of order.

4. Execute a recovery action, and check details of recovery.
5. Check results of recovery. Through comparison, we can see that the fault has been cleared and the network order has been recovered.

CVM Kernel Faults

Last updated：2024-09-26 15:47:38

Background

CVM kernel faults may result in business failure and impact the stability of the whole system. Hardware faults, kernel software defects, drivers, incompatibility, and other issues may cause kernel faults and CVM failure. For services that rely on high availability, this will cause great inconvenience and loss to users.
To improve business reliability and stability, kernel fault action experiments are required. Through the experiment, the impact of a kernel fault on business can be verified and issues caused by the fault can be revealed in advance so that the faults can be solved quickly and effectively. In solving a kernel fault, please assign personnel with sufficient system knowledge and experience to complete the operation to avoid further damage to the system.

Experiment Implementation

Step 2: Experiment Preparation

Prepare several CVM instances in which **TAT (TAT)** have been installed.

Step 2: Experiment Orchestration

1. Go to Tencent Smart Advisor > Chaotic Fault Generator > Experiment Management, and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment**. Fill in the basic information of the experiment and action groups, and add a target CVM instance.
3. Add an experiment action, select **Kernel Fault** action in **CPU Resources**, click **Next**, and go to parameter configuration.
4. Configure fault action parameters. There is no required parameter for the action. You can click **Confirm** to complete adding.
5. Confirm the configuration, and click **Submit** experiment to complete the creation.

Step 3: Experiment Execution

1. Go to experiment details, and click **Go to the action group for execution**.
2. Click **Execute** to start fault task assignment.
3. Check fault results: The existing connection is interrupted and the instance is restarted.
4. Execute recovery actions.

Note:

Different operating systems have different policies for coping with kernel faults. Automatic restart of the computer is a common method. If there is no response from the operating system, you can manually execute recovery actions on the platform to force a restart.

Experiments on CVM Intra-host Network Latency

Last updated：2024-09-26 15:47:38

Background

Network latency fault is one of the common issues. When such a fault occurs, it will result in the service not responding to user requests properly, and impact the normal operation of businesses will be affected. For businesses that rely on high availability and low delay, network latency will cause great inconvenience and loss to users.
To improve network reliability and stability in CVM, network delay fault experiments are required. Through the experiments, the capability of the system for normal operation in the situation of network delay can be verified and issues in this fault scene can be revealed in advance so that system architecture can be optimized and contingency plans can be prepared.

Experiment Implementation

Step 1: Experiment Preparation

Prepare several CVM instances that are available for the experiment.

Step 2: Experiment Orchestration

1. Check network status before fault injection. Send messages to the target machine through ping commands and wait for a response from the target machine. Check network latency.

```
[ro                    s ~]# ping 172.16.
PING 172.16      (172.16      56(84) bytes of data.
64 bytes from 172.16      icmp_seq=1 ttl=64 time=0.268 ms
64 bytes from 172.16      icmp_seq=2 ttl=64 time=0.147 ms
64 bytes from 172.16      icmp_seq=3 ttl=64 time=0.120 ms
64 bytes from 172.16.     icmp_seq=4 ttl=64 time=0.119 ms
64 bytes from 172.16.     icmp_seq=5 ttl=64 time=0.125 ms
64 bytes from 172.16.     icmp_seq=6 ttl=64 time=0.132 ms
64 bytes from 172.16.     icmp_seq=7 ttl=64 time=0.139 ms
64 bytes from 172.16.     icmp_seq=8 ttl=64 time=0.128 ms
```

2. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment**.

3. Click **Skip and create a blank experiment**. Fill in the experiment information, and select a CVM instance for the experiment objective.

4. Click **Add Now**, select **Network Resources**, click **Intra-host network latency**, and click **Next**.

5. Configure fault action parameters, and click **Confirm**.

6. After action parameter configuration, click **Next**. Configure **Guardrail Policy** and **Monitoring Metrics** considering actual situations, and click **Submit** to complete experiment creation.

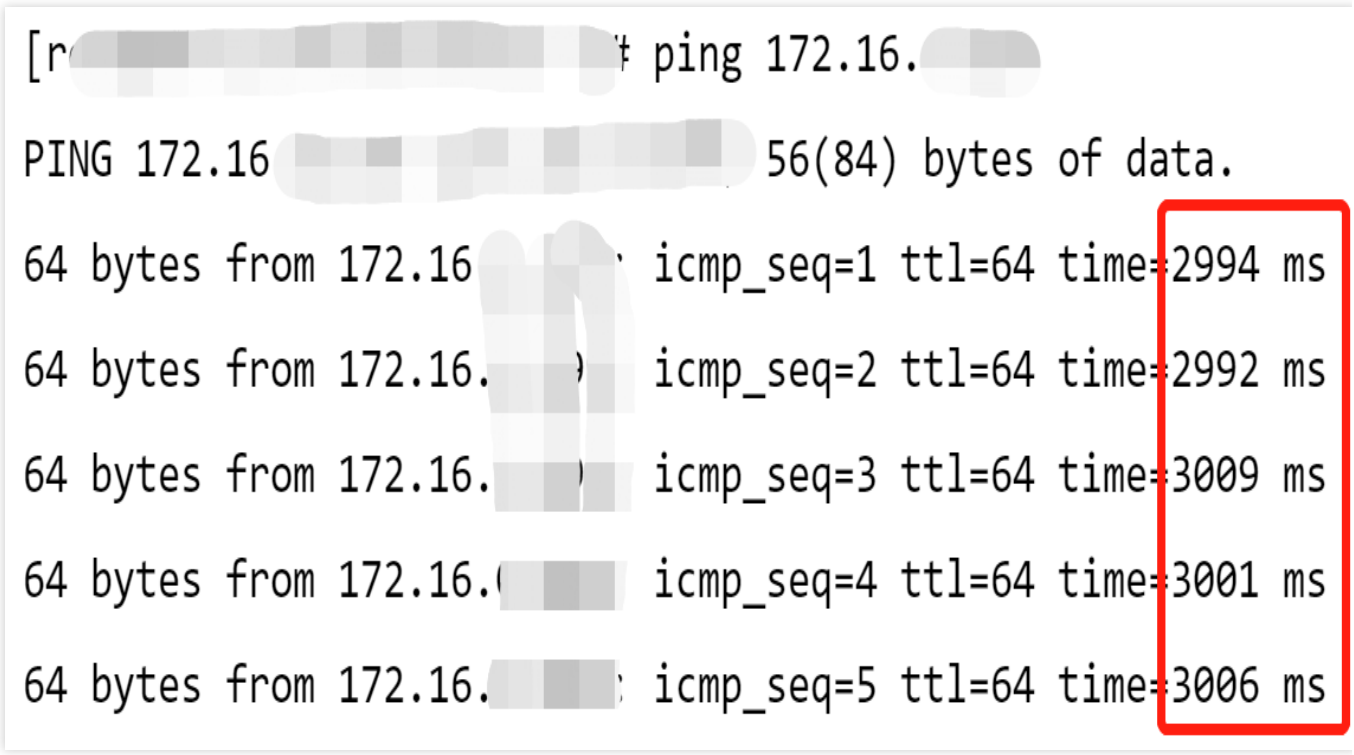
Step 3: Experiment Execution

1. Go to experiment details, and click **Go to the action group for execution**.

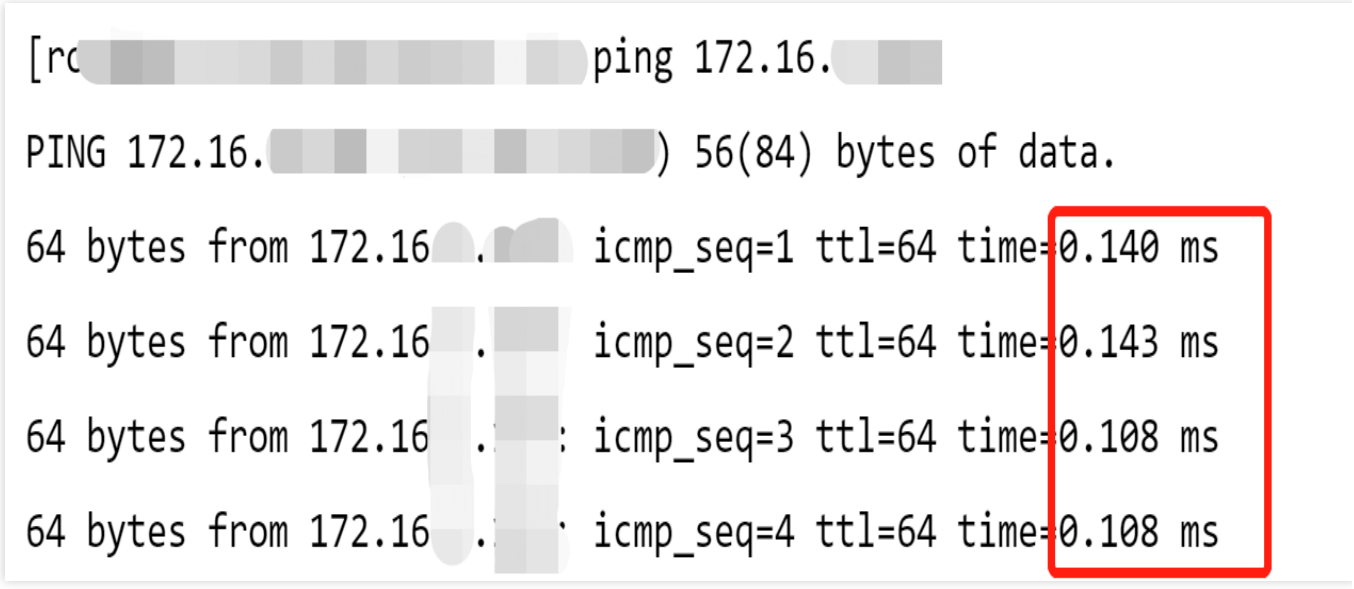
2. Click **Execute** to start an experiment.

3. Click the Action Card to check the details for the action execution results.

4. Check host network status after fault injection. It can be found that when the target machine is pinged again, the returned network packet has been delayed.



5. Execute a recovery action, and check details of the recovery action.
6. Check recovery result. When the target machine is pinged again, it can be seen that the fault has been cleared and normal network speed has recovered.



High Utilization of CVM Resources (CPU, Memory, and Disk)

Last updated：2024-09-26 15:47:38

Background

As one of the most basic cloud resources, Cloud Virtual Machine (CVM) is widely used. When a CVM is used, program errors, improper configuration, and other factors may result in faults such as high CPU utilization, high memory utilization, and high disk partition utilization, which will lead to CVM performance degradation and even service unavailability so users will suffer a loss.
To improve CVM reliability and stability, fault simulation experiments are required to verify the capability of the system for normal operation when utilization of resources such as CPU, memory, and disk is excessively high so that contingency plans can be prepared in advance.

Experiment Implementation

Step 1: Experiment Preparation

Prepare a CVM instance available for the experiment.
Go to the agent management page, and install an agent for the CVM node. For specific installation steps, see Agent Management for installation.

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, click **Create a New Experiment**, and click **Skip and create a blank experiment.**
2. Fill in the basic information of the experiment.
3. Fill in the experiment action group information, and select **Compute-CVM**.
4. Added experiment instances.
5. To add an experiment action, click **Add Now**, and configure fault action parameters.
Configure **High CPU utilization** fault action parameters.
Note:
CPU Utilization: Specify CPU load percentage, which is 0 to 100.
Duration: Duration of a fault action, upon lapse of which, the agent will automatically recover the fault.
Scheduling Priority: It affects process priority in CPU scheduling. A lower nice value makes it more likely that the process would have a CPU time slice so that its execution priority can be improved. It is effective only if utilization is

100%.

Configure **High memory utilization** fault action parameters.

Note:

Memory Usage Rate: Specify a memory load percentage that is 0 to 100.

Duration: Duration of a fault action, upon lapse of which, the agent will automatically recover the fault.

Enable OOM Protection: If it is enabled, the possibility of fault process OOM-KILL will be reduced, and business processes will be killed first.

Memory Occupation Rate: Memory usage increase per second.

Configure **High disk usage** fault action parameters.

Note:

Disk Directory: A disk directory to be populated, i.e., a directory where files are written.

File size: Size of a file populated.

Disk Usage Rate: Learn disk usage through staf commands, and calculate the file size required for specified utilization.

Reserved space: Size of remaining space.

Duration: Duration of a fault action, upon lapse of which, the agent will automatically recover the fault.

If there are file size, disk utilization, and reserved space parameters, the priority calculation logic is disk utilization > reserved space > file size.

Configure **Disk IO load** fault action parameters.

Note:

Disk Directory: Specify a directory to enhance disk IO, which will apply to the disk it resides on.

Mode: Provide both read and write modes to execute high loads.

Block Size: Specify block size for every read or write.

Number of Blocks: Specify number of blocks to be copied.

Duration: Duration of a fault action, upon lapse of which, the agent will automatically recover the fault.

6. After action parameter configuration, click **Next**. Configure **Guardrail Policy** and **Monitoring Metrics** considering actual situations. After all configurations are completed, click **Submit** to complete experiment creation.

Step 3: Experiment Execution

1. Click **Execute** high CPU utilization action to start an experiment.

2. Observe Monitoring Metrics. It can be seen that the CPU load is up to the specified utilization. Execute a rollback action and then recover.

3. Execute high memory utilization action and configured occupation rate so that specified memory utilization is obtained. Execute a rollback action and then recover a steady status.

Note:

Injection tools collect memory utilization metric from `/proc/meminfo`, and calculation formula is **Percent = (MemTotal-MemAvailable)/MemTotal**.

A metric observation system provided by cloud platform: Tencent Cloud Observability Platform, information of which is also collected from `/proc/meminfo`, but its algorithm **contains no buffer and system cache occupancy**, and there is difference from injection tools, details are given below: **Percent = (MemTotal-MemFree-Buffers-Cached-SReclaimable+Shmem)/MemTotal**.

Memory information of this experiment instance is as follows. The following results are obtained through metric substitution in the above two algorithms:

```
[root@VM-22-12-tencentos ~]# cat /proc/meminfo
MemTotal:        1721620 kB    //Total system memory (RAM) size
MemFree:          111260 kB    //Unused memory size
MemAvailable:     349964 kB     //Memory size available for starting a new process.
Buffers::           59624 kB    //Memory size for file system buffer
Cached:           570612 kB    //Memory size for file system cache
......
Shmem:            269980 kB    //Shared memory size
......
SReclaimable:      46308 kB    //Reclaimable cache size of kernel memory
```

Utilization achieved with injection tool is: (1721620-349964)/1721620 = 79.6%

Utilization achieved through Tencent Observability Platform is: (1721620-111260-59624-570612-46308+269980)/1721620 = 69.9%

4. To execute a high disk utilization action, log in to the machine and check that the disk has specified utilization through a **df** command. Execute a rollback action to recover the normal status.

In fault

```
[root@VM-22-12-tencentos ~]# df
Filesystem     1K-blocks     Used Available Use% Mounted on
devtmpfs          840484        0    840484   0% /dev
tmpfs             860808    10268    850540   2% /dev/shm
tmpfs             860808    42420    818388   5% /run
tmpfs             860808        0    860808   0% /sys/fs/cgroup
/dev/vda1       51486416 38971152  10297004  80% /
tmpfs             172160        0    172160   0% /run/user/0
```

After rollback

```
[root@VM-22-12-tencentos ~]# df
Filesystem     1K-blocks    Used Available Use% Mounted on
devtmpfs          840484       0    840484   0% /dev
tmpfs             860808   10268    850540   2% /dev/shm
tmpfs             860808  263604    597204  31% /run
tmpfs             860808       0    860808   0% /sys/fs/cgrou
/dev/vda1       51486416 7551248  41716908  16% /
tmpfs             172160       0    172160   0% /run/user/0
```

5. Execute disk IO high load action, go to the terminal, and use the iostat command for observation.

In fault

```
04/12/2024 04:36:56 PM
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           1.52    0.00    3.05   20.30    0.00   75.13

Device            r/s     w/s     rMB/s     wMB/s   rrqm/s   wrqm/s  %rrqm  %wrqm r_await w_await aqu-sz rareq-sz wareq-sz  svctm  %util
vda          4446.00    0.00     42.95      0.00     0.00     0.00   0.00   0.00    0.22    0.00   0.01     9.89     0.00   0.22 100.00
```

After rollback

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           3.03    0.00    2.02    0.51    0.00   94.44

Device            r/s     w/s     rMB/s     wMB/s   rrqm/s   wrqm/s  %rrqm  %wrqm r_await w_await aqu-sz rareq-sz wareq-sz  svctm  %u
vda              0.00   28.00      0.00      0.33     0.00    45.00   0.00  61.64    0.00    0.39   0.00     0.00    12.14   0.43   1
```

Experiments on CVM Intra-host Network Corruption

Last updated：2024-09-26 15:47:38

Background

CVM intra-host network corruption fault is one of the common issues, that hardware faults, improper network configuration, network congestion, and other issues may cause. It will result in CVM being unable to respond to user requests and impact the normal operation of businesses. For businesses that rely on high availability and low delay, network corruption will cause great inconvenience and loss to users.
To improve network reliability and stability in CVM, network corruption fault experiments are required. Through the experiments, the capability of the system for normal operation in the situation of network damage can be verified and issues in network corruption fault scenes can be revealed in advance so that system architecture can be optimized and contingency plans can be prepared.

Experiment Implementation

Step 1: Experiment Preparation

Prepare several CVM instances that are available for the experiment.

Step 2: Experiment Orchestration

1. Check network status before fault injection. Send messages to the target machine through ping commands, and wait for a response from the target machine to check network connectivity. If no response is received from the target machine or there is a high packet loss rate, there may be an issue of network corruption.

```
[ro                    s ~]# ping 172.16.
PING 172.16    (172.16    56(84) bytes of data.
64 bytes from 172.16    icmp_seq=1 ttl=64 time=0.268 ms
64 bytes from 172.16    icmp_seq=2 ttl=64 time=0.147 ms
64 bytes from 172.16    icmp_seq=3 ttl=64 time=0.120 ms
64 bytes from 172.16.   icmp_seq=4 ttl=64 time=0.119 ms
64 bytes from 172.16.   icmp_seq=5 ttl=64 time=0.125 ms
64 bytes from 172.16.   icmp_seq=6 ttl=64 time=0.132 ms
64 bytes from 172.16.   icmp_seq=7 ttl=64 time=0.139 ms
64 bytes from 172.16.   icmp_seq=8 ttl=64 time=0.128 ms
```

2. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment**.

3. Click **Skip and create a blank experiment**. Fill in the experiment information, and add a target CVM instance.

4. Click **Add Now**, select **Network Resource**, click **Intra-host network corruption**, and click Next.

5. Configure fault action parameters, and click **Confirm**.

6. After action parameter configuration, click **Next**. Configure **Guardrail Policy** and **Monitoring Metrics** considering actual situations, and click **Submit** to complete experiment creation.

Step 3: Experiment Execution

1. Go to experiment details, and click **Go to the action group for execution**.

2. Click **Execute** to start an experiment.

3. Click the Action Card to check the details for the action execution results.

4. Check host network status after fault injection. It can be seen when the target machine is pinged again that the returned network packet is partially damaged.

```
[roo                    ~]# ping 172.16.
PING 172.16.                    56(84) bytes of data.
64 bytes from 172.16    icmp_seq=2 ttl=64 time=0.150 ms
64 bytes from 172.16    icmp_seq=4 ttl=64 time=0.132 ms
64 bytes from 172.16.   icmp_seq=5 ttl=64 time=0.116 ms
64 bytes from 172.16.   icmp_seq=6 ttl=64 time=0.119 ms
64 bytes from 172.16.   icmp_seq=8 ttl=64 time=0.131 ms
64 bytes from 172.16.   icmp_seq=10 ttl=64 time=0.123 ms
```

5. Execute a recovery action, and check details of the recovery action.

6. Check recovery result. When the target machine is pinged again, it can be seen that normal network transmission has recovered and the fault has been cleared.

```
[roo                    ]# ping 172.16.
PING 172.16.6                   56(84) bytes of data.
64 bytes from 172.16    icmp_seq=1 ttl=64 time=0.176 ms
64 bytes from 172.16    icmp_seq=2 ttl=64 time=0.161 ms
64 bytes from 172.1     icmp_seq=3 ttl=64 time=0.125 ms
64 bytes from 172.16    icmp_seq=4 ttl=64 time=0.127 ms
64 bytes from 172.16    icmp_seq=5 ttl=64 time=0.130 ms
64 bytes from 172.16    icmp_seq=6 ttl=64 time=0.110 ms
64 bytes from 172.16    icmp_seq=7 ttl=64 time=0.133 ms
64 bytes from 172.16    icmp_seq=8 ttl=64 time=0.126 ms
64 bytes from 172.16    icmp_seq=9 ttl=64 time=0.133 ms
64 bytes from 172.16    icmp_seq=10 ttl=64 time=0.129 ms
```

Experiments on CVM Intra-host Network Duplication

Last updated：2024-09-26 15:47:38

Background

CVM intra-host network duplication fault is an unusual issue that is likely to occur. It may be caused by hardware faults, improper network configuration, network equipment faults, and other factors, which will result in repeated data packet transmission by CVM during network communication so that network performance is affected, network congestion is increased and business operation abnormality may be caused.
To improve CVM network reliability and stability, network duplication fault action experiments are required. Through the experiments, the system's capability for normal operation in the event of network duplication can be verified and issues in network duplication fault scenes can be revealed in advance so that system architecture can be optimized and contingency plans can be prepared.

Experiment Implementation

Step 1: Experiment Preparation

Prepare several CVM instances that are available for the experiment.

Step 2: Experiment Orchestration

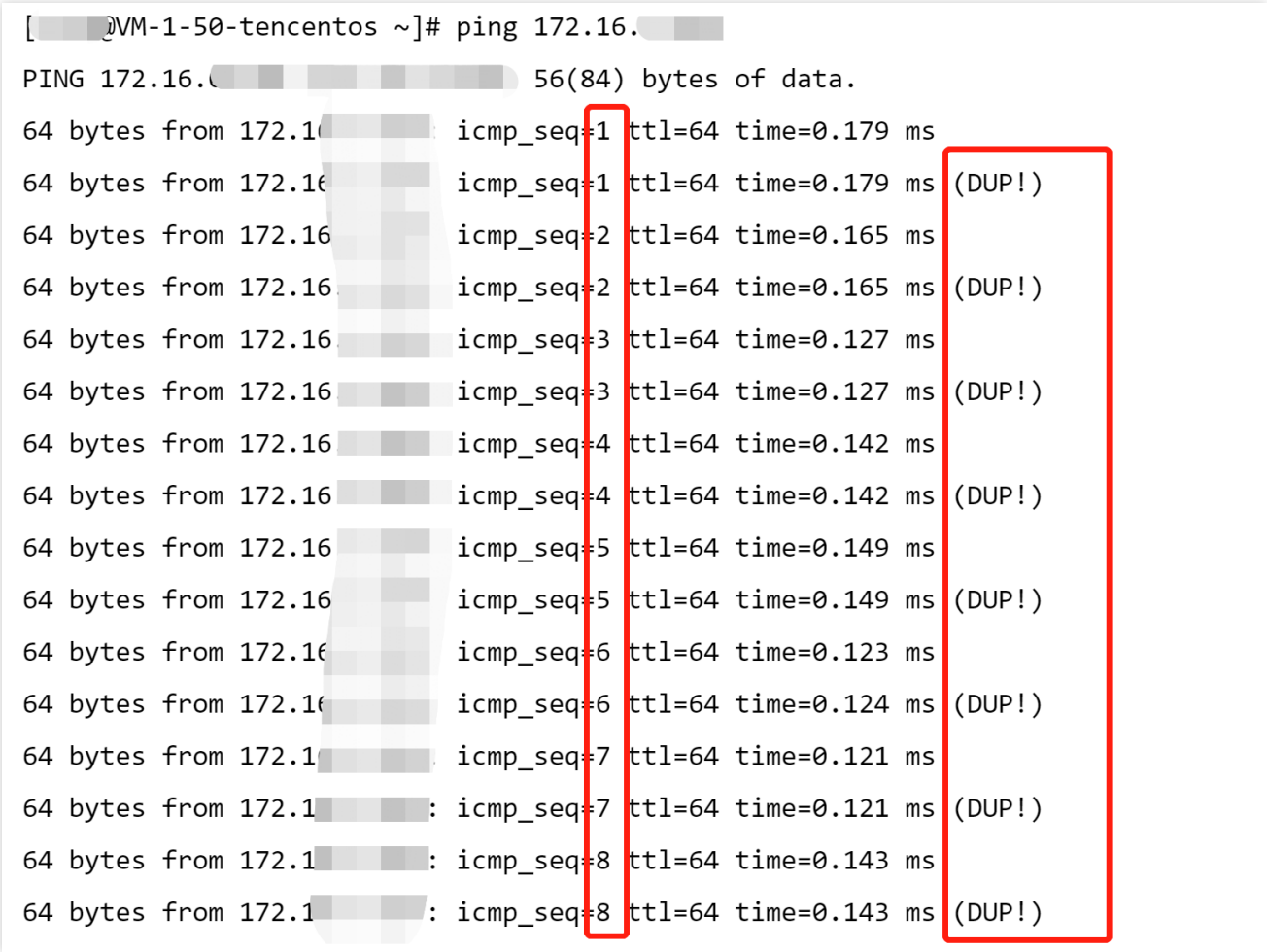
1. Check network status before fault injection. Send messages to the target machine through ping commands. It can be seen that there is no message with the same sequence number, indicating that there is no network duplication at present.

```
[ro                                    ping 172.16.
PING 172.16.                           56(84) bytes of data.
64 bytes from 172.16.                  icmp_seq=1 ttl=64 time=0.155 ms
64 bytes from 172.16.                  icmp_seq=2 ttl=64 time=0.134 ms
64 bytes from 172.16.                  icmp_seq=3 ttl=64 time=0.118 ms
64 bytes from 172.16.                  icmp_seq=4 ttl=64 time=0.114 ms
64 bytes from 172.16.                  : icmp_seq=5 ttl=64 time=0.128 ms
64 bytes from 172.16.                  : icmp_seq=6 ttl=64 time=0.112 ms
64 bytes from 172.16.                  : icmp_seq=7 ttl=64 time=0.129 ms
64 bytes from 172.16.                  : icmp_seq=8 ttl=64 time=0.122 ms
64 bytes from 172.16.                  : icmp_seq=9 ttl=64 time=0.129 ms
64 bytes from 172.16.                  : icmp_seq=10 ttl=64 time=0.121 ms
64 bytes from 172.16.                  : icmp_seq=11 ttl=64 time=0.132 ms
64 bytes from 172.16.                  : icmp_seq=12 ttl=64 time=0.117 ms
```

2. Click **Create a New Experiment**, fill in the experiment information, and add target CVM instances.
3. Click **Add Now**, select **Network Resources**, click **Intra-host network duplication**, and click **Next**.
4. Configure fault action parameters, and click **Confirm**.
5. After action parameter configuration, click **Next**. Configure guardrail policies and monitoring metrics considering actual situations, and click **Submit** to complete experiment creation.

Step 3: Experiment Execution

1. Go to experiment details, and click **Go to the action group for execution**.
2. Click **Execute** to start an experiment.
3. Click the Action Card to check the details for the action execution results.
4. Check host network status after fault injection. When the target machine is pinged once again, returned network packets have been duplicated.



5. Execute a recovery action, and check details of the recovery action.

6. Check fault recovery result. When the target machine is pinged once again, and you can see that normal network transmission has recovered and the fault has been cleared.

```
[ VM-1-50-tencentos ~]# ping 172.16.
PING 172.16.  ) 56(84) bytes of data.
64 bytes from 172.1  : icmp_seq=1 ttl=64 time=0.144 ms
64 bytes from 172.1  : icmp_seq=2 ttl=64 time=0.145 ms
64 bytes from 172.1  : icmp_seq=3 ttl=64 time=0.126 ms
64 bytes from 172.1  : icmp_seq=4 ttl=64 time=0.112 ms
64 bytes from 172.1  : icmp_seq=5 ttl=64 time=0.107 ms
64 bytes from 172.1  : icmp_seq=6 ttl=64 time=0.118 ms
64 bytes from 172.1  icmp_seq=7 ttl=64 time=0.123 ms
64 bytes from 172.16  icmp_seq=8 ttl=64 time=0.130 ms
^C
--- 172.16  ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7157ms
rtt min/avg/max/mdev = 0.107/0.125/0.145/0.017 ms
```

Experiments on CVM Intra-host Network Occupation

Last updated：2024-09-26 15:47:38

Background

CVM intra-host network occupation fault is a common issue, that malicious attacks, program errors, improper configuration, and other issues may cause. It will result in the occupation of a large amount of CVM network resources and impact normal business operations. For businesses that rely on high availability and low delay, network occupation faults will cause great inconvenience and loss to users.
To improve network reliability and stability in CVM, network occupation fault experiments are required. Through the experiments, the system's capability for normal operation in network occupation can be verified and issues in network occupation scenes can be revealed in advance so that system architecture can be optimized and contingency plans can be prepared.

Experiment Implementation

Step 1: Experiment Preparations

Prepare several CVM instances that are available for the experiment.

Step 2: Experiment Orchestration

1. Check port occupancy status before fault injection. Check network port usage in the current system of the target machine through a `netstat -tuln` command.

```
[root@VM-1-50-tencentos ~]# netstat -tuln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:5[redacted]      0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:3[redacted]      0.0.0.0:*               LISTEN
tcp6       0      0 :::5[redacted]           :::*                    LISTEN
tcp6       0      0 :::3[redacted]           :::*                    LISTEN
udp        0      0 127.0.0.1:[redacted]     0.0.0.0:*
udp6       0      0 ::1:[redacted]           :::*
```

2. Click **Create a New Experiment**, fill in the experiment information, and add target CVM instances.
3. Click **Add Now**, select **Network Resources**, click **Intra-host network occupation**, and click **Next**.
4. Configure fault action parameters, and click **Confirm**. Here, the port is set to 8080.

Note:

The port number range is 1 to 65535. 1 to 1023 is for a reserved port, which is usually used for common system services. Try to avoid the use of the same. You can select an unoccupied port from ports 1024-65535 for simulation. After a simulation ends, please clean up and recover the network environment in time to ensure normal system operation. You can check occupied ports through a `netstat -tuln` command.

5. After configuring action parameters, click **Next**. Configure guardrail policies and monitoring metrics considering actual situations, and click **Submit** to complete experiment creation.

Step 3: Experiment Execution

1. Go to experiment details, and click **Go to the action group for execution**.
2. Click **Execute** to start an experiment.
3. Click the Action Card to check the details for the action execution results.
4. Check the host network status after fault injection. You can see that port 8080 has been occupied.

```
[root@VM-1-50-tencentos ~]# netstat -tuln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:5[redacted]      0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:3[redacted]      0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN
tcp6       0      0 :::5[redacted]           :::*                    LISTEN
tcp6       0      0 :::3[redacted]           :::*                    LISTEN
udp        0      0 127.0.0.[redacted]       0.0.0.0:*
udp6       0      0 ::1[redacted]            :::*
[root@VM-1-50-tencentos ~]#
```

5. Execute a recovery action, and check the execution logs of the recovery action.

6. Check the result of recovery from a fault. Through comparison, we can see that the fault has been cleared and the occupied port 8080 has been released.

```
[root@VM-1-50-tencentos ~]# netstat -tuln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:5[redacted]      0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:3[redacted]      0.0.0.0:*               LISTEN
tcp6       0      0 :::5[redacted]           :::*                    LISTEN
tcp6       0      0 :::3[redacted]           :::*                    LISTEN
udp        0      0 127.0.0.1:[redacted]     0.0.0.0:*
udp6       0      0 ::1[redacted]            :::*
```

CVM Network Interruption

Last updated：2024-09-26 15:47:38

Background

CVM network interruption fault is a common issue, that hardware faults, improper network configuration, network equipment faults, network congestion, and other factors may cause. It will result in CVM network interruption during network communication and impact the normal operation of businesses. For businesses that rely on high availability and low delay, network interruption will cause immeasurable losses to users.
To improve network reliability and stability in CVM, network interruption experiments are required. Through the experiments, the capability of the system for normal operation in the situation of network interruption can be verified and issues caused by faults can be revealed in advance so that the system architecture can be optimized and contingency plans can be prepared.

Experiment Implementation

Step 1: Experiment Preparation

Prepare several CVM instances that are available for the experiment.

Step 2: Experiment Orchestration

1. Check network status before fault injection. Send messages to the target machine through ping commands to check network connectivity. If no response is received from the target address or there is a high packet loss rate, there may be an issue of network interruption.

```
[ro  tencentos ~]# ping 172.16.
PING 172.16.  56(84) bytes of data.
64 bytes from 172.16.  : icmp_seq=1 ttl=64 time=0.270 ms
64 bytes from 172.16.  : icmp_seq=2 ttl=64 time=0.205 ms
64 bytes from 172.16.  : icmp_seq=3 ttl=64 time=0.133 ms
64 bytes from 172.16.  : icmp_seq=4 ttl=64 time=0.147 ms
64 bytes from 172.16.  icmp_seq=5 ttl=64 time=0.126 ms
^C
--- 172.16  ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4109ms
rtt min/avg/max/mdev = 0.126/0.176/0.270/0.055 ms
```

2. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment**.

3. Click **Skip and create a blank experiment**. Fill in the experiment information, and select a CVM instance for the experiment objective.

4. Click **Add Now**, select **Network Resources**, click **Network interruption**, and click **Next**.

5. Configure fault action parameters, click **Confirm**. No parameter is required for the action, and configuration of general parameters is not required.

6. After action parameter configuration, click **Next**. Configure guardrail policies and monitoring metrics considering actual situations, and click **Submit** to complete experiment creation.

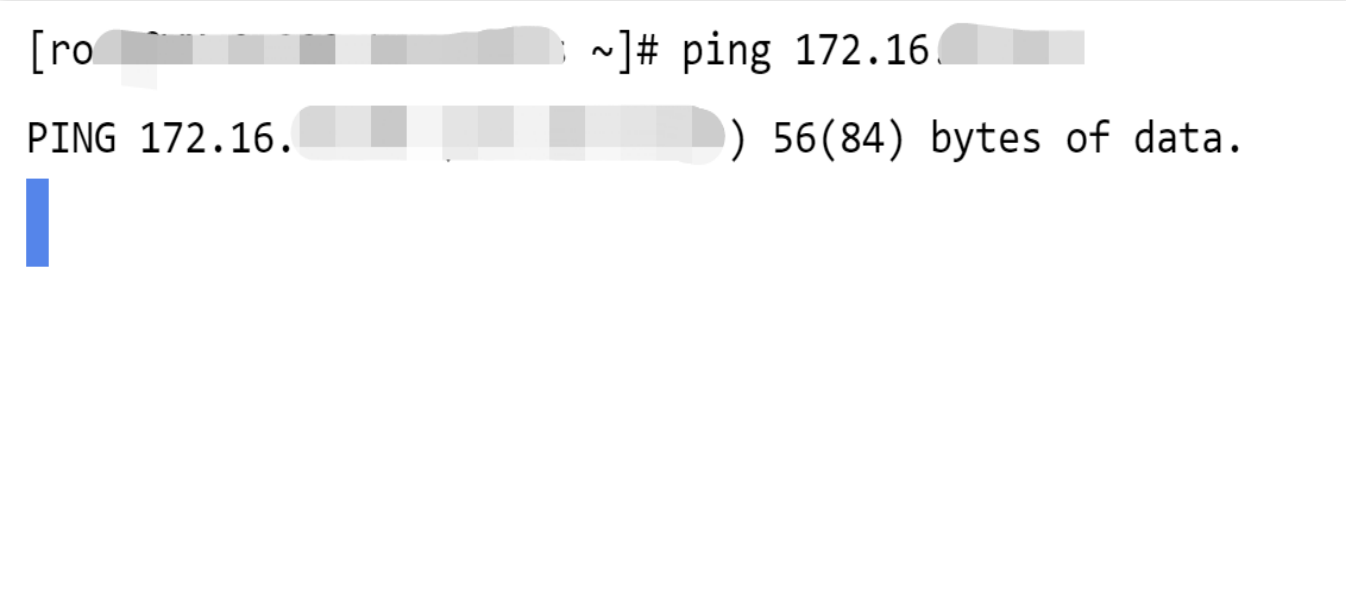
Step 3: Experiment Execution

1. Go to experiment details, and click **Go to the action group for execution**.

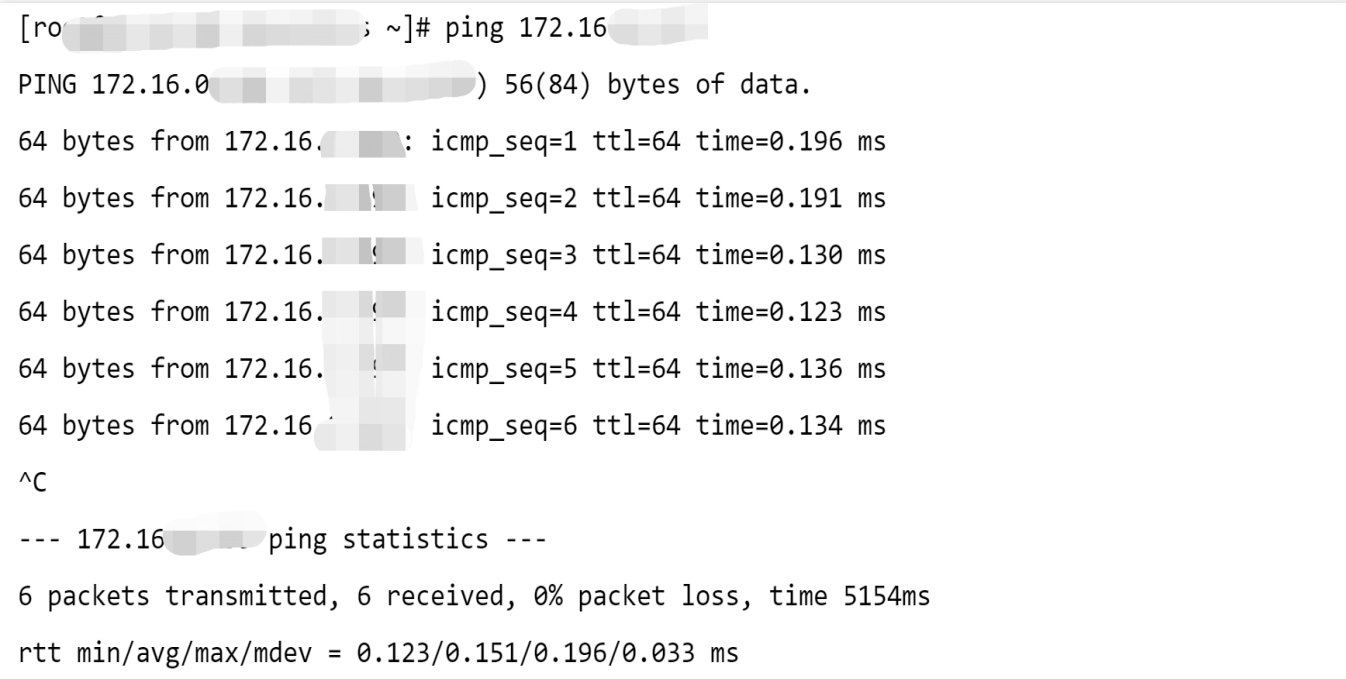
2. Click **Execute** to start an experiment.

3. Click the Action Card to check the details for the action execution results.

4. Check the network connection status of the target machine after fault injection. It can be seen that it is cannot be pinged.



5. Execute a recovery action, and check the execution logs of the recovery action.
6. Check the result of fault recovery. Log in to a machine and ping the target machine. Response messages show that the fault has been cleared and network transmission has recovered.



CVM Ping Unreachable

Last updated：2024-09-26 15:47:38

Background

CVM ping unreachable may be caused by hardware faults, improper network configuration, improper firewall settings, and routing issues. It will cause a failure in communication between CVM and other hosts or network equipment. For businesses that rely on this network connection, ping unreachable will cause great inconvenience and loss to users. To improve network reliability and stability in CVM, ping inaccessibility fault experiments are required. Through the experiments, the capability of the system for normal operation in the situation of ping unreachable can be verified and issues in this fault scene can be revealed in advance so that system architecture can be optimized and contingency plans can be prepared.

Experiment Implementation

Step 1: Experiment Preparations

Prepare several CVM instances that are available for the experiment.

Step 2: Experiment Orchestration

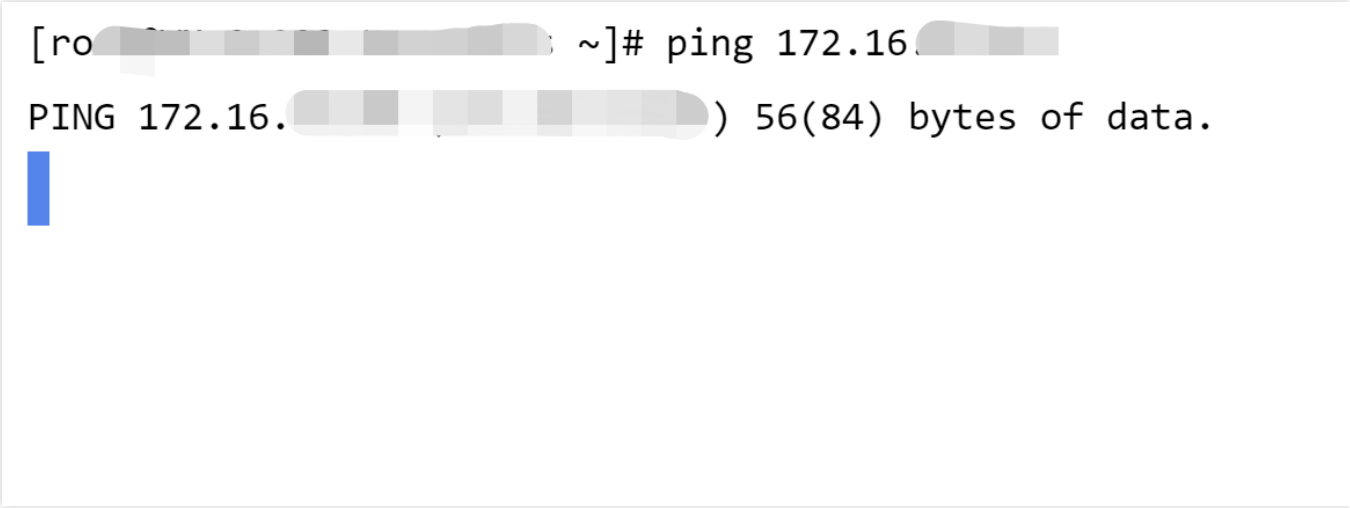
1. Check network status before fault injection. Send messages to the target machine through ping commands, and check network connectivity.

```
[ro  tencentos ~]# ping 172.16.
PING 172.16.  56(84) bytes of data.
64 bytes from 172.16.  : icmp_seq=1 ttl=64 time=0.270 ms
64 bytes from 172.16.  : icmp_seq=2 ttl=64 time=0.205 ms
64 bytes from 172.16.  : icmp_seq=3 ttl=64 time=0.133 ms
64 bytes from 172.16.  : icmp_seq=4 ttl=64 time=0.147 ms
64 bytes from 172.16.  icmp_seq=5 ttl=64 time=0.126 ms
^C
--- 172.16  ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4109ms
rtt min/avg/max/mdev = 0.126/0.176/0.270/0.055 ms
```

2. Click **Create a New Experiment**, fill in the experiment information, and add target CVM instances.
3. Click **Add Now**, select **Network Resources**, click **Ping unreachable**, and click **Next**.
4. Configure fault action parameters, click **Confirm**. No parameter is required for the action, and configuration of general parameters is not required.
5. After action parameter configuration, click **Next**. Configure guardrail policies and monitoring metrics considering actual situations, and click **Submit** to complete experiment creation.

Step 3: Experiment Execution

1. Go to experiment details, and click **Go to the action group for execution**.
2. Click **Execute** to start an experiment.
3. Click the Action Card to check the details for the action execution results.
4. Check the network connection status of the target machine after fault injection. It can be found that the target machine cannot be pinged.



5. Execute a recovery action, and check the execution logs of the recovery action.
6. Check the result of fault recovery. Log in to a machine and ping the target machine. Response messages show that the fault has been cleared and network transmission has recovered.

```
[ro                    ~]# ping 172.16
PING 172.16.0                ) 56(84) bytes of data.
64 bytes from 172.16.      : icmp_seq=1 ttl=64 time=0.196 ms
64 bytes from 172.16.      icmp_seq=2 ttl=64 time=0.191 ms
64 bytes from 172.16.      icmp_seq=3 ttl=64 time=0.130 ms
64 bytes from 172.16.      icmp_seq=4 ttl=64 time=0.123 ms
64 bytes from 172.16.      icmp_seq=5 ttl=64 time=0.136 ms
64 bytes from 172.16       icmp_seq=6 ttl=64 time=0.134 ms
^C
--- 172.16        ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5154ms
rtt min/avg/max/mdev = 0.123/0.151/0.196/0.033 ms
```

Cloud API Ban in CVM

Last updated：2024-09-26 15:47:38

Background

TencentCloud API 3.0 is an API gateway of Tencent Cloud and a channel for communication between clients and business services. At present, 400+ products and 20,000+ interfaces have been accessed, carrying tens of billions of calls in a day. Owning abundant tool ecosystems, a set of standard schemes for external APIs has been developed. As an important linkage for inter-service calls, it is featured with authentication, frequency limiting, audit, and security, and its stability has a direct effect on the reliability of cloud products and user experience.

Experiment Implementation

Step 1: Experiment Preparations

Prepare a CVM instance installed with TencentCloud Automation Tools (TAT).
Prepare a test script file, see CVM - Query Instance List interface.

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, click **Create a New Experiment**, and click **Skip and create a blank experiment.**
2. Fill in the basic information of the experiment.
3. Select Compute for the resource type, **CVM** for the resource object, and **Add Instance.**
4. Add an experiment action, click **Add Now**, and configure **Cloud API ban** fault action parameters.
5. Confirm experiment orchestration, and click **Submit** to complete experiment creation.

Step 3: Experiment Execution

1. Click **Execute**. An experiment is executed on **Cloud API ban** action.
2. Check the fault result, and you will find TencentCloud API request failed.

```
[root@VM-48-28-tencentos ~]# python3 api_ban.py
[TencentCloudSDKException] code:ClientNetworkError message:HTTPSConnectionPool(host='cvm.internal.tencentcloudapi.com', port=443): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.VerifiedHTTPSConnection object at 0x7f5844c8a6d8>: Failed to establish a new connectio
11] Connection refused',)) requestId:None
```

3. After the duration expires or a recovery action has been executed, normal access can be recovered.

```
[root@VM-48-28-tencentos ~]# python3 api_ban.py
[TencentCloudSDKException] code:ClientNetworkError message:HTTPSConnectionPool(host='cvm.internal.tencentcloudapi.com', port=443): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.VerifiedHTTPSConnection object at 0x7f5844
11] Connection refused',)) requestId:None
[root@VM-48-28-tencentos ~]# python3 api_ban.py
{"TotalCount": 11, "InstanceSet":
 "POSTPAID_BY_HOUR", "SystemDisk":
"BandwidthPackageId": null}, "Virt
```

Database
Primary-secondary Switch in TencentDB for PostgreSQL

Last updated：2024-09-26 15:47:38

Background

Tencent Smart Advisor-Chaotic Fault Generator provides fault actions for simulating a primary-secondary switch scene in TencentDB for PostgreSQL so that a scene in which primary-secondary switch occurs in PostgreSQL for some reason can be simulated. Primary-secondary switch experiment helps developers in system tests and experiments in a more complex and realistic environment so that possible problems and risks can be identified. Through Chaos Engineering experiments and tests, developers can have a more comprehensive understanding of system operating mode and performance characteristics and develop countermeasures and policies for different fault scenes to improve system stability and availability.

Note:

After a primary-secondary switch in PostgreSQL, replica machine rebuilding and migration will be started. The time required for replica machine readiness depends on data size. When an execution fails, retry the execution and do not execute the primary/replica switch experiment frequently. After a switch, it is advised that a second switch be executed after 10-20 minutes. Reference document of primary/replica switch instance in PostgreSQL .

Impact of Primary-secondary Switch

There will be momentary disconnections in primary-secondary switch. Make sure that the application has a reconnection mechanism.

If a read-only instance is mounted for primary instance, there is minute-level delay in read-only instance after primary-secondary switch.

Experiment Implementation

Step 1: Experiment Preparation

A PostgreSQL instance with cross-availability zone primary node and secondary node.

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment.**
2. Click **Skip and create a blank experiment** at the lower left quarter.
3. Fill in experiment information, select Object Type **PostgreSQL**, and click **Add Instance** to add instances for the experiment.
4. After selecting an instance, click **Add Now**in Experiment Action Module.
5. Add **Primary-secondary switch** fault action, and click **Next**.
6. Configure action parameters, close the forced switch at action parameters, and click **Confirm**.

Action Parameters Description:

Forced Switch: When it is enabled, conditions for the primary-secondary switch will not be checked and the primary-secondary switch will be performed directly; when it is closed, conditions for the primary-secondary switch must be checked before a primary-secondary switch is performed.

Conditions for primary-secondary switch: Go to PostgreSQL Console, go to the corresponding instance details page for check and editing in the availability information module.

7. Click **Next** to go to Global Configuration. See Quick Start for Global Configuration.
8. After confirmation, click **Submit.**
9. Click **Experiment Details** to go to Experiment Details page and start an experiment.

Step 3: Experiment Execution

1. Observe instance availability information data before an experiment, and pay attention to the primary node and secondary node of the instance before the experiment.
2. As the experiment is manually executed, fault actions must be executed manually. Click **Execute** in Action Card to start fault injection.
3. During a fault injection, the state of the instance observed in the basic information module of the instance corresponding to PostgreSQL Console is **Switching**.
4. After a successful fault injection, you can click blank area on the Action Card to check details of the action. It can seen that the primary node in the instance has been switched.
5. In Details page of an instance corresponding to PostgreSQL Console, a successful primary node switch can be confirmed.
6. The instance deployment state can be recovered to the state before the fault through fault recovery action. (It is advised that recovery action be executed after 10 minutes.)
7. After a successful fault recovery action, you can click the blank area on the Action Card to check the details of the action. You can see that the primary node of the instance has been switched.
8. In the Details page of an instance corresponding to PostgreSQL Console, you can confirm that the primary node has successfully switched to the pre-fault primary availability zone.

MySQL Instance Overall Unavailable

Last updated：2024-09-26 15:47:38

Background

Tencent Smart Advisor-Chaotic Fault Generator provides fault actions that simulate a scene in which MySQL instance is overall unavailable. You can verify disaster recovery and the overall high availability of your business MySQL through the fault action.

Note:

For MySQL instance creation and connection, see the following documents:

Creating MySQL Instance

Connecting MySQL Instance

Experiment Implementation

Step 1: Experiment Preparation

A MySQL Instance

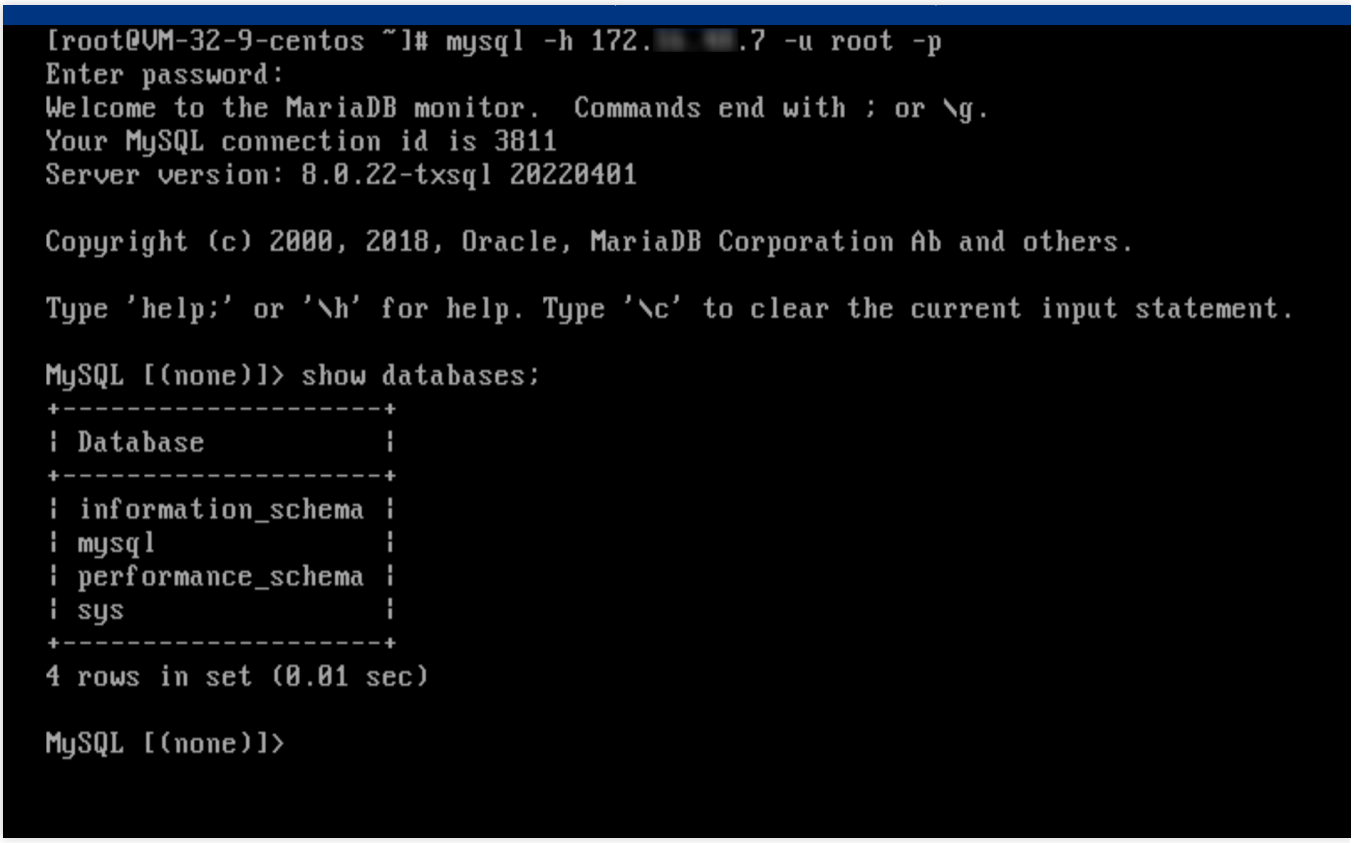
A CVM instance (used only for connecting database example. You can use other means of connection.)

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment.** Fill in the experiment information, and select TencentDB for MySQL instance for the Experiment Objective.
3. Click **Add Now** to add an experiment action, select **MySQL instance overall unavailable**, and click **Next** .
4. Configure action parameter information for fault actions and click **Confirm.**
5. After the configuration of action parameters, click **Next**. After confirming all configurations, click **Submit** to complete experiment creation.

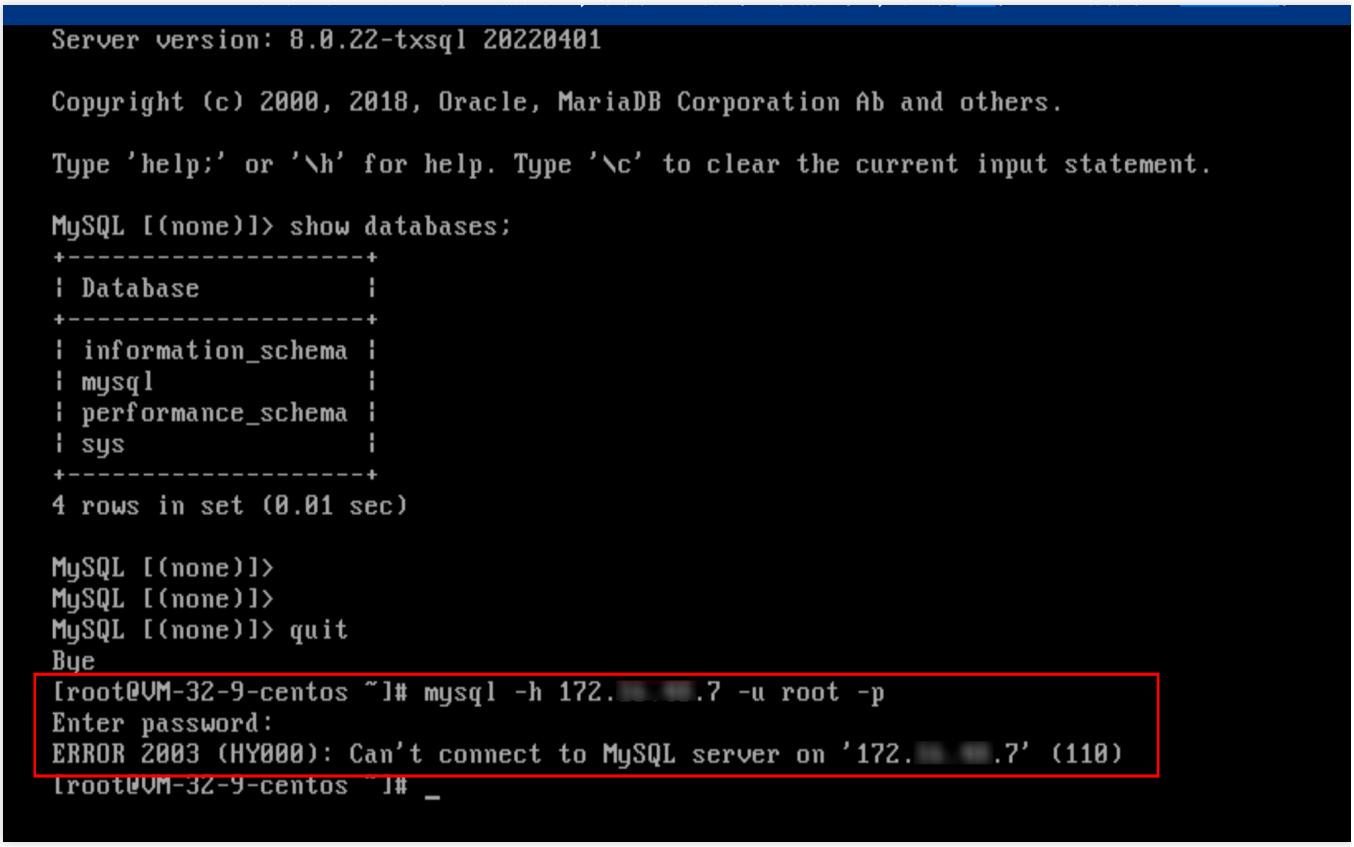
Step 3: Experiment Execution

1. Attempt to log in to MySQL instance through CVM.



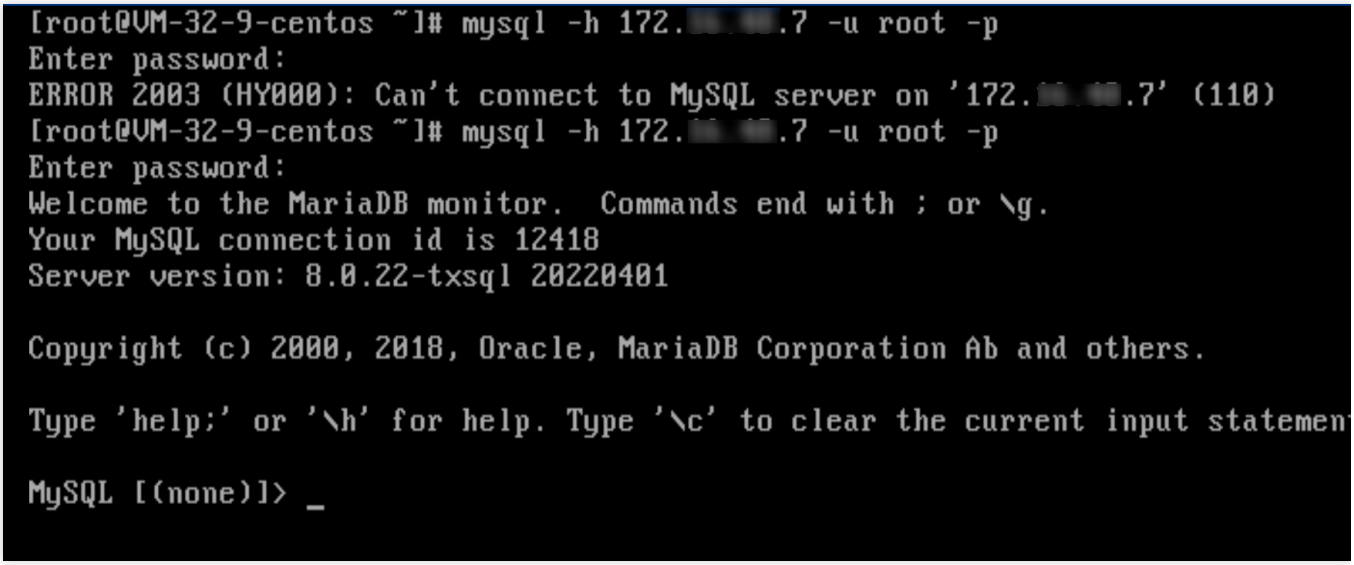
2. Go to Experiment Details. In the Experiment Action Group, click **Execute** to start executing an experiment.

3. After fault action injection, log in to TencentDB for MySQL instance through CVM once again. You can find that it's impossible to connect and log in normally.



4. Execute recovery actions.

5. Reconnect and log in to the instance.



Primary-secondary Switch in MySQL

Last updated：2024-09-26 15:47:38

Background

TencentDB for MySQL provides a dual-node instance with one primary and one secondary nodes and a three-node instance with one primary and two secondary nodes. To allow users to perform timely primary-secondary switch and normal services when a fault occurs in the primary instance, Tencent Smart Advisor-Chaotic Fault Generator provides users primary-secondary switch capabilities, supports users in manual primary-secondary switch, and helps users in verifying primary-secondary switch reliability, data integrity and overall stability of businesses.

Note:

This fault action requires that your TencentDB for MySQL instance be a multi-node architecture. If this condition is not satisfied, the fault action will not be executed. You can go to TencentDB for MySQL console to check instance architecture type and adjust instance architecture.

Experiment Implementation

Step 1: Experiment Preparation

A three-node MySQL instance cross different availability zones (with the primary nodes in ap-guangzhou-3 and secondary nodes in ap-guangzhou-6 and ap-guangzhou-3)

A CVM Instance

Step 2: Create an Experiment

1. Log in to Tencent Smart Advisor > Chaotic Fault Generator, create a new experiment. See Quick Start for details.
2. Fill in the basic information of an action group, select **MySQL** for Object Type, and click Add MySQL Instance.
3. Add **Switching primary-secondary roles of instance** action.
4. Configure action parameters, **Switch Mode** is **Prefer switching across availability zones**.

Note:

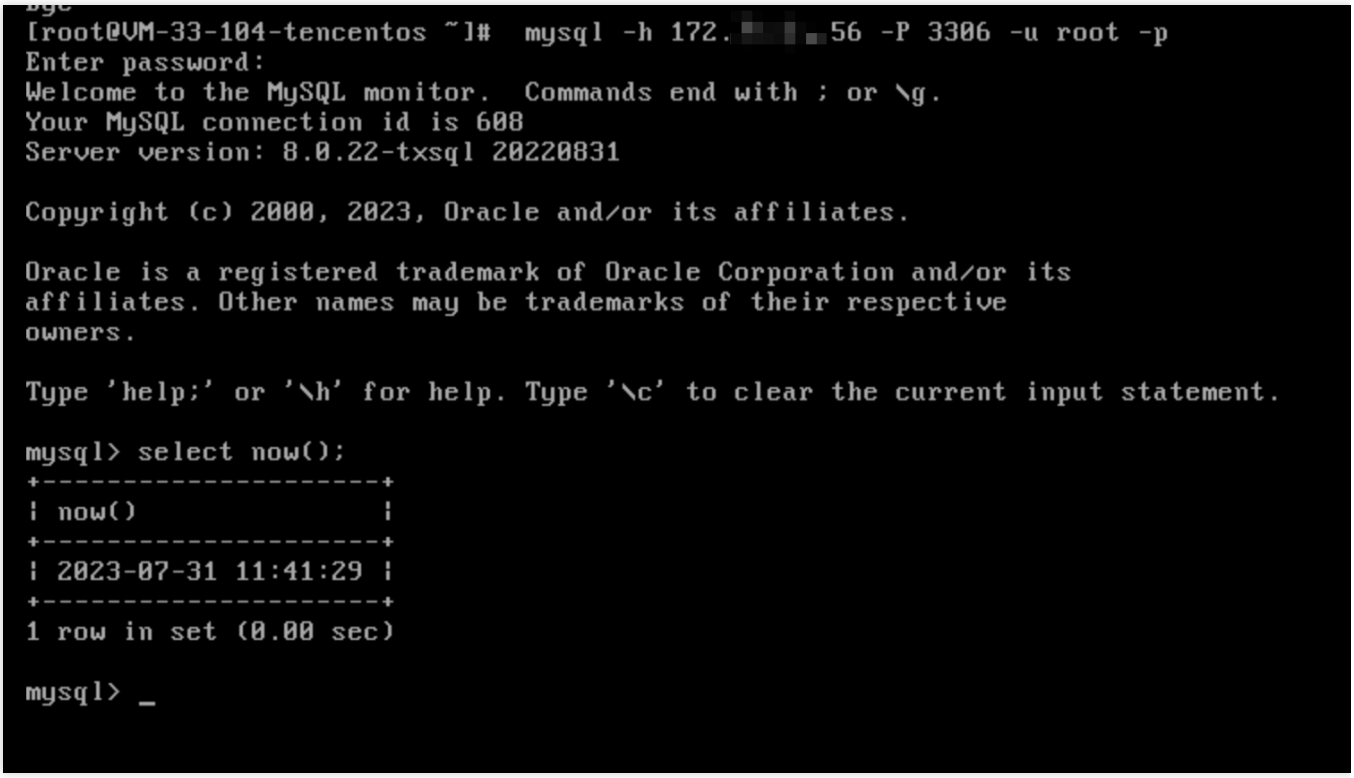
The platform provides two fault injection switch modes for primary-secondary switch actions: Prioritize Switch in the Same Availability Zone and Prioritize Cross-Availability Zone switch. Users can select a mode based on specific scenes. The platform recommends the following mode:

1. Primary Node Fault Scene: It is advised that priority be given to switch in the same availability zone. In simulating a primary node fault, select a secondary node in the same availability zone under a normal HA mechanism as the primary node, and the original primary node will become a secondary node.

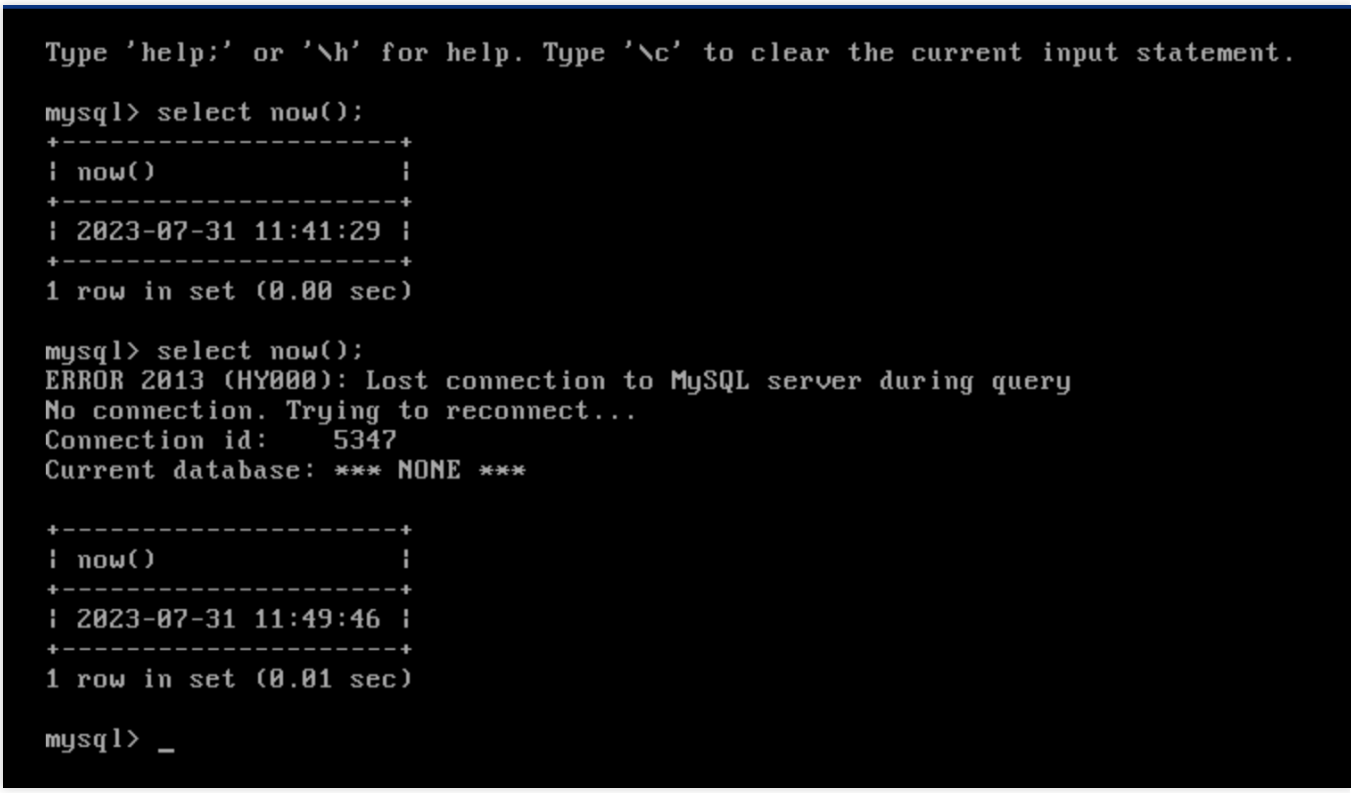
2. Power outage scene in an availability zone where the primary node is located: An instance is required for cross-availability zone deployment, it is advised that prioritized cross-availability zone switch action be used to simulate the primary node migrated to another availability zone due to primary node availability zone fault.
5. The platform has built-in Tencent Cloud Observability. Monitoring Metrics can be selected. After confirmation, click **Submit** to create an experiment.

Step 3: Experiment Execution

1. Connect MySQL instances through CVM:



2. In a new chaos engineering experiment, click **Execute** fault action.
3. Check action execution logs and instance status.
After the fault is completed, the primary node will be in ap-guangzhou-6, and the two secondary nodes will be in ap-guangzhou-3.
When a persistent connection is interrupted, client driver will automatically start reconnection when an inquiry is initiated once again.



4. Click **Execute** fault recovery action to recover initial node state.

Setting Maximum Number of Connections in MySQL

Last updated：2024-09-26 15:47:38

Background

MySQL is a widely used database management system. The maximum number of connections in the database directly impacts system performance and stability. If the maximum number of connections in MySQL is improperly set, the database may be unable to handle a large number of concurrent requests, therefore, the normal operation of the business is affected.
To improve database service reliability and stability, a fault experiment on the maximum number of connection is required. Through experiments, the system can be verified to see if it can operate normally under improper maximum connection settings, allowing for quick and effective fault resolution.
Note: For MySQL instance creation and connection, see the following documents.
Creating MySQL Instance
Connecting MySQL Instance

Experiment Implementation

Step 1: Experiment Preparation

A MySQL instance available for experiment
A CVM instance (used only for connecting database example. You can use other means of connection.)

Step 2: Experiment Orchestration

1. Check parameter configuration max_connections for number of connections in MySQL instance first, and pay attention to the initial value.
2. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment.**
3. Click **Skip and create a blank experiment**. Fill in the experiment information and add a ready MySQL instance.
4. Click **Add Now**, and select **Network Resources.**
5. Select **Set maximum connections**, and then click **Next**.
6. Configure fault action parameters, and click **Confirm**.
7. After action parameter configuration, click **Next**. Configure **Guardrail Policy** and **Monitoring Metrics** considering actual situations, and click **Submit** to complete experiment creation.

Step 3: Experiment Execution

1. Go to experiment details, and click **Go to the action group for execution**.
2. Click **Execute** to start an experiment.
3. Click the Action Card, and check details of action execution.
4. Check instance execution logs and confirm a successful execution.
5. Check the effects after the execution of the failure action, check max_connections parameter of MySQL instance, and you will find that a change has taken place.
6. Execute a recovery action, and check the execution logs of the recovery action.
7. Check the result after recovery. Check max_connections parameter of MySQL instance, and you will find that the parameter has been reset to the initial value.

Primary Node Fault Experiment on TencentDB for MySQL

Last updated：2024-09-26 15:47:38

Background

CFG provides a fault action for simulating primary node faults in TencentDB for MySQL. Disaster recovery and overall high availability of your business MySQL can be verified through the fault action.

Note:

This fault action requires that your TencentDB for MySQL instance be a multi-node architecture. If this condition is not satisfied, the fault action will not be executed. You can go to TencentDB for MySQL Console to check instance architecture type and adjust instance architecture.

Fault Description

This fault will inject a fatal error into the instance to simulate a primary node fault in a multi-node architecture instance. During fault action execution, temporary database disconnection or failure in connection may occur, which will result in the database access failure. Prudence is required during operation. After fault injection, MySQL instance will perform primary-secondary switch, the original secondary node will become a new primary node, and a new node will be selected from the original primary availability zone as a new secondary node.

Experiment Implementation

Step 1: Experiment Preparation

A TencentDB for MySQL instance with a dual-node architecture.
A CVM instance used for testing MySQL instance connection.

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment.**
2. Click **Skip and create a blank experiment.**
3. Fill in experiment information and select TencentDB for MySQL instance for Experiment Objective.
4. Add MySQL primary node fault experiment action. Click **Next**.

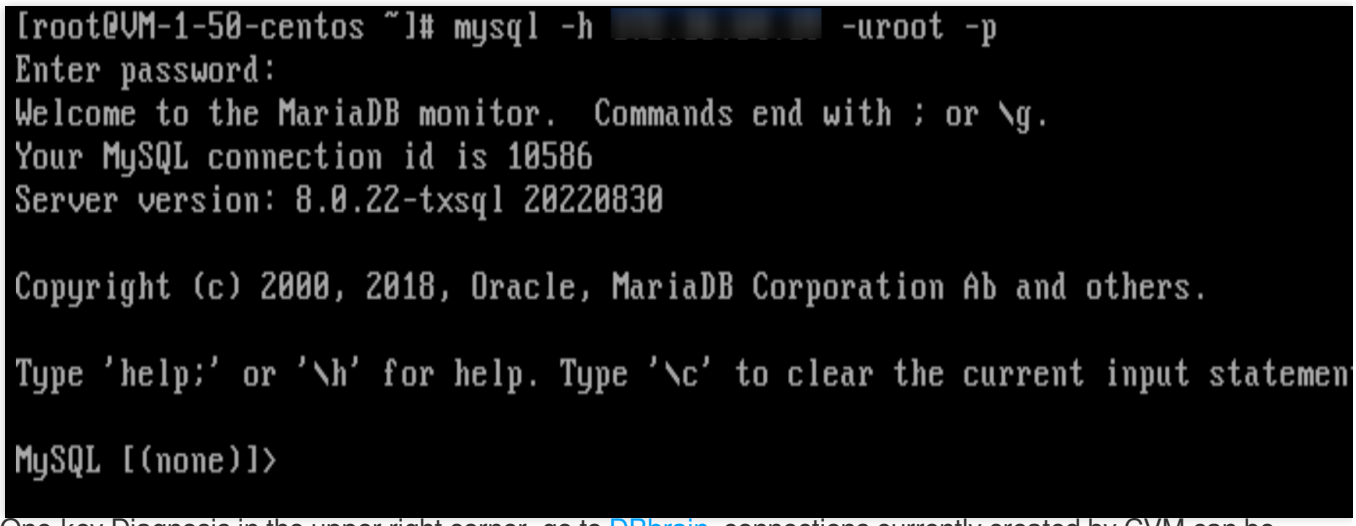
5. Configure action parameter information for fault actions and click **Confirm.**
6. After the configuration of action parameters, click **Next**. After confirming all configurations, click **Submit** to complete experiment creation.

Step 3: Experiment Execution

During fault execution, a primary node fault in MySQL instance will be triggered, and a primary-secondary switch will occur. An alarm can be received and changes in primary-secondary node architectures can be observed from TencentDB for MySQL Console.

Before a Fault Occrs:

Observe the availability zone where primary-secondary nodes of MySQL instance are located. Log in to current MySQL instance through CVM instance and create a client connection.



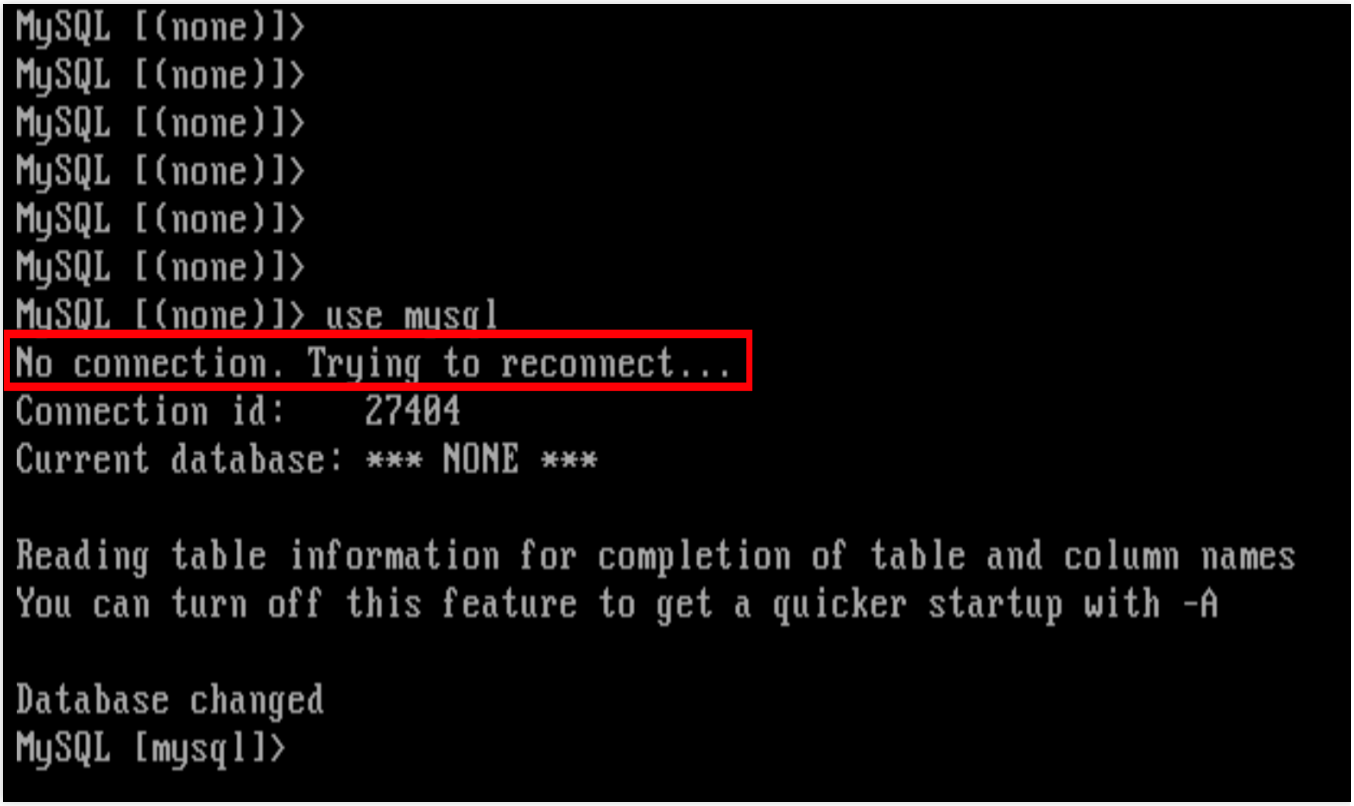
Click One-key Diagnosis in the upper right corner, go to DBbrain, connections currently created by CVM can be checked.

Execute an Experiment:

In Experiment Details page, click **Execute** in Experiment Action Group to start executing an experiment. After execution of a fault action, go to TencentDB for MySQL console to check alarms and node changes.
After a successful fault execution, you can check fault execution results through TencentDB for MySQL console. That is to say, the original replica node is promoted to primary node after the fault, and a new node is selected as replica node from the availability zone where the original primary node is located; and you can receive an abnormality alarm in the upper right corner (data is from TencentDB for DBbrain inspection).
And task details of the fault execution can be checked through TencentDB for MySQL console-Task List.

Monitor an Instance After a Fault Occurs

After a fault occurs in MySQL primary node, you can see through the instance monitoring panel that the current instance is temporarily unavailable when there is a fault in the primary node.
In such period, execute SQL through CVM, and the original connection has been interrupted and reconnection is required.



Meanwhile, the original connection monitored through DBbrain session no longer exists and a new connection appears.

TencentDB for MySQL Read-only Instance Group Unavailable

Last updated：2024-09-26 15:47:38

Background

Multiple read-only instance groups (RO groups) can be configured for TencentDB for MySQL instance. RO groups can handle read-only requests from primary instance and improve the overall read overall performance of MySQL. Tencent Smart Advisor-Chaotic Fault Generator provides a simulated fault scene in which MySQL read-only instance groups are unavailable. The scene can be used to verify the following scenes:
1. In the event of cross-availability zone RO groups, whether RO groups in another availability zone can provide normal services to the outside when the single cross-availability zone fails.
2. Impact of RO group instance fault on business (whether the primary node can withstand read pressure).
Note:
This fault action will perform an environmental check to check if a read-only instance is configured for the number of user security groups and MySQL instances. If your instance does not meet this condition, fault injection will fail. To create a read-only instance for a MySQL instance, refer to TencentDB for MySQL documentation Creating a Read-Only Instance.

Fault Description

This fault action will block RO group instances in the user-assigned injection range so that they are unavailable to the outside world, and existing connections in the instances in fault-related RO groups will be interrupted.

Experiment Implementation

Step 1: Experiment Preparations

A TencentDB for MySQL instance, for which a read-only instance has been configured.
A CVM instance for testing database connection. (Users can select other connections for verification tests)

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment**.

2. Go to Template Selection interface, click **Skip and create a blank experiment.**
3. Fill in the basic information of the experiment.
4. Fill in experiment action group information, select **Database** for **Instance Type**, MySQL for **Instance Object**, and click **Add Instance**.
5. Click **Add Now** to add a fault action, select **Read-only instance group unavailable** fault action, and click **Next**.
6. Fill in fault action parameters.
Now, a primary availability zone RO group is selected for the fault injection range so that the fault will be injected into an RO group instance of the selected MySQL instance in the primary instance availability zone.
Note:
Duration of Continuous Kill Connection (s) : Current action will create a continuous kill connection task to kill existing persistent connections. This parameter is used to specify the duration of a continuous kill task. If you have a large number of MySQL connections, this parameter can be increased so that all connections can be killed within the duration. The default time is 60s.
Fault Injection Scope: Select the availability zone where a fault is to be injected.
Primary availability zone: RO groups of the availability zone where the primary instance is located.
Non-primary availability zone: RO groups of the availability zone where other non-primary instances are located.
All availability zones: RO groups of all availability zones.
Assigned availability zones: RO group for availability zones assigned.
7. Click **Next**, go to Global Configuration. For Global Configuration, refer to Quick Start. After confirmation, click **Submit** to create an experiment.

Step 3: Experiment Execution

Before a Fault Occurs:
Before a MySQL RO group instance fault occurs, fault injection will be performed for selected read-only instance based on the injection range configured in previous step. Use CVM instance to log in to current RO instance, and create a client connection.

```
[root@VM-22-12-tencentos ~]# mysql -h 172.[redacted]7 -P 3306 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1097
Server version: 8.0.30-txsql 20221221

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Go to DBbrain to check current connections created by CVM.

Experiment Execution:

1. Go to Experiment Details, click **Execute** on Action Card, and start executing an experiment action.

After fault injection, click **Details** to check execution result information.

Now, it can be seen that existing connections have been interrupted.

```
[root@VM-22-12-tencentos ~]# mysql -h 172.[redacted].7 -P 3306 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1292
Server version: 8.0.30-txsql 20221221

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
ERROR 2013 (HY000): Lost connection to MySQL server during query
No connection. Trying to reconnect...
ERROR 2003 (HY000): Can't connect to MySQL server on '172.[redacted].7:3306' (110)
```

2. Click **Execute** a recovery action.

After a successful recovery action, reconnect the RO instance through CVM. After recovery, this RO instance can be reconnected, and the fault has been cleared.

```
[root@VM-22-12-tencentos ~]# mysql -h 172.[redacted].7 -P 3306 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1292
Server version: 8.0.30-txsql 20221221

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Primary-secondary Switch in TDSQL for MySQM

Last updated：2024-09-26 15:47:38

Background

In a database system, the primary-secondary switch is one of the important means for ensuring the high availability of a database. Primary-secondary switch allows a secondary node to take over the work of a primary node to ensure system continuity and stability upon failure of the primary node. Tencent Smart Advisor-Chaotic Fault Generator provides a fault action for simulating a scene of TDSQL for MySQL primary-secondary switch. You can verify the overall high availability of your business TDSQL for MySQL through this fault action.
Primary-secondary switch experiments are intended to help developers in system tests and experiments in a more complex and realistic environment so that possible problems and risks can be identified. Through experiments and tests in chaos engineering, developers can have a more comprehensive understanding of system operating modes and performance characteristics so that they can develop countermeasures and policies for different fault scenes to improving system stability and availability.
Note:
Primary-secondary switch will switch the primary node of the instance to another secondary node. It can be used to simulate a switch upon occurrence of an availability zone or node fault. Disconnection may occur during the switch.
There are two ways of fault injection:
1. Prioritize injection in the same availability zone: Secondary node in the same availability zone will be selected as the target node for switch. If there is no node satisfying the condition, switchable secondary node will be searched in other availability zones.
2. Prioritize cross-availability zone injection: Cross-availability zone secondary node across will be preferentially selected as the target node for switch. If there is no node satisfying the condition, switchable secondary node will be searched in the same availability zone.

Experiment Implementation

Step 1: Experiment Preparation

A TencentDB for TDSQL for MySQL instance, which has the deployment mode of one primary and two secondary nodes.

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment.**
2. Click **Skip** at the lower left quarter and create a new blank experiment.
3. Fill in the experiment information, and select Object Type TDSQL for MySQL.
4. Add an instance and click **Add Now** to add a fault action.
5. Select Primary-secondary switch fault, and configure fault action parameters.
Configure fault parameters. For an experiment instance, switch mode is configured as **Prefer switching across availability zones.**
6. After the action parameter configuration is completed, click **Next**. After all configurations are confirmed, click **Submit** to complete experiment creation.

Step 3: Experiment Execution

During fault execution, the TDSQL for MySQL instance primary-secondary switch will be triggered. Primary-secondary node architecture change in the instance can be observed through Cloud Database TDSQL Console.
Go to Experiment Details, click **Execute** to start executing an experiment.
During fault injection, you can observe node changes through Cloud Database TDSQL Console.
After a successful fault execution, primary node change can be observed through Cloud Database TDSQL Console.
The primary node has been switched from the original primary availability zone to the cross-availability zone secondary availability zone node.
Execute fault recovery action to recover the instance to the state before the fault.
Recovery from a fault will trigger another primary-secondary switch, and TDSQL console instance information will show the switch.
After a successful recovery, it can be seen through the console that the node has been recovered to the pre-fault state.

Primary-secondary Switch in MariaDB

Last updated：2024-09-26 15:47:38

Background

In a database system, primary-secondary switch is one of the important means to ensure the high availability of the database. Primary-secondary switch can ensure that when there is a fault in primary node, secondary node can quickly take over work of the primary node so that system continuity and stability are ensured. Tencent Smart Advisor-Chaotic Fault Generator provides a fault action for simulating a primary-secondary switch scene in TencentDB for MariaDB. Overall high availability of your business MariaDB can be verified through the fault action.
Primary-secondary switch experiments are intended to help developers in system tests and experiments in a more complex and realistic environment so that possible problems and risks can be identified. Through experiments and tests in chaos engineering, developers can have a more comprehensive understanding of system operating modes and performance characteristics so that they can develop countermeasures and policies for different fault scenes to improving system stability and availability.
Note:
Primary-secondary switch will switch the primary node of the instance to another secondary node. It can be used to simulate a switch upon occurrence of an availability zone or node fault. Disconnection may occur during the switch.
This fault can be injected in the following ways:
Prioritize injection in the same availability zone: Secondary node in the same availability zone will be selected as the target node for switch. If there is no node satisfying the condition, switchable secondary node will be searched in other availability zones.
Prioritize cross-availability zone injection: Cross-availability zone secondary node across will be preferentially selected as the target node for switch. If there is no node satisfying the condition, switchable secondary node will be searched in the same availability zone.
Note:
After initiating a task and a successful switch, you can observe the changes in System Monitoring-Primary-secondary Switch, and 1s momentary disconnection will occur. Make sure that there is a database reconnection mechanism for your business.
The system prioritizes data consistency, indicating that a switch may fail. Retry after at least 5 minutes as required.

Experiment Implementation

Step 1: Experiment Preparation

Prepare a TencentDB for MariaDB instance which has one primary and two secondary.

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaos Engineering Console, go to the **Experiment Management** page, and click **Create a New Experiment.**
2. Click **Skip and create a blank experiment at the lower left quarter.**
3. Fill in experiment information, select Object Type **Database > MariaDB**, and click **Add Instance** to add instances for the experiment.
4. Add an instance and click **Add Now** to add a fault action.
5. Select **Primary-secondary switch failure.**
6. Click **Next**, configure fault action parameter switch mode to **Prefer switching across availability zones**, and click **Confirm** .
7. After all configurations are confirmed, click **Next** and then click **submit** to complete experiment creation.

Step 3: Experiment Execution

During fault execution, the primary-secondary switch in MariaDB instance will be triggered. Change in primary-secondary node architecture in the instance can be observed through TencentDB for MariaDB console.

Note:

You can go to **monitoring and alarm** module of the instance corresponding to TencentDB for MariaDB console, and observe instance node ID and corresponding role. M is the primary node and S is the secondary node.

1. Go to **Experiment Details**, click **Execute** on Fault Action Card to start executing an experiment.
2. During fault injection, node changes can be observed through TencentDB for MariaDB console.
3. After a successful fault injection, click **Action Card** to check the details of the execution. You can see execution logs, and the primary node has been switched from the primary availability zone to secondary availability zone.
4. Go to TencentDB for MariaDB console to check node changes, and you will find that the switch from primary node has occurred.
5. Executing fault recovery action will trigger another primary-secondary switch and recover instance deployment to the state before the fault, that is, the primary node will be switched back to the original primary availability zone.
6. After a successful recovery, check execution logs, and primary node has been switched back to the original primary availability zone.
7. Go to TencentDB for MariaDB Control Panel to check instance information.

Primary-secondary Switch in TDSQL-C

Last updated：2024-09-26 15:47:38

Background

Tencent Smart Advisor-Chaotic Fault Generator provides fault actions for simulating a primary-secondary switch scene in the TencentDB for TDSQL-C to verify the overall high availability of TDSQL-C in your business. Primary-secondary switch experiments are intended to help developers in system tests and experiments in a more complex and realistic environment so that possible problems and risks can be identified. Through experiments and tests in chaos engineering, developers can have a more comprehensive understanding of system operating modes and performance characteristics so that they can develop countermeasures and policies for different fault scenes to improving system stability and availability.

Note:

This fault action only supports injection of primary-secondary switch faults into cross-availability zone instances. If an instance deployed in a single availability zone is selected, injection will fail and a prompt will be given in environmental check. Corresponding instances can be selected through TDSQL-C Console to adjust deployment mode.

Example of Instance Deployment Adjustment:

1. Log in to TDSQL-C Console, click Cluster List, and select the instance in a single AZ.
2. Click Edit Icon in **Deployment Method** in instance information.
3. Select Multiple Availability Zones (Yes), complete multi-availability zone configuration, and click **Confirm**.

Experiment Implementation

Step 1: Experiment Preparation

A cross-availability zone TDSQL-C instance. Pay attention to availability zones where primary and secondary nodes are located.

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment**.
3. Fill in experiment information and select Object Type **TDSQL-C**.
4. Add instances and add fault actions.
5. Add Primary-secondary switch experiment actions. No parameter is to be configured for this fault. Click **Next**.

6. Click **Next**, configure the Experiment Orchestration method and optionally configure Guardrail Policy Monitoring Metrics, and then click **Submit** to create an experiment.

Step 3: Experiment Execution

During fault execution, primary-secondary switch in TDSQL-C instance will be triggered. Changes in primary and secondary node architectures can be observed through Cloud Database TDSQL-C Console.
Go to Experiment Details, click **Execute** to start executing an experiment.
Execute fault actions. Changes in primary and secondary node architectures can be observed through Cloud Database TDSQL-C Console.
During fault injection, state of switch can be observed through TDSQL-C Console.
After a successful fault execution, you can observe changes in the primary node through Cloud Database TDSQL-C Console, and the primary node has been switched from the original primary availability zone to cross-availability zone replica availability zone node.
Execute fault recovery actions to recover instance deployment to the pre-fault state.
After a successful recovery, it can be seen through the console that the node has been recovered to the pre-fault state.

Practice of TencentDB for Redis Proxy Node Faults

Last updated：2024-09-26 15:47:38

Background

TencentDB for Redis allows automatic read-write separation through the Proxy layer. Proxy node can determine whether replica read-only mode is to be enabled. Proxy layer acts as a proxy for Redis sharded cluster. A fault in Proxy layer will have direct impact on access to data in TencentDB for Redis. Users can use the TencentDB for Redis Proxy node fault action provided by the platform to check if a fault in Proxy layer node will have impact on their businesses.

Experiment Implementation

Step 1: Experiment Instance Preparation

Select one or more TencentDB for Redis instances as instance objects of an experiment.

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment**. Fill in experiment information, select **Redis Memory Edition** under Cloud Resources for Resource Object, add target instance.
3. Click **Add Now** to add an experiment action, select **Redis Proxy node failure** , and click **Next**.
4. Fill in corresponding action parameters, and click **Confirm**.

Note:

There are two action parameters required for this action: fault injection availability zone and fault node proportion (%).

Fault Injection Availability Zone:

Primary Availability Zone: This means that a Proxy node where a fault is to be injected will be selected from primary availability zone.

Secondary Availability Zone: This means that a Proxy node where a fault is to be injected will be selected from secondary availability zone.

All Availability Zones: This means that a Proxy node where a fault is to be injected will be selected from all availability zones.

Faulty Node Ratio (%) : This parameter is an integer ranging 1~100, indicating that a certain number of proxy nodes will be selected for fault injection.
For example: Parameters for a certain experiment can be so configured that fault is injected into 50% of Proxy nodes in the primary availability zone. If the total number of proxy nodes in the primary availability zone in the instance is five, 5 * 50% = 3 (rounded up) nodes will be randomly selected from the 5 proxy nodes in the primary availability zone for fault injection.
5. After the configuration of action parameters, click **Next**. After confirming all configurations, click **Submit** to complete experiment creation.

Step 3: Experiment Execution

1. Initial state of a TencentDB for Redis Proxy node. You can go to TencentDB for Redis console, click corresponding instances and select **System Monitoring** > **Monitoring Metrics** > **Instance Monitoring** > **Proxy Monitoring**.
2. To execute an experiment, click Action Card and observe the results of execution.
It can seen that a fault has been injected into the specified node and the node has crashed. And then there will be a new Proxy node.

```
[2023-02-02 10:23:10]: General Parameters Configuration:
[2023-02-02 10:23:10]:  Lead waiting time (s): 0
[2023-02-02 10:23:10]:  Action timeout (s): 1800
[2023-02-02 10:23:10]:  Post waiting time (s): 0
[2023-02-02 10:23:10]: Custom parameters configuration:
[2023-02-02 10:23:10]:  Fault node proportion (%): 60
[2023-02-02 10:23:10]:  Fault injection availability zone: primary availability
zone
[2023-02-02 10:23:10]: Start execution [Redis Proxy Node Fault] action,
instances involved: ['crs-xxxxxxx']
[2023-02-02 10:23:11]: [Redis Proxy node fault] action has been executed, and
monitoring on returned results of action is started.
[2023-02-02 10:23:14]: [Redis Proxy Node Fault] action call [Successful],
details:
 Instance: [crs-xxxxxxx] Execution result: [Successful] Details: Instance: crs-
xxxxxxxx, fault injection into CRS Proxy node is successful, injected Proxy
node object ID = ['d06d2ffb1b600ffa877137c1c5c5265221cd55fa',
'1ce5fec58407759a9ff04c30c1a9e485911ac25d']
```

TencentDB for Redis Primary Node Fault

Last updated：2024-09-26 15:47:38

Background

TencentDB for Redis supports a primary-secondary architecture. Users can simulate a scene of a primary node fault in TencentDB for Redis by means of Redis primary node fault action provided by the platform.

Experiment Implementation

Step 1: Experiment Preparation

Prepare a few TencentDB for Redis instances.

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment.**
2. Click **Skip and create a blank experiment.**
3. Fill in the basic information of the experiment, and click **Next**.
4. Add an Action Group. Fill in the basic information of an action group, and click **Add Instance.**
5. Select the Redis instance for the experiment, click **Confirm**. If there is no instance, click **Create Instance**, and continue experiment creation after purchasing an instance.
6. Click **Add Now**, select a fault action.
7. Select **Redis master node failure**, click **Next**.
8. Configure action parameters considering actual situations, click **Confirm**.
9. After action parameter configuration, click **Next**.
10. Selectively configure **Guardrail Policy** and **Monitoring Metrics** considering actual situations. After checking and confirming all configurations, click **Submit** to complete the experiment creation.

Step 3: Experiment Execution

1. Check node status and primary and replica availability zones in TencentDB for Redis before an experiment. Click **Instance ID** in **Instance List** in **Experiment Details** to go to instance details.
2. Go to experiment details, and click **Go to Action Group Execution**.
3. Click **Execute** to start an experiment. During the experiment, click **Instance ID** to skip to the instance details page where instance node information can be checked.

Primary-secondary Nodes in TencentDB for Redis Instance Unavailable

Last updated：2024-09-26 15:47:38

Background

TencentDB for Redis supports multi-availability zone deployment. Users can deploy their businesses in multiple availability zones to improve the high availability of systems. Users can use primary-secondary nodes unavailability fault in TencentDB for Redis provided by Tencent Smart Advisor-Chaotic Fault Generator to simulate such scenes so that disaster recovery and overall high availability of their business systems can be verified.

Experiment Implementation

Step 1: Experiment Preparation

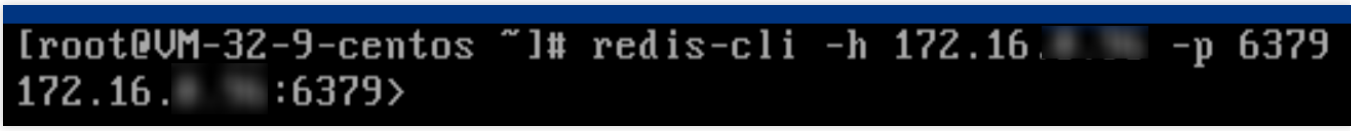
A TencentDB for Redis Instance
A CVM instance connected to Redis instance network (used for logging in a CRS instance).
Note: For creating and logging in a CloudDB instance, see the following documents:
Creating a TencentDB for Redis Instance
Logging in to a TencentDB for Redis Instance

Step 2: Experiment Orchestration

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment**, fill in experiment information, select TencentDB for Redis Memory Edition under Cloud Resource Type for Experiment Resource Object, and add an instance.
3. Click **Add Now** to add an experiment action, select **Primary-secondary node unavailable** and then click **Next**.
4. Configure action parameter information for fault actions and click **Confirm.**
5. After action parameter configuration, click **Next**. Selectively configure Guardrail Policy and Monitoring Metrics considering actual situations. After checking and confirming all configurations, click **Submit** to complete the experiment creation.

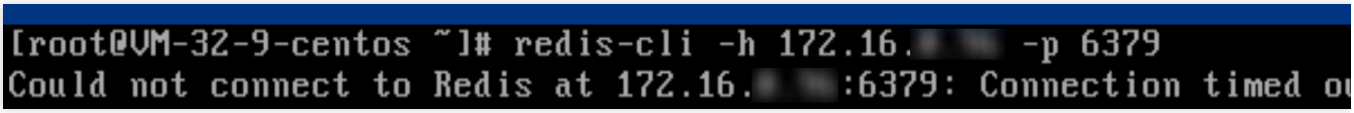
Step 3: Experiment Execution

1. Try to log in to a TencentDB for Redis instance through CVM.

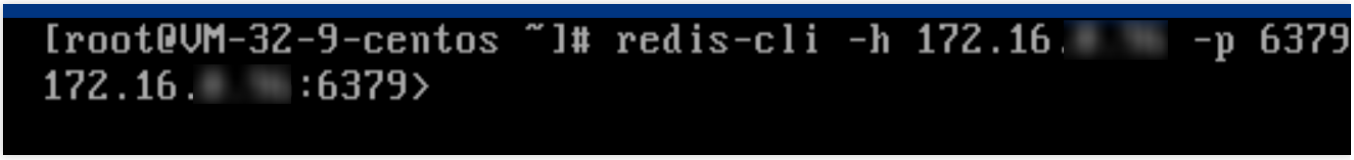


2. Go to Experiment Details. In the Experiment Action Group, click **Execute** to start executing an experiment.

3. After fault injection, log in to a TencentDB for Redis instance through CVM once again.



4. Execute a recovery action and log in to TencentDB for Redis through CVM once again. Redis connection is recovered.



Simulating Primary-secondary Switch in Redis

Last updated：2024-09-26 15:47:38

Background

Redis cluster is an important component for storing hot data of businesses. To ensure business availability, Gossip protocol between nodes in the cluster is used to determine node status. Heartbeat timeout (cluster-node-timeout) is 15s by default. If the fault node is the primary node, Tencent Cloud Redis will apply a failover mechanism and select a new primary node from the secondary node.
Based on the above features, Tencent Smart Advisor-Chaotic Fault Generator provides a manual method to skip node fault stage and directly simulate fault actions for HA policy. You can simulate the impact on businesses in the short period when failover occurs in Redis cluster by manual fault method.

Experiment Implementation

Experiment Preparation

Prepare a multi-node cross-availability zone Redis instance.

Experiment Steps

Step 1: Create an Experiment

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment** .
3. Fill in basic information**.** For Experiment Resource Object, select TencentDB for Redis memory edition under Cloud Resource Type, and add an instance.

Step 2: Add Actions

1. Click **Add Now** to add a fault action. Select Redis primary-secondary switch in Redis for fault action.

2. In action parameters setting, flexibly select primary/replica switch mode based on a simulated disaster recovery scene:

Prefer switching within the same availability zone

Simulate the real HA policy scene of Tencent Cloud Redis when the primary node is faulty: the latest data node is uplifted to the primary node; when data is identical, priority is given to other nodes in the same availability zone.

Prefer switching within the same availability zone

When an entire availability zone is faulty, nodes in another availability zone will be raised for the primary scene.

Step 3: Execute Experiment Actions

Go to Experiment Details. In the Experiment Action Group, click **Execute** to start executing an experiment.

Check Results

Take cross-availability zone mode as an example, check whether the availability zone state has changed before and after fault injection.

Simulating MongoDB Storage Node Fault

Last updated：2024-09-26 15:47:38

Background

MongoDB instances provide multiple data storage nodes and HA mechanisms for users to ensure data security and high availability of services. To verify usage and disaster recovery of TencentDB for MongoDB in businesses of users, Tencent Smart Advisor-Chaotic Fault Generator provides multiple fault scenes to simulate storage node faults.

Experiment Implementation

Experiment Preparation

Purchase cross-availability zone cloud MongoDB instances, deploy a local or cloud server-side test environment, and connect MongoDB instances.
Scripts for simulating common client requests.

```
#!/usr/bin/python
"""
Through simple ways of data reading and writing, the operation of a data volume of
This script is for reference only. In real experiments, it is advised that faults b
"""
import pymongo
import random

# During an experiment, use the MongoDB instance Uri where a fault is injected for
mongodbUri = 'mongodb://mongouser:thepasswordA1@ip:port/admin'
client = pymongo.MongoClient(mongodbUri)
# Database assignment
db = client.somedb
# Data sets deletion
db.user.drop()
# Customize the size of data to be inserted. It is advised that a larger data size
element_num = 3 * 10 ** 6
for id in range(element_num):
    # Insert random documents
    name = random.choice(['R9', 'caÒt', 'owen', 'lee', 'J'])
    sex = random.choice(['male', 'female'])
    try:
        db.user.insert_one({'id': id, 'name': name, 'sex': sex})
    except Exception as e:
```

```
        print('error id', id)
# Query full-load documents
content = db.user.find()
for i in content:
    print(i)
```

Experiment Steps

Step 1: Create an Experiment

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment.**
2. Click **Skip and create a blank experiment**.

Step 2: Add MongoDB Instances and Actions

1. In the experiment object configuration, you can enter or select VPC filtering and add MongoDB instances for the experiment through batch instance ID.
2. Add experiment actions.

MongoDB master node restart: Simulate the impact of MongoDB primary node restart on businesses and the HA mechanism for MongoDB.

MongoDB primary-secondary switch: Simulate the process of primary-secondary nodes switch in MongoDB and the scene in which node IP changes after the switch. The action supports two execution modes: prioritizing switch in the same availability zone and prioritizing cross-availability zone switch.

Step 3: Add Monitoring Metrics as Required

Step 4: Go to Experiment Details and Execute an Experiment

Check Results

Simulate user behaviors in a production environment using the script, and observe response at three time states: before, during and after a fault.

Primary Node Restart

Before a Fault

Business Performance: Data of a large size is continuously inserted into the database, and MongoDB processes the data at a stable rate.

Instance State: Primary nodes, secondary nodes and hidden nodes have been configured and taken effect.

During a Fault

Business Performance: When insertion into the script fails, the insertion can be retried through a configuration-driven retry mechanism, Mongo node abnormality will be detected, and temporary data backlog will occur.
Instance State: Select a node from secondary nodes as the primary node. As no node is selected from hidden nodes, the original secondary node will be selected as the primary node, and the original primary node will be regarded as the secondary node.
After a Fault
Business Performance: When backlog data and current data yet to be inserted are inserted simultaneously, MongoDB processes the data at nearly twice the rate.
Instance State: MongoDB automatically recovers primary-secondary nodes relation and selects the node with higher weight as the primary node. When the weights are identical, they will remain unchanged.
Conclusion
After completion, check that database data is normal, businesses are insensitive to faults, and the overall situation is as expected.

Primary-secondary Switch

Before a Fault
Business Performance: Data of a large size is continuously inserted into the database, and MongoDB processes the data at a stable rate.
Instance State: Primary nodes, secondary nodes and hidden nodes have been configured and taken effect.
During a Fault
Business Performance: Availability zone switch occurs in the node, and a situation similar to node restart occurs in the business, resulting in short-term data backlog.
Instance State: Based on the action mode, a node in the same availability zone or cross-availability zone node is preferentially selected from secondary nodes in the instance and promoted to the primary node. The original primary node is re-started as a secondary node.
After a Fault
Business Performance: After an availability zone switch, business operations are normal, and data insertion and data inquiry actions are unaffected.
Manual Recovery from a Fault
Instance State: Execute a fault recovery action to recover the state before the fault.
Conclusion
Business is insensitive to faults, data in the database is intact, and overall performance is normal and as expected.

Simulating Self-Built MySQL Crash Through Network Blocking

Last updated：2024-09-26 15:47:38

Background

In migration to cloud platforms in the early stage, cost will be considered and some middleware will be built as production environment by reusing CVM. But with increase of business size, in subsequent businesses, Self-Built middleware will be gradually replaced with cloud middleware so that the high availability of the middleware can be improved. For a Self-Built middleware, its high availability can be verified by injecting a network blocking fault.

Experiment Objectives

Verify the high availability of Self-Built MySQL.

Architecture

Use Keepalived+MySQL form to achieve multi-site high availability of MySQL.

Create an Experiment

Step 1: Confirm Fault Injection Objectives

Select injection objective as Master based on the architecture, check if switch to Slave is allowed when Master is confronted with network blocking, and if business insensitivity to faults is achieved.

Step 2: Create an Experiment

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment**, and fill in the basic information of the experiment.
3. Configure Experiment Object, select CVM under host type for Experiment Resource Object, and add an instance. Select **Intra-host network packet loss** for experiment actions.

Configure action parameters in the Action Parameters page, because the target port for this experiment is 3306 and the private network interface is eth0. To accurately block a specified port, specified parameters must be configured. After configuration, click **Confirm** to go to the next step.

4. Configure Monitoring Metrics (optional): Some basic CVM metrics can be configured to allow metrics selection.
5. Complete experiment tasks creation.

Start an Experiment

1. Check if businesses are at a normal state before starting. Try connection to check that the Master node is normal and can be connected.
2. In an Experiment Action Group, click **Execute** to execute an experiment.
3. After fault injection, check if business connection is normal.
4. Execute fault recovery actions to recover the state before the fault.

Restarting TencentDB for MySQL

Last updated：2024-09-26 15:49:18

Fault Description

This fault action will restart the TencentDB for the MySQL instance specified by the user to simulate fault scenes such as momentary disconnection of database services.

Impact of Faults

1. During a restart, the instance will be unable to properly provide services, and existing connections may be interrupted. Make preparations to avoid impact.
2. During a restart, existing connections may occur flash interruptions.
3. During a restart, instance change and other operations are not allowed.
4. During a restart, if the size of the business write-in is large, there will be a lot of dirty pages. To reduce the duration of service unavailability, restart may fail.
5. During a read-only instance restart, weights and elimination policies will be invalid. Please stay tuned.

Experiment Implementation

Step 1: Experiment Preparation

A TencentDB for MySQL Instance.

Step 2: Experiment Orchestration

1. Log in to Tencent Cloud Smart Advisor > Chaotic Fault Generator, go to Experiment Management page, and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment** at the lower left quarter.
3. Fill in experiment information, select a TencentDB for MySQL instance for the experiment objective, and click **Add Now** to add fault actions.
4. Select **MySQL restart** fault action, and click **Next**.
5. There is no fault parameter to be filled in for this fault action. Click **Confirm**, and then click **Next**.
6. Submit to create an experiment.

Step 3: Experiment Execution

1. Go to Experiment Details, click **Execute** to start executing an experiment.
2. After a fault occurs, click **Details** to check execution results. Restart ID, asynchronous request ID, and estimated restart duration can be obtained from the log. You can click Task Details List link in the log to go to TencentDB for MySQL Task List and check task details.
3. After a successful fault injection, Instance Monitoring Metrics can be observed through TencentDB for MySQL Console.

Task Details: Restart task initiated can be checked in the Task List. For more information about the execution, click Task Details on the right side.

Instance Monitoring: During a restart, the fracture can be seen in Monitoring Metrics such as CPU.

Primary-secondary Switch in SQL Server

Last updated：2024-09-26 15:47:38

Background

TencentDB for SQL Server provides an instance with dual nodes: a primary node and a secondary node. To allow a timely primary-secondary switch by the user upon occurrence of a primary instance fault and normal services, Tencent Smart Advisor-Chaotic Fault Generator provides users capabilities for primary-secondary switch, supports manual primary-secondary switch for users, and helps users in verifying primary-secondary switch reliability, data integrity and overall stability of services.

Note:

This action requires that your TencentDB for SQL Server instance meets the following conditions:

Must be a dual-node cloud disk architecture.

Primary and secondary node instances must be in operation.

Primary and secondary node instances must be non-read-only instances.

Read-only instance not associated.

Publish and subscribe not enabled.

Contains at least one self-built database.

Does not contain automatically generated test database test19700101.

Experiment Implementation

Step 1: Experiment Preparation

A dual-node TencentDB for SQL Server.

Step 2: Create an Experiment

1. Log in to Tencent Cloud Smart Advisor > Chaos Engineering Console, create a new experiment. See Quick Start for details.
2. Fill in the basic information of an action group, select **Database > SQL Server** for Object Type, and click Add SQL Server Instance.
3. Click **Add Instance** to add instances for the experiment.
4. Click **Add Now** to add **Primary-secondary switch** fault Action, click **Next**.
5. Configure action parameters and click **Confirm**.
6. The platform has built-in Tencent Cloud Observability. Monitoring Metrics can be selected. After confirmation, click **Submit** to create an experiment.

Step 3: Experiment Execution

1. In a new Chaotic Fault Generator, click **Execute** fault action.
2. Click to execute card, and check action execution logs.
3. Click to execute the fault recovery action to recover the initial node state.

Network
VPC Subnet Network Isolation

Last updated：2024-09-26 15:47:38

Background

In **intra-city double site active-active** or **cross-region multi-site active-active** disaster recovery scenarios, disaster recovery resources are typically deployed in different availability zones or regions from the main resources, each belonging to different subnets. When a fault occurs at the availability zone or regional level, a disaster recovery switch can be initiated. To verify the effectiveness of your disaster recovery architecture, you can use the CFG VPC Subnet Network Isolation action to block the subnet where the main resources are located, simulating the inaccessibility of resources due to a fault.

Fault Effect

The VPC subnet isolation fault provides two ways of network isolation: Single Subnet and All Subnets.

Single Subnet

When the fault action parameter **Isolation Scope** is set to **Single Subnet**, the inbound and outbound traffic of the selected subnet will be blocked, and access between subnets will also be restricted. For example, as shown in the figure below, when the **Single Subnet** network isolation fault is injected into Subnet A and Subnet B, **all the inbound and outbound traffic of Subnet A and Subnet B will be blocked. Subnet A cannot access Subnet B or other subnets, but internal traffic within each subnet will not be affected**.

All Subnets

When the fault action parameter **Isolation Scope** selects **All Subnets**, the inbound and outbound traffic of the selected subnets will be blocked, but access between the selected subnets will not be affected. For instance, as shown in the figure below, when the **All Subnets** network isolation fault is injected into Subnet A and Subnet B, **internal traffic within Subnet A and Subnet B remains unaffected, but traffic attempting to access other subnets will be blocked**. You can inject faults into subnets within the same availability zone to simulate a network isolation fault between that availability zone and other availability zones.

Experiment Preparation

Create two subnets under the same VPC, and associate private network CLB, CVM, and MySQL resources with these subnets. The network topology is as follows: the primary availability zone is associated with two CVMs, one private network gateway, and one MySQL instance, while the replica availability zone is associated with one CVM and one private network gateway. Since instances under the same VPC are interconnected via the network by default, resources between subnets can access each other before fault injection. **When the primary availability zone subnet is blocked, instances within the primary availability zone subnet can still access each other normally, but external access to instances in the primary availability zone subnet will fail.**

Note:
Subnet network isolation is achieved by setting network ACL rules on the subnet, and any existing persistent connections will be immediately disconnected.
If the target subnet has existing network ACL rules, those rules will be temporarily unbound during fault injection and re-bound during recovery. Do not manually modify or delete network ACL rules during the experiment.
Subnet network isolation cannot be used to simulate a database single availability zone fault. For database single availability zone fault, see Database-Related Failures action.
When the Isolation Scope is set to All Subnets, a maximum of 20 subnets can be added. When you need to isolate all subnets for more than 20 subnets, please contact Tencent Cloud Assistant. Submit a Ticket.
This fault action does not block external Ping detection to CLBs within the subnet.

Experiment Implementation

Step 1: Create an Experiment

1. Log in to the Tencent Cloud Smart Advisor > Chaotic Fault Generator Console.
2. In the left sidebar, select the **Experiment Management** page, click **Create a New Experiment**, and select to create a blank experiment.
3. After filling in the basic information, enter the Experiment Object Configuration. Select the **VPC Subnet** under **Network** for the object type, and click **Add Instance**.
Click **Add Instance**, and a list of all VPC subnet information in the target region will be displayed. Filter the subnets based on subnet ID, VPC instance ID, tags, and availability zone keys.
Note:
The subnet network isolation has a wide impact range. Please carefully select the scope of the fault injection instances.
4. After you select the target subnets, click **Add Now** to add the experiment actions.
Select **Network isolation** for the experiment action, and then click **Next**.

Set the **Isolation Scope** parameter and click **Confirm**.
5. After you click **Confirm**, you can see that the fault action will automatically bring up the recovery action.
6. Click **Next** to enter the Global Configuration. For global configuration details, see Quick Start. Note that subnets do not have corresponding basic resource monitoring metrics. After you confirm, click **Submit**, and the platform will automatically perform an environment pre-check.

Step 2: Execute the Experiment

In the Experiment Action Group, click **Execute** to execute an experiment. Since the experiment is manual, manually execution of the fault action is needed.

Step 3: Verify the Injection Effect

In the **Virtual Private Cloud** console, under the **Network Topology Map** menu, you can see that the **Network ACL** rules have been added to the corresponding subnets.

For testing instance access, it is expected that access between instances in the subnets within the faulty availability zone will not be affected, but access from outside the subnet to instances in the faulty subnet fails.
CVM accessing CVM
Normal access within the same subnet
Failure when accessing different subnets (command blocked)

CVM accessing MySQL
Normal access within the same subnet
Failure access within different subnets
CVM accessing CLB
Since CLB uses VPCGW for Ping detection, the subnet network isolation will not block Ping detection. You can use telnet to detect the service-side ports.
Normal access within the same subnet

Failure when accessing different subnets (command blocked)

Step 4: Execute Fault Recovery Actions

Click Execute Recovery Action and wait for the action to be executed successfully.

Step 5: Verify the Recovery Effect

See **Step 3** for verification. The expected result is that instances within the subnet and between subnets should be accessed normally.

NAT Gateway Fault Experiment Case

Last updated：2024-09-26 15:47:38

Background

Network Address Translation (NAT) gateway is a crucial network service that performs IP address translation between a virtual private cloud (such as a VPC) and a public network (such as the Internet). A NAT gateway allows resources in a virtual private cloud to access the public network through a single public IP address while ensuring the security and isolation of the virtual private cloud. When a NAT gateway encounters a fault, it may prevent resources in the virtual private cloud from accessing the public network, impacting the normal operation of business. NAT gateway faults can be caused by misconfigurations, network issues, and hardware faults.
To enhance the reliability and stability of NAT gateways, it is necessary to conduct NAT gateway fault experiments. Through experiments, we can verify whether the system can operate normally in NAT gateway fault scenarios and expose potential issues in advance, allowing for system architecture optimization and emergency planning.

Experiment Execution

Step 1: Experiment Preparation

Log in to NAT Gateway and create a gateway service. If there is already a gateway service available for the experiment, proceed directly to create the experiment.

Step 2: Create an Experiment

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, go to the **Experiment Management** page, and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment**, and fill in the basic details.
3. Select **Network** as the instance type and **NAT Gateway** as the instance object, then **Add Instance.**
4. Click **Add Now** to add fault action.
5. Select the fault action.
6. Set action parameters and click **OK**.

7. After action parameter configuration, click **Next**. Configure **Guardrail Policy** and **Monitoring Metrics** considering actual situations, click **Submit** to complete experiment creation.

Step 3: Execute the Experiment

1. View the performance metrics of the NAT Gateway instance before executing the fault.
2. Go to experiment details, and click **Go to the action group for execution**.
3. Click **Execute** to start an experiment.
4. View the details of the action execution results.
5. View the execution logs to confirm it has been executed successfully.
6. After executing the fault, view the NAT Gateway instance's performance metrics again. You can see that the maximum concurrent connections have been updated to the value specified in the action execution parameters, indicating that the fault injection was successful.
7. Execute the fault recovery actions, view the execution logs, and confirm that the recovery actions were successful.
8. After fault recovery, view the NAT Gateway instance's performance metrics once more. You can see that the maximum concurrent connections have been recovered to the initial value, indicating that the fault recovery was successful.

Container
Simulating Container Resource Network Faults

Last updated：2024-09-26 15:47:38

Background

Network faults are among the most common issues in Kubernetes, and troubleshooting them can be challenging, often requiring intervention from specialized Ops personnel. To address this, the CFG provides five types of network fault actions: network disorder, network duplication, network package loss, network latency, and network DNS anomalies. These actions allow you to independently stabilize network performance in your business and ensure the normal operation of services.

Experiment Implementation

Experiment Preparation

Purchase container instances in a standard cluster and deploy test services.
Enter Agent Management page and install the agents.

Experiment Steps

Step 1: Create an Experiment

1. Log in to Tencent Smart Advisor > Chaotic Fault Generator, enter the **Experiment Management** page, click **Create a New Experiment.**
2. Click **Skip and create a blank experiment**, and fill in the experiment details.

Step 2: Add Experiment Instances and Actions

Node Type Network Faults

Note:

Applicable resource objects: Regular node in standard cluster.

1. During the experiment object configuration, select the container object type **Standard Cluster Node**.
2. Add a Regular Node in Standard Cluster node instance. Select the **Cluster ID** where the fault will be injected, and the node under that cluster will be automatically added.
3. Add experiment actions. Select Network latency fault action under network resources.
4. Configure the action parameters.

Pod Type Network Faults

Note:

Applicable resource objects: Standard cluster Pod and Serverless cluster Pod.

1. Select **Standard Cluster Pod** as the container object type.
2. Add a Pod instance, then **sequentially** select the required **Cluster ID**> **Load Type** (Workload) > **Namespace**> **Load Name** (Workload) to select a Pod instance that meets the criteria.
3. Select Network latency fault action under network resources.
4. Configure fault action parameters. On this page, you can configure whether to enable Container faults. If it is enabled, after you enter the container name, the fault can be injected into the specified container.

Step 3: Configure Monitoring (Optional)

In the Global Configuration stage, you can select monitoring metrics, check the necessary ones, and add them to the configuration.

Step 4: Execute the Experiment

Click **Execute** Experiment. Since network actions are high-risk, authentication must be completed before executing the action.

Experiment on Container Resource Pod Operation Faults

Last updated：2024-09-26 15:47:38

Background

Pods are the smallest deployable units in a Kubernetes cluster that run applications. In practical use cases, Pods may encounter issues such as container crashes, resource shortages, Pod termination, and Pod failures, which can prevent applications from running properly.
To enhance the reliability and stability of container services, it is necessary to conduct Pod fault experiments. These experiments help verify whether the system can continue to operate normally in the event of a Pod fault, and they allow for early identification of potential issues in such fault scenarios, enabling quick and effective resolution.

Experiment Execution

Step 1: Experiment Preparation

Purchase container instances and deploy test services. If there is already a container instance available for the experiment, proceed directly to create the experiment.
Enter Agent Management page and install the agents.

Step 2: Create an Experiment

1. Log in Tencent Smart Advisor > Chaotic Fault Generator, go to **Experiment Management** page, and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment.**
3. Select **Container** as the instance type, select **Standard Cluster Pod** as the instance object, and then **Add Instance**.
4. Add a fault action. Click **Add Now** and select **Pod Operation**.
5. Select the fault action **Pod deletion**.
6. Set the action parameters and click **Confirm**.

7. After action parameter configuration, click **Next**. Configure **Guardrail Policy** and **Monitoring Metrics** considering actual situations, click **Submit** to complete experiment creation.

Step 3: Execute the Experiment

1. View the Pod's information before executing the fault, focusing on the Pod's creation time and running time.
2. Go to experiment details, click **Go to the action group for execution**.
3. Click **Execute** to start an experiment.
4. View the details of the action execution results.
5. View the execution logs to confirm it has been executed successfully.
6. Verify the execution effect. View the Pod's information after the fault execution. Based on the Pod's creation time and running time, you can observe that a new Pod has been created, indicating that the previous Pod was deleted and a new Pod has been recreated.

Experiment on Container Resource Node Faults

Last updated：2024-09-26 15:47:38

Background

Container nodes (such as worker nodes in a Kubernetes cluster) host container resources and are responsible for running and managing container instances. However, container nodes may encounter hardware faults, resource shortages, network faults, etc., which could lead to container instances not operating correctly.
To enhance the reliability and stability of container services, node fault experiments are needed. Through these experiments, you can verify whether the system can operate normally in the event of container node faults, and uncover potential issues in advance for system architecture optimization and emergency planning.

Experiment Execution

Step 1: Experiment Preparation

Create a container node, add instances, and deploy the test service. If there is already a container node available for the experiment, proceed directly to create the experiment.
Enter Agent Management page and install the agents.

Step 2: Create an Experiment

1. Log in Tencent Smart Advisor > Chaotic Fault Generator, go to **Experiment Management** page, and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment**, and fill in the experiment details.
3. Select **Container** as the instance type, and select **Standard Cluster Node** as the instance object, then click **Add Instance.**
4. Click **Add Now** to add fault action.
5. Select the fault action **Node Operation - Node Shutdown**.
6. Set action parameters and click **Confirm**.

7. After action parameter configuration, click **Next**. Configure **Guardrail Policy** and **Monitoring Metrics** considering actual situations, click **Submit** to complete experiment creation.

Step 3: Execute the Experiment

1. View the node status before executing the fault.
2. Go to experiment details, click **Go to the action group for execution**.
3. Click **Execute** to start an experiment.
4. Click the Action Card, and check details of action execution.
5. View the execution logs to confirm it has been executed successfully.
6. View the node status after the fault execution. You can see that the node is in an abnormal status now. It indicates that the fault injection was successful, and the Pods under the cluster node are also running abnormally.
7. Execute the recovery actions, view the execution logs, and confirm that the recovery actions were successful.
8. After successful execution of the fault recovery action, view the status of the cluster node. You can see that the node is operating normally, and the Pods under the cluster node are also functioning properly, indicating that the fault has been successfully resolved.

Experiment on Container Resource Application Process Faults

Last updated：2024-09-26 15:47:38

Background

Container resources provide a lightweight, portable, and scalable runtime environment for applications. However, application processes within containers may encounter faults such as crashes, deadlocks, or resource leaks, causing the application to malfunction.
To enhance the reliability and stability of container services, application process fault experiments are necessary. These experiments help verify whether the system can operate normally in the event of process faults, uncovering potential issues in advance for system architecture optimization and emergency planning.

Experiment Execution

Note:
Applicable resource objects: Regular node in standard cluster, standard cluster Pod, and Serverless cluster Pod.

Step 1: Experiment Preparation

Purchase container instances and deploy test services. If there is already a container instance available for the experiment, proceed directly to create the experiment.
Enter Agent Management page and install the agents.

Step 2: Create an Experiment

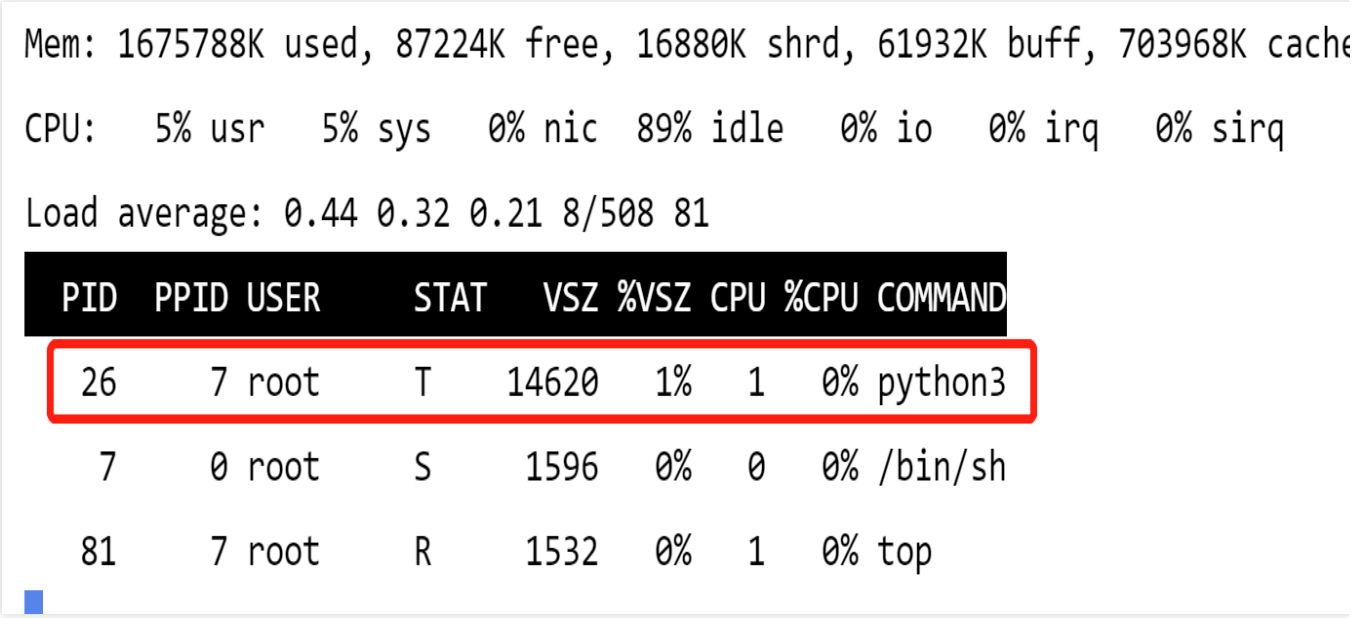
1. Log in to Tencent Smart Advisor ＞ Chaotic Fault Generator, and enter the **Experiment Management** page, click **Create a New Experiment**.
2. Click **Skip and create a blank experiment**, and fill in the experiment details.
3. Select **Container** as the instance type, select **Standard Cluster Pod** as the instance object, and then **Add Instance.**

4. Click **Add Now** to add a fault action and select **Application Process.**
5. Select the fault action **Process Stop**, then click **Next**.
6. Set action parameters and click **Confirm**.

All containers: The objective process in every container will be stopped.
Select the first container alphabetically: The objective process in the first container will be stopped.
Specify container name: The objective process in the specified container will be stopped.
7. After action parameter configuration, click **Next**. Configure **Guardrail Policy** and **Monitoring Metrics** considering actual situations, click Submit to complete experiment creation.

Step 3: Execute the Experiment

1. Log in to the machine where the fault will be executed and view the current process management details. You can see a running Python process.



2. Go to experiment details, click **Go to the action group for execution**.
3. Click **Execute** to start an experiment.
4. Click the Action Card, and check details of action execution.
5. View the execution logs to confirm it has been executed successfully.
6. View the effects after the fault execution. View the current process management details again, and you can see that the Python process has been terminated.

```
Mem: 1594128K used, 168884K free, 16880K shrd, 53952K buff, 647364K cache
CPU:   6% usr   2% sys   0% nic  90% idle   0% io   0% irq   0% sirq
Load average: 0.16 0.24 0.19 3/487 102
  PID  PPID USER     STAT   VSZ %VSZ CPU %CPU COMMAND
    7     0 root     S     1596   0%   0   0% /bin/sh
   81     7 root     R     1532   0%   1   0% top
```

Standard Cluster and Serverless Cluster Super Node Faults

Last updated：2024-09-26 15:47:38

Background

Super nodes in container services, compared to regular nodes, support rapid auto scaling. They consolidate what would have been multiple nodes into a single node for management, simplifying the administration of container resources. However, container nodes may encounter hardware faults, resource shortages, network faults, etc., which could lead to container instances not operating correctly.
Through these experiments, you can verify whether the system can operate normally in the event of container node faults, and uncover potential issues in advance for system architecture optimization and emergency planning. Through these experiments, you can verify whether the system can operate normally in the event of container node faults, and uncover potential issues in advance for system architecture optimization and emergency planning.

Experiment Execution

Step 1: Experiment Preparation

Purchase standard cluster container instances and super nodes, and deploy test services.
Purchase Serverless cluster container instances with built-in super nodes and deploy test services.

Step 2: Create an Experiment

1. Log in to Tencent Smart Advisor > Chaotic Fault Generator, enter the **Experiment Management** page, and click **Create a New Experiment**.
2. Create a blank experiment. Click **Skip and create a blank experiment**.
3. Fill in the basic information for the experiment, add related resource tags as needed, then click **Next**.
4. Create two action groups, select **Container** as the Resource Type, and add **Standard Cluster Super Node** and **Serverless Cluster Super Node** Resource Objects respectively.
5. Find the experiment instances and add them to the action groups.

6. Click **Add Now** to add experiment actions, then select fault actions: **Node lockdown and Node drain (Node Eviction).**
7. For global configuration, confirm the Experiment Action Group, select the Experiment Execution Method and configure the Guardrail Policies.
8. Click **Submit** to complete the experiment creation.

Step 3: Execute the Experiment

1. Go to experiment details, click **Go to Action Group Execution**.
2. Execute the actions in the sequential execution:
3. Execute **Node lockdown** and **Node lockdown Recovery**.
3.1 Execute the Node Block Failure Action, view the execution logs on the action card, and observe the node status to confirm it is locked and no longer schedulable.
3.2 Execute the Node Block Fault Recovery Action to unlock the node and observe its status.
4. Execute **Node Drain** and **Node Drain Recovery**.
4.1 View the current Pod list on the node, check the high availability policies for the services, and ensure that there is sufficient capacity on other nodes to restart the Pods after eviction.
4.2 Execute the Node Drain Failure Action, which will evict resources from the node. The action card will show the evicted Pods, making it easier to observe the affected services. The super node will be locked and not schedulable. At the same time, through cluster scheduling, the evicted Pods are reconstructed on other nodes to recover services.
4.3 Execute fault recovery. Unblock the node and recover its schedulable status.

Serverless Pod Fault Experiment Case

Last updated：2024-09-26 15:47:38

Background

Pods are the smallest deployable units in a Kubernetes cluster that run applications. In practical use cases, Pods may encounter issues such as container crashes, resource shortages, Pod termination, and Pod downtime, which can prevent applications from running properly.
To enhance the reliability and stability of container services, it is necessary to conduct Serverless Pod fault experiments. These experiments help verify whether the service can continue to provide service normally in the event of a Pod fault, and they allow for early identification of potential issues in such fault scenarios, ultimately improving business stability.

Experiment Execution

Step 1: Experiment Preparation

Purchase container instances and deploy test services. If there is already a container instance available for the experiment, proceed directly to create the experiment.
Enter Agent Management page and install the agents.

Step 2: Create an Experiment

1. Log in Tencent Smart Advisor > Chaotic Fault Generator, go to **Experiment Management** page, and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment**
3. Fill in basic information of the experiment.
4. Select **Container** as the resource type, select **Serverless Cluster Pod** as the resource object, and then **Add Instance.**
5. Click **Add Now** to add fault action. Select **Pod Operation.**
6. Select the fault action **Pod Failure**, then click **Next**.
7. Set action parameters and click **Confirm**.

8. After action parameter configuration, click **Next**. Configure **Guardrail Policy** and **Monitoring Metrics** considering actual situations, click **Submit** to complete experiment creation.

Step 3: Execute the Experiment

1. Go to experiment details, and click **Go to the action group for execution**.
2. Click **Execute** to start an experiment.
3. View the action execution results.
4. Click the action card and view the execution logs to confirm it has been executed successfully.

Cluster Node Resource (CPU, Memory, Disk) Stress Test Faults

Last updated：2024-09-26 15:47:38

Background

Container nodes (such as worker nodes in a Kubernetes cluster) host container resources and are responsible for running and managing container instances. When the QPS of a node business increases suddenly or the service memory leaks, the node resource utilization may increase, thus affecting the business or even causing business processes to be killed.
To enhance the reliability and stability of container services, node fault experiments are needed. Through these experiments, you can verify whether the system can operate normally in the event of container node faults, and uncover potential issues in advance for system architecture optimization and emergency planning.

Experiment Execution

Step 1: Experiment Preparation

Create a container node, add instances, and deploy the test service. If there is already a container node available for the experiment, proceed directly to create the experiment.
Go to the agent management page, and install an agent for the CVM node. Please see Agent Management for installation.

Step 2: Create an Experiment

1. Log in to Tencent Smart Advisor > Chaotic Fault Generator, and enter the **Experiment Management** page, click **Create a New Experiment**.
2. Go to Template Selection interface, click **Skip and create a blank experiment** .
3. Fill in the experiment name and description, then click **Next**.
4. Fill in the action group information. Select the resource type as **Container** and select the resource object as **Standard Cluster Node.**
5. Click **Add Instance**, and select the instances to be included in the experiment from the instance list.
6. In the experiment actions section, click **Add Now** to add experiment actions. Select actions for **CPU Resources**, **Memory Resources**, and **Disk Resources**.
7. Modify the action parameters as needed.
High CPU utilization

Note:

CPU Utilization: Specify CPU load percentage, which is 0 to 100.

Duration: Duration of a fault action, upon lapse of which, the agent will automatically recover the fault.

Scheduling Priority: It affects process priority in CPU scheduling. A lower nice value makes it more likely that the process would have a CPU time slice so that its execution priority can be improved. It is effective only if utilization is 100%.

High memory utilization

Note:

Memory Usage Rate: Specify a memory load percentage that is 0 to 100.

Duration: Duration of a fault action, upon lapse of which, the agent will automatically recover the fault.

Enable OOM Protection: If it is enabled, the possibility of fault process OOM-KILL will be reduced, and business processes will be killed first.

Memory Occupancy Rate: Memory usage increase per second.

High disk usage

Note:

Action Parameters Description:

Disk Directory: A disk directory to be populated, i.e., a directory where files are written.

File Size: Size of a file populated.

Disk Usage Rate: Learn disk usage through staf commands, and calculate the file size required for specified utilization.

Reserved Space: Size of remaining space.

Duration: Duration of a fault action, upon lapse of which, the agent will automatically recover the fault.

If there are file size, disk utilization, and reserved space parameters, the priority calculation logic is disk utilization > reserved space > file size.

Disk IO load

Note:

Disk Directory: Specify a directory to enhance disk IO, which will apply to the disk it resides on.

Mode: Provide both read and write modes to execute high loads.

Block Size: Specify block size for every read or write.

Number of Blocks: Specify number of blocks to be copied.

Duration: Duration of a fault action, upon lapse of which, the agent will automatically recover the fault.

8. Complete the action group edit and click **Next**.

9. Click **Add Monitoring Metrics.**

10. Click **Submit** to complete the creation of the experiment.

Step 3: Execute the Experiment

1. View the node agent status through pre-check and install the nodes that have not been installed according to the pre-check result prompts.

2. Click **Execute** to start injecting the high CPU load fault.
3. Observe the node monitoring metrics to ensure CPU utilization reaches the preset value and that recovery is completed by the specified time.
4. Execute the high memory utilization action and observe the monitoring metrics.
5. Execute the high disk utilization action and use the **df Command** in the terminal to view utilization.

```
[root@VM-0-211-tencentos ~]# df
Filesystem     1K-blocks     Used Available Use% Mounted on
devtmpfs         3858596        0   3858596   0% /dev
tmpfs            3875780    10268   3865512   1% /dev/shm
tmpfs            3875780     3372   3872408   1% /run
tmpfs            3875780        0   3875780   0% /sys/fs/cgroup
/dev/vda1       51539404 39014948  10306196  80% /
```

6. Execute the high disk I/O load action and use the **iostat Command** in the terminal to observe.

```
04/07/2024 03:25:12 PM
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           1.39    0.00    1.89   13.60    0.00   83.12

Device            r/s     w/s     rMB/s     wMB/s   rrqm/s   wrqm/s  %rrqm  %wrqm r_await w_await aqu-sz rareq-sz wareq-sz  svctm  %
vda          2373.50    2.00     23.04      0.03     0.00     0.50   0.00  20.00    0.42    0.50   0.10     9.94    13.00   0.42  9
```

Note:
Use iostat -x to view detailed information about the running status of block devices. This action primarily focuses on the **%util** parameter, which indicates the percentage of time the device spends processing I/O requests. When this value approaches 100%, it means the device's bandwidth utilization is nearly at maximum capacity, leading to a decline in overall disk performance and severely affecting the processing of other read/write requests.

Serverless Pod Virtual Node Shutdown Faults

Last updated：2024-09-26 15:47:38

Background

The TKE Serverless clusters, along with the traditional clusters, provide super nodes capabilities to enhance the efficiency of dynamic resource scaling and reduce resource usage costs. The cluster delivers computing resources in the form of Pods, eliminating the need for users to manage the actual nodes running behind the Pods. Using super nodes is similar to using a very large specification CVM, facilitating resource management and scaling-in.
The super node itself is just a logical concept. When Pods are run on super nodes, the cluster dynamically creates a temporary virtual node exclusive to the Pod, which is terminated when the Pod is deleted. When a virtual node fault is detected, the Pod will perform disaster recovery drift, rebuild the virtual node, and rebuild the Pod. You can deploy super nodes across different availability zones to achieve resource dispersion and mitigate disaster risks due to availability zone faults. The CFG provides a Serverless Pod virtual node shutdown fault scenario, which can help you verify the impact of Pod disaster recovery drift on your services after a virtual node fails. Additionally, it can simulate the impact of single availability zone faults on Pods scheduled to super nodes to verify your disaster recovery design's effectiveness.

Note:

Injecting this fault scenario on Pods not scheduled to super nodes will fail.
The Kubernetes clusters have some self-healing capabilities, but during large-scale faults, such as sudden abnormalities in over 50% of nodes or 50% of Serverless containers, this self-healing ability may retreat or even break. This is to prevent large-scale eviction or traffic removal actions that could lead to greater fault risks, such as a cluster-wide avalanche effect. **It is recommended that you do not exceed half of the total Serverless Pods in the cluster in a single-fault experiment**.
The fault determination time is affected by the timeout configuration on the Agent within the instance (The default value is 5 minutes). If you need to promptly remove abnormal Pod traffic, see Automatic Rebuilding and Self-Healing for configuration.

Fault Parameters Description

Duration (s) : The maximum time allowed for a fault. The default value is 30 minutes. After the fault injection is successful, the fault will automatically recover after the configured duration (even if manual recovery actions are not executed).
Whether to auto-Rebuild Pod or not: Whether to enable the Pod's automatic rebuilding feature, which is enabled by default. It is used to simulate persistent faults that cannot self-heal. For details, please see Automatic Rebuilding

and Self-Healing.

Wait Time for Pod Rebuilding After Recovery (s): After the fault is self-healed, the system will continue to detect the Pod's health status within this time to determine if the fault is recovered. The default value is 8 minutes. Ensure this duration is greater than the maximum Pod rebuilding time.

Experiment Execution

Step 1: Experiment Preparation

Run Pod instances on the super nodes, such as Serverless clusters and super nodes in a standard cluster.
Enter the Agent Management page to install fault injection agents in the cluster where the Pods are located.

Step 2: Create an Experiment

1. Log in to Tencent Cloud Smart Advisor > Chaotic Fault Generator, go to **Experiment Management** page, and click **Create a New Experiment.**
2. Click **Skip and create a blank experiment**, and fill in the experiment details.

Step 3: Add Experiment Instances and Actions

1. In the experiment object configuration section, select either **Standard Cluster Pod** or **Serverless Cluster Pod** for the object type container.
2. Add an instance and select the corresponding Pod name. Select the **Cluster ID** and **Namespace** for injection, which will automatically retrieve Pods in that namespace under the cluster.
3. Add experiment actions. Select the **Serverless Pod Virtual node shutdown** fault under **Pod Operation** category.
4. For detailed action parameters configuration, see **Fault Parameter Description**.

Step 4: Global Configuration

You can configure advancing methods, monitoring metrics, guardrails, and other policies during the global configuration phase.

Fault Result Observation

Click **Execute** and wait for the fault injection to complete. Observe the Pod status in the TKE Console > Cluster. Enter the cluster basic information page, and click **Pod**. When the Pod’s Ready status shows **ProbeTimeout**, it indicates that the underlying virtual node has been successfully shut down, causing the Agent to report a timeout (At this point, the traffic of the abnormal Pod has been removed).

If the **Auto-Rebuild Pod** is set to Yes in the fault parameters, the Pod rebuild will be automatically triggered after the fault. Pods will rebuild and self-heal within a certain time, and you can observe the corresponding Pod rebuild events.
If the **Auto-Rebuild Pod** is set to No in the fault parameters, the fault will persist until a manual execution of rollback is executed or until the fault duration ends, at which point the fault will automatically recover. The fault recovery process will also involve reallocating underlying virtual nodes and rebuilding Pods.
Additionally, by observing the Pod monitoring metrics, you can see the fault process. You can observe that the various metrics of the Pod show checkpoints during the fault and then recover after the fault is recovered (Note: Since Pods are re-scheduled, the Pod restart count will not increase).

FAQ

How to Simulate a Single Availability Zone Fault of a Super Node?

If you configured too many availability zones when purchasing the super nodes, you can first lockdown the super nodes in the availability zone to be faulted. Then, execute the virtual node shutdown fault on all Pods scheduled to that super node, simulating a scenario where the entire super node is unavailable due to a single availability zone fault, and the Pods are migrated to other availability zones.

Simulating Serverless Cluster Pod Network Faults

Last updated：2024-09-26 15:47:38

Background

The CFG offers simulations of five types of network faults for Serverless clusters: network disorder, network duplication, network package loss, network delay, and network DNS anomalies. In a Kubernetes cluster, the stability of network performance is crucial for service response speed and user experience, as Pod lifecycles are closely linked to requests. Simulating these network faults helps you better understand your service's performance under different network conditions, allowing you to optimize your application and enhance its fault tolerance.

Experiment Execution

Experiment Preparation

Purchase Serverless cluster container instances and deploy test services.
Enter Agent Management page and install the agents.

Experiment Steps

Step 1: Create an Experiment

1. Log in to Tencent Smart Advisor > Chaotic Fault Generator, enter the **Experiment Management** page, click **Create a New Experiment.**
2. Click **Skip and create a blank experiment**, and fill in the experiment details.

Step 2: Add Experiment Instances and Actions

Adding Node Type Fault for Regular Node in Standard Cluster

1. In the experiment object configuration step, select Container — Serverless Cluster Pod.

2. Select Add Instance, and select the required Cluster ID and Namespace, which will automatically fetch all Pods in that namespace.
3. Add experiment actions. Select network fault actions under Network Resources.
Network disorder: Simulates scenarios where network data packages arrive at the destination in an order different from the sending order.
Network duplication: Simulates scenarios where network data packages are repeatedly sent during transmission.
Network packet loss: Simulates scenarios where network data packages are lost during transmission.
Network latency: Simulates scenarios where network data packages are delayed in reaching the destination.
DNS tampering: Simulates scenarios with DNS parsing anomalies.
4. Configuring Fault Action Parameters
Network Card: The network interface name must exist in the selected instances; otherwise, fault injection may fail. Use a semicolon (;) to separate multiple network interfaces.
Whitelist Port: Ports that are not affected. Use a semicolon (;) to separate multiple ports.
Local Port: Packages sent from the local machine on specified ports will be affected. Use a semicolon (;) to separate multiple ports.
Remote Port: Packages sent to specified ports will be affected. Use a semicolon (;) to separate multiple ports.
Remote IP: Packages sent to specified IPs will be affected. Use a semicolon (;) to separate multiple IPs.
5. After action parameter configuration, click Next. Configure guardrail policy and monitoring metrics considering actual situations, and click Submit to complete experiment creation.

Step 3: Execute the Experiment

Click **Execute Experiment**. Since network actions are high-risk, authentication must be completed before executing the action.

High Cluster Pod Resource (CPU, Memory, and Disk) Utilization Rate

Last updated：2024-09-26 15:47:38

Background

In today's cloud computing and microservice architecture, Kubernetes has become the actual standard for container orchestration and management. It not only provides efficient container management capabilities but also supports advanced features like auto-scaling, service discovery, and load balancing, greatly simplifying the deployment and Ops of cloud-native applications. However, as the complexity of applications increases, ensuring high availability and stability of the system becomes crucial. Chaos Engineering, a practice method of testing system stability by actively introducing faults, is essential for enhancing the robustness of Kubernetes clusters.
This document introduces how to conduct fault simulation for CPU, memory, and disk resources in a Tencent Cloud Standard Cluster environment to help development and Ops teams better understand and prepare for potential system faults.

Experiment Execution

Step 1: Experiment Preparation

Prepare a TKE container instance and deploy a test Pod.
Enter the Agent Management page to install the agent for the TKE container. For specific installation steps, see Agent Management.
Note:
Since the fault injection process and fault behavior are consistent between Serverless clusters and standard clusters, you can refer to the standard cluster fault practices in this document.

Step 2: Experiment Orchestration

1. Log in to Tencent Smart Advisor > Chaotic Fault Generator, go to **Experiment Management** page, click **Create a New Experiment**, click **Skip and create a blank experiment.**
2. Fill in basic information of the experiment.
3. Select **Container** as the resource type, select **Standard Cluster Pod** as the resource object, and then **Add Instance.**
4. To add an experiment action, click **Add Now**, and configure fault action parameters.
Configuring High CPU utilization Action

Note:

CPU Utilization: Specify CPU load percentage, which is 0 to 100.

Duration: Duration of a fault action, upon lapse of which, the agent will automatically recover the fault.

Scheduling Priority: It affects process priority in CPU scheduling. A lower nice value makes it more likely that the process would have a CPU time slice so that its execution priority can be improved. It is effective only if utilization is 100%.

Container Selection Mode: Select the fault range and inject faults into some or all containers under the Pod.

Configuring High memory utilization Action

Note:

Memory Utilization: Specify a memory load percentage that is 0 to 100.

Duration: Duration of a fault action, upon lapse of which, the agent will automatically recover the fault.

Enable OOM Protection: If it is enabled, the possibility of fault process OOM-KILL will be reduced, and business processes will be killed first.

Memory Occupation Rate: Memory usage increase per second.

Configuring High disk usage Action

Note:

Disk Directory: A disk directory to be populated, i.e., a directory where files are written.

File Size: Size of a file populated.

Disk utilization: Learn disk usage through staf commands, and calculate the file size required for specified utilization.

Reserved Space: Size of remaining space.

Duration: Duration of a fault action, upon lapse of which, the agent will automatically recover the fault.

If there are file size, disk utilization, and reserved space parameters, the priority calculation logic is disk utilization > reserved space > file size.

Configuring High disk IO load Action

Note:

Disk Directory: Specify a directory to enhance disk IO, which will apply to the disk it resides on.

Mode: Provide both read and write modes to execute high loads.

Block Size: Specify block size for every read or write.

Number of Blocks: Specify number of blocks to be copied.

Duration: Duration of a fault action, upon lapse of which, the agent will automatically recover the fault.

5. After action parameter configuration, click **Next**. Configure **Guardrail Policy** and **Monitoring Metrics** considering actual situations. After all configurations are completed, click **submit** to complete experiment creation.

Step 3: Execute the Experiment

1. Click **Execute** to start the experiment with the high CPU utilization action, and observe the **monitoring metrics.**

2. Execute the high memory utilization action and observe the monitoring metrics.

3. Execute the high disk utilization action, log into the container, and use the **df** command to observe the gradual increase in disk load until it reaches the specified utilization. After the rollback action is executed, it will recover to

normal.

```
bash-4.2# df
Filesystem     1K-blocks     Used Available Use% Mounted on
overlay         51486416 13301168  35966988  27% /
tmpfs              65536        0     65536   0% /dev
tmpfs            1874884        0   1874884   0% /sys/fs/cgroup
/dev/vda1       51486416 13301168  35966988  27% /etc/hosts
shm                65536        0     65536   0% /dev/shm
tmpfs            1048576       12   1048564   1% /run/secrets/kubernetes.io/servic
tmpfs            1874884        0   1874884   0% /proc/acpi
tmpfs            1874884        0   1874884   0% /proc/scsi
tmpfs            1874884        0   1874884   0% /sys/firmware
bash-4.2# df
Filesystem     1K-blocks     Used Available Use% Mounted on
overlay         51486416 29465124  19803032  60% /
tmpfs              65536        0     65536   0% /dev
tmpfs            1874884        0   1874884   0% /sys/fs/cgroup
/dev/vda1       51486416 29588004  19680152  61% /etc/hosts
shm                65536        0     65536   0% /dev/shm
tmpfs            1048576       12   1048564   1% /run/secrets/kubernetes.io/servic
tmpfs            1874884        0   1874884   0% /proc/acpi
tmpfs            1874884        0   1874884   0% /proc/scsi
tmpfs            1874884        0   1874884   0% /sys/firmware
bash-4.2# df
Filesystem     1K-blocks     Used Available Use% Mounted on
overlay         51486416 38972460  10295696  80% /
tmpfs              65536        0     65536   0% /dev
tmpfs            1874884        0   1874884   0% /sys/fs/cgroup
/dev/vda1       51486416 38972460  10295696  80% /etc/hosts
shm                65536        0     65536   0% /dev/shm
tmpfs            1048576       12   1048564   1% /run/secrets/kubernetes.io/servic
tmpfs            1874884        0   1874884   0% /proc/acpi
tmpfs            1874884        0   1874884   0% /proc/scsi
tmpfs            1874884        0   1874884   0% /sys/firmware
bash-4.2# df
Filesystem     1K-blocks     Used Available Use% Mounted on
overlay         51486416 13303184  35964972  28% /
tmpfs              65536        0     65536   0% /dev
tmpfs            1874884        0   1874884   0% /sys/fs/cgroup
/dev/vda1       51486416 13303184  35964972  28% /etc/hosts
shm                65536        0     65536   0% /dev/shm
tmpfs            1048576       12   1048564   1% /run/secrets/kubernetes.io/servic
tmpfs            1874884        0   1874884   0% /proc/acpi
tmpfs            1874884        0   1874884   0% /proc/scsi
```

```
tmpfs            1874884        0   1874884   0% /sys/firmware
```

4. Execute the high disk I/O load action, log into the container, and observe using the iostat command.

Fault in progress

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           6.19    0.00    9.28   23.71    0.00   60.82

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %uti
vda               0.00    22.00 3689.00    4.00 36660.00   104.00    19.91     0.01    0.25    0.25    0.50   0.27  99.3
```

After rollback

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           4.55    0.00    4.04    0.00    0.00   91.41

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %ut
vda               0.00     2.00    0.00    3.00     0.00    20.00    13.33     0.00    0.67    0.00    0.67   0.33   0.
```

.

Cluster Pod TencentCloud API Ban

Last updated：2024-09-26 15:47:38

Background

TencentCloud API 3.0 is an API gateway of Tencent Cloud and a channel for communication between clients and business services. At present, 400+ products and 20,000+ interfaces have been accessed, carrying tens of billions of calls in a day. Owning abundant tool ecosystems, a set of standard schemes for external APIs has been developed. As an important linkage for inter-service calls, it is featured with authentication, frequency limiting, audit, and security, and its stability has a direct effect on the reliability of cloud products and user experience.

Experiment Execution

Step 1: Experiment Preparation

Prepare a TKE container instance and deploy a test Pod.
Prepare a test script file, which can see the Cloud Virtual Machine - DescribeInstances API.
Enter the Agent Management page to install the agent for the TKE container. For specific installation steps, see Agent Management.
Note:
Since the fault injection process and fault behavior are consistent between Serverless clusters and standard clusters, you can refer to the standard cluster fault practices in this document.

Step 2: Experiment Orchestration

1. Log in to Tencent Smart Advisor > Chaotic Fault Generator, go to **Experiment Management** page, click **Create a New Experiment**, click **Skip and create a blank experiment.**
2. Fill in basic information of the experiment.
3. Select **Container** as the resource type, select **Standard Cluster Pod** as the resource object, and then **Add Instance.**
4. Click **Add Now** to add the experiment action and configure the fault parameters for the Cloud Product **Cloud API ban**.
5. Confirm experiment orchestration, and click **Submit** to complete experiment creation.

Step 3: Execute the Experiment

1. Click **Execute** to start the Cloud API ban action for the experiment.
2. Check the fault result, and you will find a TencentCloud API request failed.

```
[root@api-ban-6798884b49-wjcmj ~]# python3 api_ban.py
[TencentCloudSDKException] code:ClientNetworkError message:HTTPSConnectionPool(host='cvm.internal.tencentcloudapi.com', port=443): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPSC
 object at 0x7fe3de50e610>: Failed to establish a new connection: [Errno 111] Connection refused')) requestId:None
```

3. After the duration expires or a recovery action has been executed, normal access can be recovered.

```
[root@api-ban-6798884b49-wjcmj ~]# python3 api_ban.py
[TencentCloudSDKException] code:ClientNetworkError message:HTTPSConnectionPool(host='cvm.internal.tencentcloudapi.com', port=443): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.co
ection refused')) requestId:None
[root@api-ban-6798884b49-wjcmj ~]# python3 api_ban.py
{"TotalCount": 11, "
R", "SystemDisk": {"
Id": null}, "Virtual
```

Big Data
Elasticsearch Service Node Down

Last updated：2024-09-26 15:49:18

Background

An Elasticsearch cluster comprises multiple nodes that work together to process client requests. In production environments, nodes may encounter abnormal issues due to hardware faults, network problems, or software defects. If a node encounters a fault, it can lead to a decrease in the overall cluster performance and even disrupt normal business operations. Therefore, the CFG provides node fault simulation.
Node fault simulation can help us understand how the Elasticsearch cluster performs under various fault scenarios. For example, by simulating node down, network partitions, disk damage, and other faults, you can observe the cluster's recovery process and assess risks such as data loss and inquiry delay. Continuous fault simulation helps identify and fix potential issues, optimize cluster configuration, and enhance cluster robustness. Additionally, node fault simulation can be used for training and experiments. By simulating real-world fault scenarios, team members can become familiar with fault troubleshooting processes and improve their ability to respond to faults. Meanwhile, fault simulation can also serve as a stress testing tool to verify the cluster's stability under high-load conditions.
Conducting node fault simulations for Elasticsearch is a crucial method for ensuring cluster stability and reliability. By simulating various fault scenarios, you can proactively discover and resolve issues, improve the cluster's fault tolerance and availability, and ensure the smooth operation of the business.

Experiment Preparation

Prepare an ES cluster instance for experiments.

Step 1: Create an experiment

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator Console.
2. In the left sidebar, select **Experiment Management** page, and click **Create a New Experiment**.
3. Click **Skip and create a blank experiment**.
4. After filling in the basic information, you can enter the experiment object configuration. Select **Big Data** as the resource type, and **Elasticsearch Cluster** as the resource object, then click **Add Instance**. After you click Add Instance, a list of all Elasticsearch cluster instances in the current region will appear. You can filter instances based on cluster name, cluster ID, or private IP address.

5. After selecting the target instance, click **Add Now** to add the **ES Node down** experiment action, then click **Next**.
6. Set action parameters. In this document, the **Random Node Downtime** is selected. Click **Confirm**.(Specific fault parameters can be selected based on the experiment's objectives.)
7. Click **Next** to go to Global Configuration. See Quick Start for Global Configuration.
8. After confirmation, click **Submit**.
9. After creating the experiment, click **Experiment Details** in the pop-up dialog box to enter the Experiment Details page.

Step 2: Execute the experiment

1. Observe the instance monitoring data before the experiment, focusing on the advanced monitoring metrics. You can go to ES console and click **Elasticsearch Cluster** > **Cluster ID/Name** > **Node Monitoring** to view.
2. On the Experiment Details page, click **Execute** to initiate the fault actions.
3. After the fault injection is successful, click the Fault Action panel to view the results and the executed nodes.

Cloud Load Balancer
CLB Stop Fault

Last updated：2024-09-26 15:49:18

Background

As a proxy, Cloud Load Balancer (CLB) generally serves various business services by providing a cost-effective and transparent method to expand network device and server bandwidth, increase throughput, enhance network data processing capabilities, and improve network flexibility and availability.
As a crucial component of network transmission, CFG provides you the ability to simulate CLB stop faults, assisting you in achieving instance or listener unavailability.

Fault Principle

Injecting a stop fault into CLB causes instances or listeners to stop, and leads to client access failures.

Experiment Execution

Experiment Preparation

Create a virtual private cloud and deploy a CVM and CLB instance within this network. Deploy test services on the CVM instance and create listeners on the CLB instance to monitor the CVM service ports.

Experiment Steps

Step 1: Create an experiment

1. Log in to Tencent Smart Advisor ＞ Chaotic Fault Generator, and enter the **Experiment Management** page, click **Create a New Experiment**.
2. Click **Skip and create a blank experiment**, and fill in the experiment details**.**

3. Add actions. The platform provides **CLB Stop Fault Action.**
Three stop objects are provided: **CLB instance**, **Layer 4 path listener**, and **Layer 7 path listener**.

Step 2: Execute experiment actions

Click **Execute** to distribute the fault actions. Log in to the same VPC instance and analyze the business service responses.

Observe Results

Stop Object: CLB Instance

1. In a steady-state metric performance, the corresponding instance status in the CLB console will show as **Normal**.
2. In a steady-state metric performance, the corresponding service responds normally when accessed via curl.

```
[root@VM-1-11-tencentos ~]# curl http://[redacted]:8080
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
    <head>
        <title>Test Page for the Nginx HTTP Server on  TencentOS Linux</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <style type="text/css">
            /*<![CDATA[*/
```

3. After fault injection, the corresponding instance status in the CLB console will show as stopped.
4. After fault injection, the service does not respond via curl.

```
[root@VM-1-11-tencentos ~]# curl http://[redacted]:8080
```

Stop Object: Layer 4 Path Listener

In a steady-state metric performance, the listener service matched by the rule responds normally via curl.

```
[root@VM-1-11-tencentos ~]# curl http://[redacted]:8080
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
    <head>
        <title>Test Page for the Nginx HTTP Server on  TencentOS Linux</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <style type="text/css">
            /*<![CDATA[*/
```

After fault injection, the listener service matched by the rule does not respond via curl.

```
[root@VM-1-11-tencentos ~]# curl http://[redacted]:8080
```

Stop Object: Layer 7 Path Listener

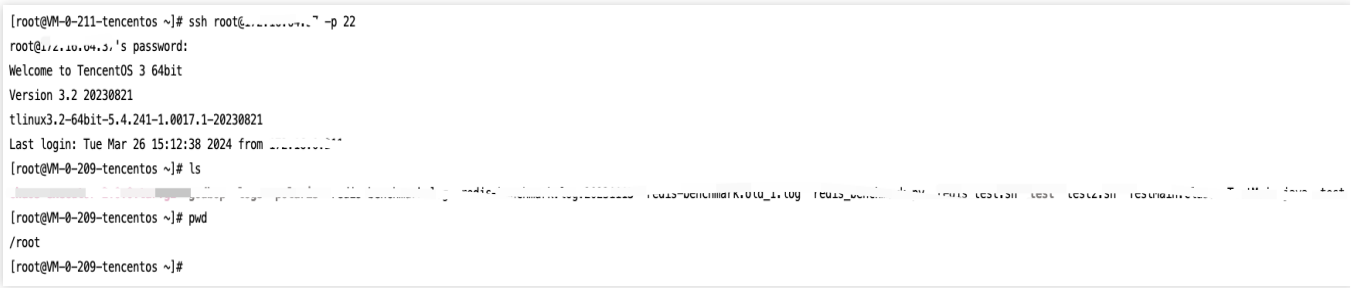
In a steady-state metric performance, the listener service matched by the rule responds normally via curl.

```
[root@VM-1-11-tencentos ~]# curl http://[redacted]:8785/image/
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
    <head>
        <title>Test Page for the Nginx HTTP Server on  TencentOS Linux</title>
```

After fault injection, the listener service matched by the rule does not respond via curl.

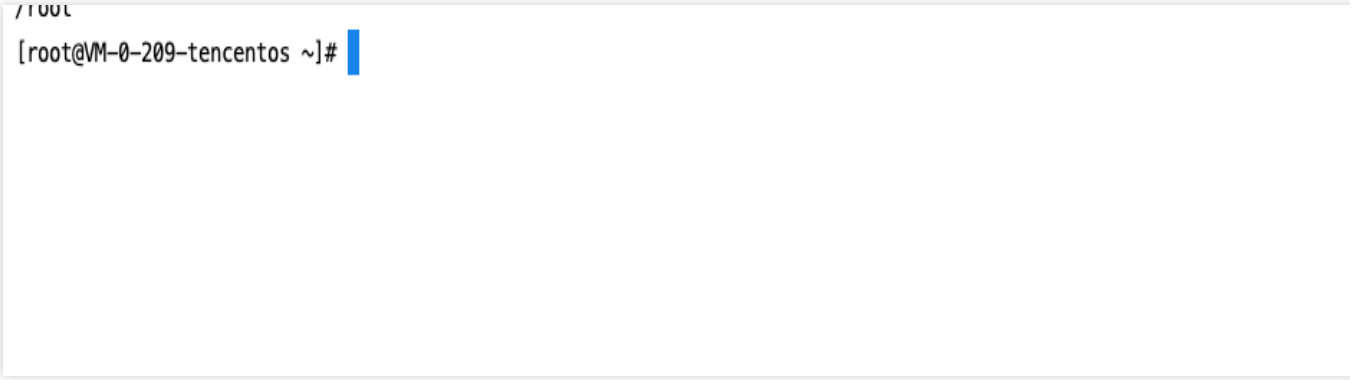
Long Link Results Observation (SSH Simulated Long Link)

In a steady-state metric performance, the client can access normal and input command operations.

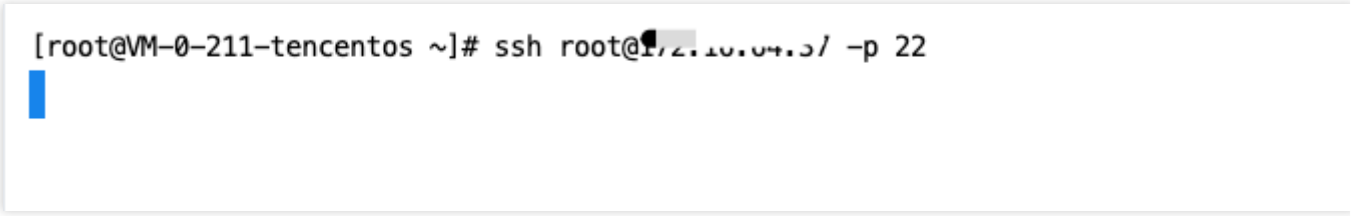


In a steady-state metric performance, long links with the client can be observed via `netstat -tu` on the server side.

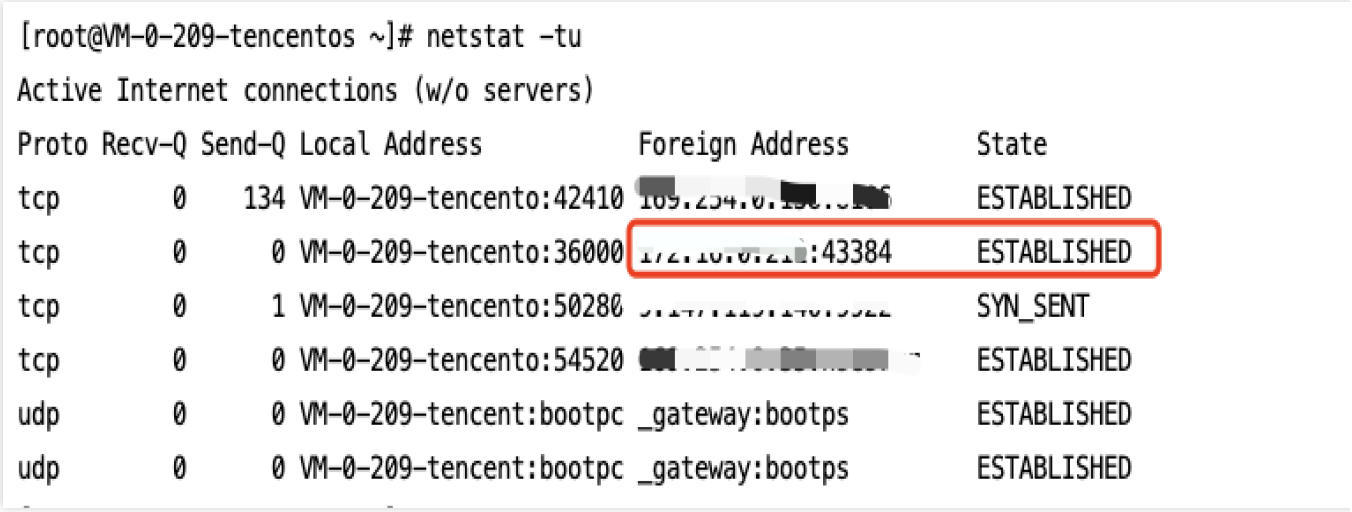
After the fault injection, the original client can no longer be accessed, and command input is not possible.



After the fault injection, establishing a new long link does not respond.



After the fault injection, **the server-side long link still exists**.



Message Queue
CKafka Broker High Disk IO Load

Last updated：2024-09-26 15:49:18

Background

The message middleware plays a crucial role in distributed systems. However, in actual production environments, various factors can lead to high disk I/O load on broker nodes. Here are some common real-world scenarios:
High concurrent write: A large number of messages are written to the broker at the same time. This can occur in high-traffic message production environments, such as large-scale real-time data transmissions or high-frequency log records.
Message storage pressure: If the volume of messages stored on the broker is very large, or if the messages are of considerable size, this may occur in long-running systems or applications that need to retain a large number of historical messages.
High volume of message copy and synchronization: When the broker is part of a cluster, the process of copying and synchronizing a large number of messages increases the disk I/O load. This includes copying messages from one broker to others to ensure high availability and data redundancy.
Index and retrieval operations: If the message storage on the broker uses index structures, maintaining and retrieving indexes when the data volume is large can also increase the disk I/O load. This may occur in applications that require quick message retrieval, such as inquiring messages based on specific conditions.
Disk fault and recovery: When a disk fault occurs or data recovery is needed, the broker executes disk I/O-intensive operations, such as recovering data from backups or fixing disks.
To address these issues, the CFG provides CKafka Broker high disk I/O load experiment actions. These experiment actions test the business system's ability to process unexpected delays and recovery when faced with high disk I/O loads on CKafka Brokers, thereby improving business security and stability.

Must-Knows

Instance type: This action is only open for fault injection on instances of the **CKafka Pro Edition** type. **CKafka Standard Edition** instances do not support experiments yet.
Instance status: It is recommended that the instance used for the experiment has real message production and consumption traffic, with the number of topic partitions greater than 3, to better observe the impact of the fault on the business.(non-mandatory item)

Experiment Preparation

Prepare a CKafka Pro Edition instance for the experiment.

Step 1: Create an experiment

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator Console.
2. In the left sidebar, select **Experiment Management** page, and click **Create a New Experiment**.
3. Select **Skip and create a blank experiment.**
4. After filling in the basic information, enter the Experiment Object Configuration. Select the **Middleware** > **CKafka** for the object type, and click Add Instance. After clicking **Add Instance**, all CKafka instances in the target region will be listed. You can filter instances based on **Instance ID, VPC ID, Subnet ID**, and **Tags**.
5. After you select the target instance, click **Add Now** to add the experiment action.
6. Select the experiment action **Broker disk IO high load**, and then click **Next**.
7. After setting action parameters, click **Confirm**.
8. Click Next to enter the Global Configuration. For global configuration, see Quick Start.
9. After confirmation, click **Submit.**
10. Click **Experiment Details** to start the experiment.

Step 2: Execute the experiment

Note:

Currently, the CKafka console does not have disk I/O load monitoring metrics. When CKafka traffic is high, you can observe production and consumption duration monitoring metrics. For disk I/O load monitoring data, you can submit a ticket to contact CKafka Ops.

1. Observe the instance monitoring data before the experiment, and you can focus on the monitoring metrics in Advanced Monitoring. You can view this on the CKafka Console.
2. As the experiment is manually executed, fault actions must be executed manually. Click **Execute** in Action Card to start fault injection.
3. While fault injection is in progress, you can click the links in the logs to view monitoring metrics such as production and consumption duration in **Advanced Monitoring**.

CKafka Broker High CPU Load

Last updated：2024-09-26 15:49:18

Background

The message middleware plays a crucial role in distributed systems. However, in actual production environments, various factors can lead to a high CPU load on broker nodes. Here are some common scenarios:
High message throughput: If a specific topic or partition within the CKafka cluster receives a very high message throughput, the broker node needs to process a large number of read and write operations.
Large number of consumer groups: If a large number of consumer groups subscribe to the same topic or partition, the broker should distribute and manage messages for each consumer group.
Copy and synchronize: If the CKafka cluster has data copy and synchronization features enabled, the broker needs to process read and write operations for copy tasks and synchronization communication with other brokers.
Compression and decompression: If messages are stored compressed, the broker needs to perform compression and decompression operations, which may consume significant CPU resources.
Index and log compression: CKafka uses indexes to speed up message searches. If the index is too large or needs compression, the broker must perform index maintenance and compression operations.
High concurrent connections: If there are a large number of producers and consumers connected to the broker, the broker needs to process the establishment and maintenance of these connections, increasing CPU load.
When a broker node experiences high CPU load, several issues may arise:
Increased delay: High CPU load may slow down message processing, thereby increasing message transmission and processing delays. This can affect the speed at which consumers read messages from CKafka, potentially preventing them from obtaining the latest messages in time.
Decreased throughput: Due to the CPU resources being occupied by high-load tasks, the CKafka Broker may be unable to process more messages, leading to an overall decrease in throughput. This will affect the speed at which producers send messages and consumers consume messages.
Network congestion: High CPU load may cause the CKafka Broker to be unable to process network requests in time, leading to network congestion. This will affect data copy and synchronization with other brokers, potentially causing increased data copy delays or untimely synchronization.
Increased response time: Due to the high CPU load, the CKafka Broker may be unable to respond to client requests in time, leading to increased client wait time. This will affect the performance and response time of applications interacting with the CKafka cluster.
To address these issues, the CFG provides CKafka Broker high CPU load experiment actions. These experiment actions test the business system's ability to process unexpected delays and recovery when faced with high CPU loads on CKafka Brokers, thereby improving business security and stability.

Must-Knows

Instance type: This action is only open for fault injection on instances of the **CKafka Pro Edition** type. **CKafka Standard Edition** instances **do not support** experiments yet.
Instance status: It is recommended that the instance used for the experiment has real message production and consumption traffic, with the number of topic partitions greater than 3, to better observe the impact of the fault on the business.(non-mandatory item)

Experiment Preparation

Prepare a CKafka Pro Edition instance for the experiment.

Step 1: Create an experiment

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator Console.
2. In the left sidebar, select **Experiment Management** page, and click **Create a New Experiment**.
3. Click **Skip and create a blank experiment**.
4. After filling in the basic information, enter the Experiment Object Configuration. Select the **Middleware** > **CKafka** object type, and **click Add Instance**. After clicking **Add Instance**, all CKafka instances in the target region will be listed. You can filter instances based on **Instance ID, VPC ID, Subnet ID**, and **Tags**.
5. After you select the target instance, click **Add Now** to add the experiment action.
6. Select the experiment action **Broker-CPU High Load**, and then click **Next**.
7. Set action parameters. In this document, the CPU load rate **80%** is selected and the duration is set to **200 s**, then click **Confirm**.
8. Click Next to enter the Global Configuration. For global configuration, see Quick Start.
9. After confirmation, click **Submit.**
10. Click **Experiment Details** to start the experiment.

Step 2: Execute the experiment

1. Observe the instance monitoring data before the experiment, and you can focus on the monitoring metrics in Advanced Monitoring. You can view this on the CKafka Console.
2. As the experiment is manually executed, fault actions must be executed manually. Click **Execute** in Action Card to start fault injection.
3. While fault injection is in progress, you can click the links in the logs to observe in **Advanced Monitoring**.
4. Observe that the CPU utilization reaches the set value.

CKafka Broker Down

Last updated：2024-09-26 15:49:18

Background

Nowadays, Kafka, a highly performed, highly reliable, and distributed message queue system, is widely applied in large-scale internet services by renowned companies like Tencent, Facebook, LinkedIn, Netflix, and Airbnb. However, in large-scale distributed systems, the unpredictability, complexity, and coupling of services often lead to unforeseen fault events. When a Kafka Broker node is down, the following faults may appear:
Data loss: If messages are being written to a broker that is down, data loss may occur. During this period, the producer may not be able to write messages to the partitions and copy them to other replicas, resulting in message loss.
Reduced availability: A down broker no longer processes requests, which may cause producer and consumer requests to time out. If multiple broker nodes go down, the availability of the cluster will be further reduced.
Increased delay: A down broker no longer processes requests, which may cause increased delay for producer and consumer requests. When requests time out and are resent, they may have to wait for responses from other nodes, resulting in longer delays.
Unbalanced leader election : If a down broker is the leader of the partition, leader election is required. If the down broker is restarted and its backups are not deleted before exit, it may result in an unbalanced leader election.
Replica synchronization delay: If a down broker is the replicator of one or more replicas, it may cause a delay in replica synchronization. If this delay is high, it will cause producers and consumers to read or write outdated data.

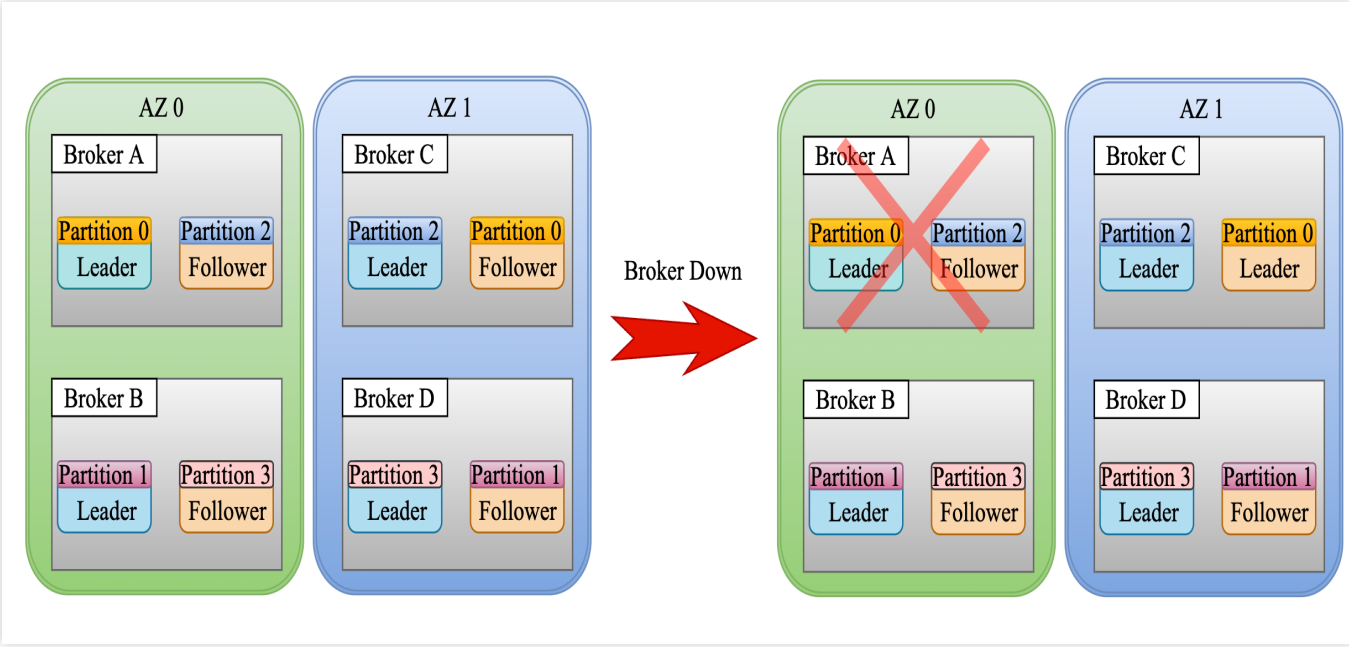
Tencent Cloud Message Queue CKafka (Cloud Kafka)

Based on the open-source Apache Kafka TDMQ engine, Tencent Cloud Message Queue CKafka (Cloud Kafka) provides a TDMQ service with high throughput performance and high scalability. It is perfectly compatible with the APIs of Apache Kafka v0.9, v0.10, v1.1, v2.4 v2.8, and v3.2. It has significant strengths in performance, scalability, business security assurance, and Ops, allowing you to enjoy powerful features at a low cost while eliminating tedious Ops work.

Fault Principle

Fault effect: For the CKafka instance, a broker or a broker in a specific availability zone is down and goes offline. This will result in unsynchronized replicas in the instance.

CKafka Broker down logic: When a broker is down (a new broker node will not automatically start), a new topic-partition leader replica is elected from the remaining brokers in the cluster. If one broker fails, the replicas on that machine become unavailable. It will result in one fewer replica for the topic and will not recover to the expected number of replicas for the user. For CKafka, as long as there is one available replica, production and consumption can proceed normally. After the fault is recovered, the replicas on the original broker will be copied from other broker nodes to recover.
User-side risks of CKafka Broker node down: In the case of a fault, the leader replica switch usually occurs within seconds. During this period, users may receive retry warnings. Due to the very fast switching, message loss usually does not occur (Extremely low probability: Messages are written to the page cache and flushed to disk by an asynchronous thread. If the data in the page cache has not been flushed to disk, and the follower has not yet synchronized the messages from the leader replica, there is a possibility of losing some messages).
Fault example diagram: Consider a CKafka instance (4 broker nodes deployed across two availability zones). After Broker A is down and goes offline, the CKafka instance will select the replica of Partition 0 located on Broker C as the new leader replica to provide messaging services.



Must-Knows

Instance type: This action is only open for fault injection on instances of the **CKafka Pro Edition** type.
Instance status: The instance used for experiments must contain valid data (i.e., existing topics).
Avoid frequent experiments in a short time: Frequent experiments can result in insufficient instance health status data, making it difficult to determine whether the issues were caused by the experiment or other factors. This can lead

to experiment request rejections and experiment failures. If an experiment fails, please retry after the interval of **15 minutes**.

Prevent overloading instance production and consumption traffic during the experiment: During the broker down fault experiment, the **bandwidth utilization** of other brokers, excluding the one that is down, must not exceed **80%**.

Experiment Preparation

Prepare a CKafka Pro Edition instance for the experiment, with a certain amount of valid data. This document uses a cross-AZ disaster recovery instance (with 4 brokers) as an example.

Step 1: Create an experiment

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator Console.
2. In the left sidebar, select **Experiment Management** page, and click **Create a New Experiment.**
3. Click **Skip and create a blank experiment**.
4. After filling in the basic information, enter the Experiment Object Configuration. Select **Middleware** > **CKafka** for the object type, and click Add Instance. After clicking **Add Instance**, all CKafka instances in the target region will be listed. You can filter instances based on **Instance ID, VPC ID, Subnet ID**, and **Tags**.
5. After you select the target instance, click **Add Now** to add the experiment action.
6. Select the experiment action **Broker down (Professional Edition)**, and then click **Next**.
7. Set action parameters. In this document, the **Specified availability zone downtime** mode is selected. Select the target availability zone for injection, and click **Confirm**.

Note:

This fault action supports the following 2 injection methods:

1. **Random single Broker downtime**: This will randomly select one node from all the broker nodes in the CKafka instance to inject the fault.
2. **Random availability zone downtime (Disaster recovery instance)**: This injection method is only supported for disaster recovery instances. It will randomly select one availability zone from the CKafka instance's availability zones to inject the fault. After the fault is injected, all brokers in the selected availability zone will be down and go offline. If a non-disaster recovery instance is used, no fault will be injected, and the injection will fail.
3. **Specified availability zone downtime (Disaster recovery instance)**: When you inject a fault into a specified availability zone, the fault will only be injected successfully if there is an instance in the specified availability zone. If no instance exists in the specified availability zone, the injection will fail. If you need to specify multiple availability zones for fault injection, it is recommended to split them into separate action groups.

8. Click Next to enter the Global Configuration. For global configuration, see Quick Start.
9. After confirmation, click **Submit.**
10. Click **Experiment Details** to start the experiment.

Step 2: Execute the experiment

1. Observe the monitoring data of the instance before the experiments, focusing on two key monitoring metrics: the broker survival rate and the number of unsynchronized replicas. You can view this on the CKafka Console.

Note:

There will be a certain delay in CKafka Console monitoring. Data changes can be observed after the fault injection is successfully carried out. **The Broker survival rate is collected every 5 minutes.**

2. As the experiment is manually executed, fault actions must be executed manually. Click **Execute** in Action Card to start fault injection. Start fault injection and wait for the fault injection to succeed.

3. Once the fault injection is successful, you can click the action card to view the corresponding execution details. It can be found that the fault has been injected successfully. At this point, you can go to the CKafka Console to observe data changes.

Monitor the broker survival rate metric. Notice if it decreases, it indicates that one broker has been down.

At the same time, observe the number of unsynchronized replicas metric, and there are unsynchronized replicas.

4. To execute recovery, click **Execute** recovery action.

5. After the recovery is successful, continue to observe the metrics.

The broker survival rate metric should recover to 100%.

The number of unsynchronized replicas should recover to 0.

TDMQ for RabbitMQ Broker Down

Last updated：2024-09-26 15:49:18

Background

TDMQ for RabbitMQ (TDMQ for RabbitMQ, referred to as TDMQ RabbitMQ version) is a message queue service independently developed by Tencent. It supports the AMQP 0-9-1 protocol and is fully compatible with all components and concepts of the open-source RabbitMQ. Additionally, it provides strengths like compute-storage separation and flexible scaling-in/out. TDMQ for RabbitMQ is equipped with highly flexible routing to accommodate various business message shipping rules. It can buffer upstream traffic pressure and ensure the stable running of the message system. The persistence mechanism ensures the high reliability of TDMQ for RabbitMQ. Set the persistence of exchanges, queues, and messages to prevent metadata and message content loss after service restarts. The messages are stored using a three-replica policy, allowing for quick data migration in case of a physical machine fault, ensuring 3 backups of user data are available. The service availability reaches 99.95%.
To help users verify TDMQ for RabbitMQ's disaster recovery capabilities in the face of an availability zone-level fault, the CFG provides an availability zone-level broker down fault experiment action for TDMQ for RabbitMQ (cross-AZ instances) to simulate a real availability zone disaster. By conducting a disaster recovery experiment on the Broker, users can verify the disaster recovery capabilities of their business, test the durability mechanism, and evaluate the scope and duration of the fault's impact.

Must-Knows

The broker down fault experiment action only supports **TDMQ for RabbitMQ instances deployed across availability zones**. Otherwise, fault injection cannot be performed. For instances that are not deployed across availability zones, please upgrade to a cross-availability zone instance before conducting the experiment.
If you need to inject faults into multiple instances, it is recommended to split them into multiple action groups, with each action group processing one instance.
After fault injection, **there may be momentary disconnections on the producer or consumer side**. Please proceed with caution.

Experiment Preparation

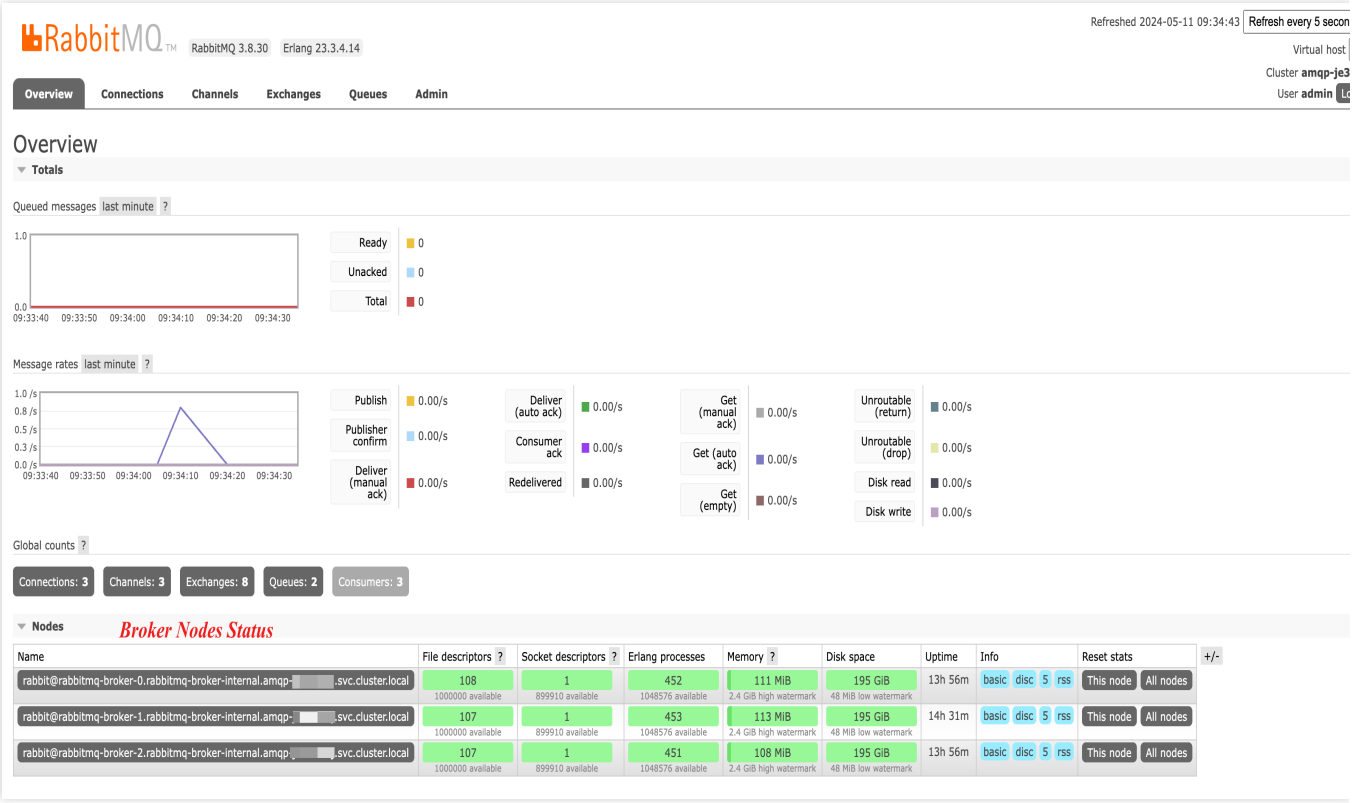
Prepare a TDMQ for RabbitMQ instance deployed across availability zones that is ready for the experiment.

Step 1: Create an experiment

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator Console.
2. In the left sidebar, select **Experiment Management** page, and click **Create a New Experiment**.
3. Click **Skip and create a blank experiment**.
4. After filling in the basic information, enter the Experiment Object Configuration. Select the **Middleware** > **RabbitMQ** object type, and click </5>Add Instance</5>. After you click Add Instance, all RabbitMQ instance information in the target region will be listed. You can filter instances based on **Instance ID** and **Instance Name**.
5. After you select the target instance, click **Add Now** to add the experiment action.
6. Select the experiment action **Broker Down**, and then click **Next**.
7. Select the corresponding availability zone for injection, then click **Confirm**.
8. Click **Next** to go to Global Configuration. See Quick Start for Global Configuration.
9. After confirmation, click **Submit.**
10. Click **Experiment Details** to start the experiment.

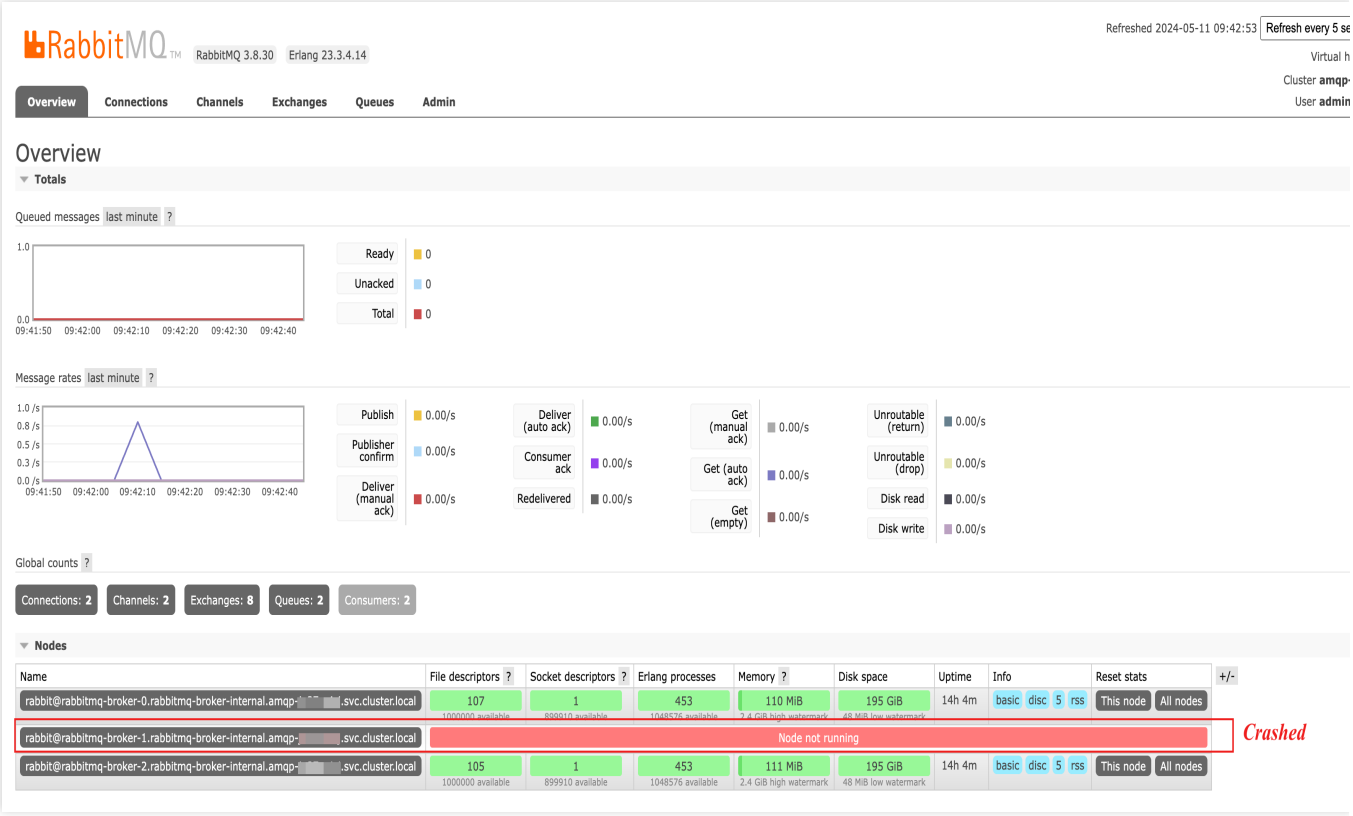
Step 2: Execute the experiment

1. Before executing the experiment, you can go to the TDMQ > RabbitMQ > Cluster Management section and use the **Web Console Access Address** of the corresponding instance to view the RabbitMQ Console for observation.
2. After entering the RabbitMQ Console, you can see the survival status of the corresponding Broker.

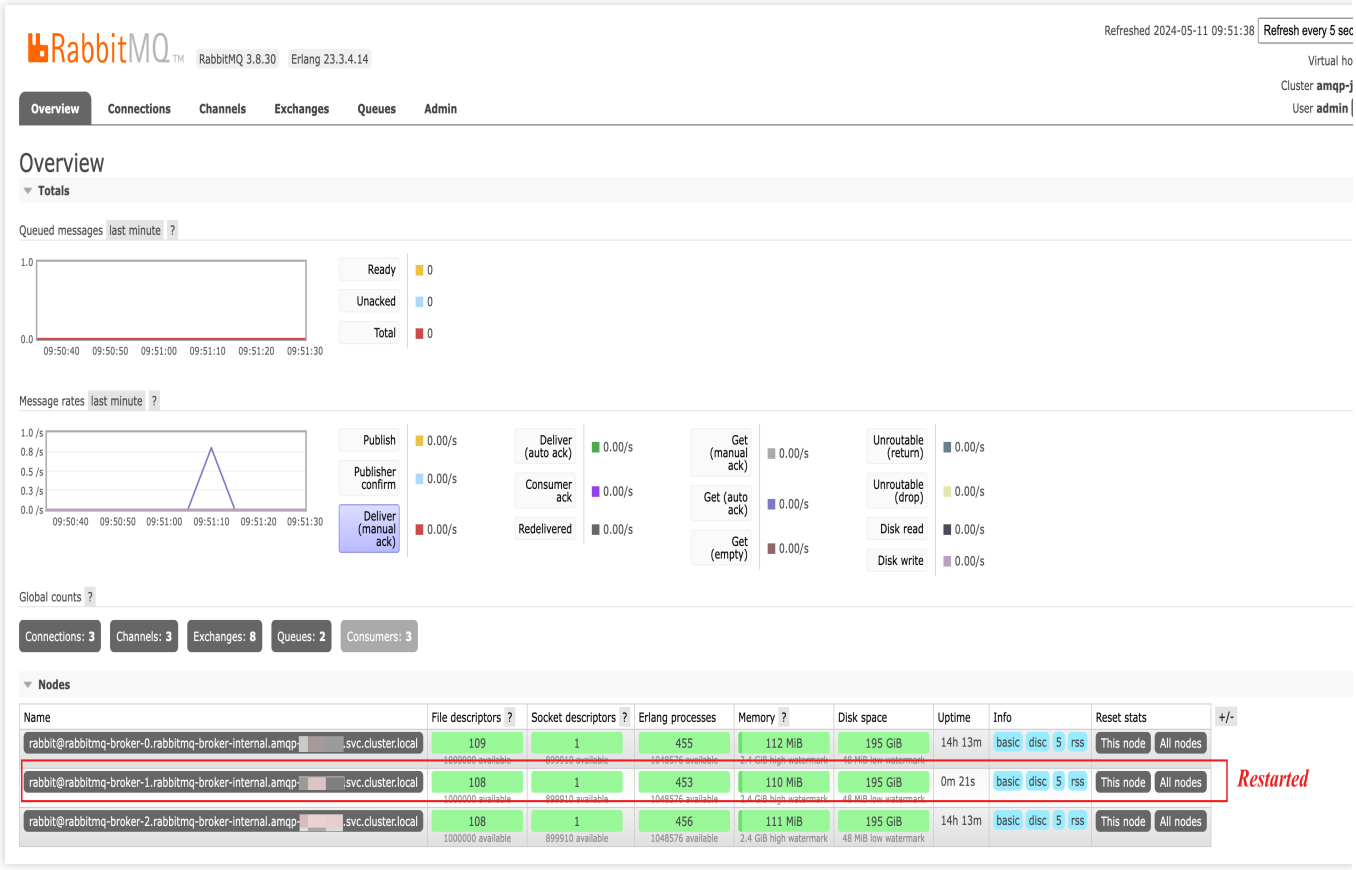


3. Since the experiment is manual, manually execution of the fault action is needed. As the experiment is manually executed, fault actions must be executed manually. Click **Execute** in Action Card to start fault injection.

4. Once the fault injection is successful, you can click the action card to view the corresponding execution details. It can be observed that the fault is successfully initiated and that the broker in the specified availability zone of the instance has gone offline.
5. Go to the RabbitMQ Web Console, and you can see that one broker node is down and not running.



You can also see on the Tencent Cloud TDMQ > RabbitMQ > Cluster Management page that the current instance is in abnormal status, indicating that a broker has not started up.
6. Click the **Execute** button of the recovery action to execute the recovery action.
7. It takes some time to resume execution. Once the recovery is complete, you can observe through the RabbitMQ Web Console that the down broker node has restarted.



You will also see on the Tencent Cloud TDMQ > RabbitMQ > Cluster Management page that the current instance has recovered to normal status.

Direct Connect
Simulating DC Tunnel Disconnection Faults

Last updated：2024-09-26 15:49:18

Background

Direct connect (DC) provides a fast and secure method to connect the cloud services and the local data center. Users can use a connection to connect Tencent Cloud computing resources across multiple regions, enabling flexible and reliable hybrid cloud deployment. DC tunnel is a network linkage segmentation of the connection, allowing creation of different DC tunnels associated with various DC gateways. In production environments, improper alarm configurations or unreasonable disaster recovery planning can lead to failures in receiving alarms or triggering disaster recovery plans during real faults, resulting in business losses. To proactively identify architectural risks, you can use chaos engineering to verify the disaster recovery of your direct connect deployment architecture in advance.
By simulating the DC tunnel disconnection fault, you can:
Verifying Alarm Reachability After DC Tunnel Disconnection
You can set alarm rules for connections, DC tunnels, and DC gateways in the Tencent Cloud observability platform. In the case of the DC tunnel fault, the corresponding alarm policy can be triggered. To verify the effectiveness of alarm rule configuration, you can simulate a DC tunnnel fault via the **Channel Disconnection Fault Simulation** action to observe whether the alarm rules are triggered.
Note:
For direct connect alarm configuration, see DC - Configure Alarms.
Verifying Disaster Recovery Capability of High Availability DC Deployment Architecture
Tencent Cloud DC can ensure the high availability of services in various fault scenarios (e.g. port abnormalities/optical module faults, network device failures, and access point IDC faults). To enhance the disaster recovery capability, DC typically adopt high-availability deployment. To verify the effectiveness of high availability deployment architecture and the business disaster recovery performance for actual faults, you can use the **Tunnel Disconnection Fault Simulation** action for fault verification.
Note:
For direct connect disaster recovery architecture, see DC - Network Planning.

Practical Examples

Verifying Alarm Reachability After DC Tunnel Disconnection

Experiment Preparation

An exclusive or shared **Dedicated Private Channel**. Note that the DC tunnel must be in a connected status and the version must be 2.0 (Currently, fault simulation supports only Direct Connect version 2.0).
In the Tencent Cloud observability platform, the corresponding alarm policy is configured for the DC tunnel disconnection fault.

Experiment Steps

Step 1: Create an experiment

Create an experiment. Tencent Smart Advisor-Chaotic Fault Generator (CFG) provides two types of DC tunnel disconnection fault actions: **Exclusive direct connect tunnel disable** and **Shared direct connect tunnel disable**. The corresponding fault recovery actions are **Exclusive direct connect tunnel enable** and **Shared direct connect tunnel enable**. You can select the appropriate fault action based on your tunnel type. The following takes an exclusive DC tunnel as an example.
1. Log in to Tencent Smart Advisor > Chaotic Fault Generator and enter the **Experiment Management** page. Click **Create a New Experiment**, select **Dedicated Line - Exclusive direct connect tunnel** for the object type, and then click **Add Instance**.
2. After clicking **Add Instance**, you can filter your DC tunnels based on the search criteria.
3. After selecting the instance, click **Add experiment action**.
4. Select **Exclusive direct connect tunnel disable**.
5. The fault action will automatically bring out the corresponding recovery action:
If the recovery action is **Automatic Execution**, you can click the action and set pre-action and post-action waiting time to control the fault duration.
If the recovery action is **Manual Execution**, you can manually control the timing of the fault and recovery. Click **Next** to enter Global Configuration.
6. In Global Configuration, you can set the experiment execution method to Manual or Automatic. **The default is Manual**. Continue to **Add Monitoring Metrics**, which will refresh in real-time during the experiment execution process (There may be a 1~2 minute delay for different objects). Click **Submit** to enter Environmental Check.
7. **Environmental Check** will not execute the experiment. It checks only whether the status of your experiment objects meets the experiment requirements. For example, it checks if your DC tunnel version is 2.0.
8. For now, the experiment creation is complete. You can click **Experiment Details** to execute the experiment.

Step 2: Execute the experiment action

1. Click the Action Execution button.
2. Wait for the fault action to successfully execute, and meantime you can real-time observe the fault performance through monitoring metrics (Network inbound and outbound bandwidth dropping to 0).
3. Once the fault injection is complete, you can click **Execute recovery action** to recover the tunnel status at the appropriate time.

Step 3: Observe the results

After the fault, you can see the DC tunnel in a disabled status, with 100% package loss detected using the tunnel detection tool.
After the fault, verify the effectiveness of alarm delivery by viewing if the **TCOP Alert Strategy** is triggered and if the alarm is recovered after the fault recovery. You can also observe the overall fault injection and recovery effects using monitoring metrics.

Note:

A certain delay exists between monitoring metrics and actual faults.

Verifying the Disaster Recovery Capability of the High Availability DC Deployment Architecture (Taking Dual Lines with Dual Access Points as an Example)

Experiment Preparation

Dual lines with dual access points deployment architecture: The user's IDC is connected to two Tencent Cloud access points through two connections. The local router on the IDC side establishes BGP neighbor relationships with two DSR clusters via the BGP protocol. When a fault is detected on Connection 1, the system automatically switches traffic to Connection 2, ensuring normal business operations. After the fault is fixed, the traffic automatically switches back.

Experiment Steps

1. Create an experiment, select **Dedicated Line - Exclusive direct connect tunnel**, and click **Add Instance** to filter all DC tunnels by DC ID.
2. The subsequent steps for experiment creation are the same as that in 1.2 Experiment Steps. For details, see 1.2 Experiment Steps.

Result Observation

DC tunnel disconnection fault will disable the BGP sub-API of the tunnel, resulting in the inability to establish a BGP connection. After the fault, you can see that the BGP connection is in a disabled status in the console. If the tunnel is configured with BFD or NQA health detection, the corresponding detection will also fail. If your DC is configured with automatic convergence rules, the tunnel traffic should automatically switch to the backup DC. If 50% capacity is reserved in connection capacity planning, you can observe the inbound and outbound traffic of the DC doubling by monitoring the DC traffic.

Custom Actions
Expanding Fault Injection Actions with Custom Scripts

Last updated：2024-09-26 15:49:18

To meet users' personalized fault action needs, CFG has developed custom actions. You can package fault scripts as a fault action for experiment orchestration. During experiment execution, custom actions are distributed to the selected CVM for execution. Additionally, custom actions support dynamic parameter configuration and action resource sharing under the root account, allowing for creation once and multiple uses. Below is an introduction about how to use custom actions with custom scripts to kill the specified process.
During the operation of the business, processes may terminate due to various unexpected situations. To verify whether the service can automatically restart, you can use a custom script to simulate this scenario by killing the process.

Directions

Step 1: Create an action

Log in to Tencent Smart Advisor > Chaotic Fault Generator, select **Action Library Management**, and click **Create Custom Action**.

Step 2: Fill in action basic information

1. Fill in the basic information for the custom action:
Note:
Custom actions support two command types:
Shell: means Linux script commands.
Powershell: means Windows script commands.
Command content supports dynamic parameter replacement. Use {{}} to enclose parameters that need to be dynamically replaced in the script, then click **Use Parameters** to automatically extract dynamic parameters.

```
ps -ef | grep -w {{process_name}} | grep -v grep | awk '{print $2;}' | xargs kill -
```

2. Click **Save** to see the newly created custom action in the action list. If you have more granular permission management needs, you can specify tags during creation so that only sub-users under the same tag can share the custom action.

Step 3: Create an experiment

1. Enter the **Experiment Management** page and click **Create a New Experiment**.
2. Custom actions can participate in experiment orchestration. During experiment creation, click **Skip and create a blank experiment**, then fill in the experiment details and add experiment Instances**.**
3. Click **Add Now**, and select the experiment action to add.
4. After selecting the relevant custom action, click **Next**.
5. Set dynamic parameters. If not set, default values for creation will be used.

Custom actions will be distributed to the corresponding CVM instances. If multiple instances are chosen, the script can be distributed for batch parallel execution.

Step 4: Execute the experiment

After completing experiment orchestration, click **Execute Experiment**. If the action is executed successfully (Return code 0 indicates success.), you can log in to the server for verification.

Performing Single-Core CPU Stress Test with Custom Actions

Last updated：2024-09-26 15:49:18

Background

CFG allows users to customize fault actions for chaos engineering experiments. Users can configure action parameters and execute script commands based on specific scenarios.

Practical Examples

This section demonstrates how to configure custom actions to perform chaos engineering experiments for CVM single-core CPU stress test.

Step 1: Experiment preparation

A CVM instance that can be used for fault injection (The example instance runs TencentOS Server 2.6 (Final) system.)
Fault action script file

Step 2: Create custom actions

1. Log in to Tencent Smart Advisor > Chaotic Fault Generator, click Action Library Management, and select **Create Custom Action.**
2. Configure custom actions, and configure command content and command parameters. Click **Save** after completing the configuration.
3. After the above contents are saved, the corresponding custom actions will appear in the custom action list.
The example command content is as follows, where the percentage and timeout parameters can be passed during execution.

```
#!/bin/bash
user=$(whoami)
if [ !$user == 'root' ]
then
    sudo -i
fi
echo -e "[`date +"%Y-%m-%d %H:%M:%S"`] \\c"
echo "installing stress-ng..."
yum install stress-ng -y
```

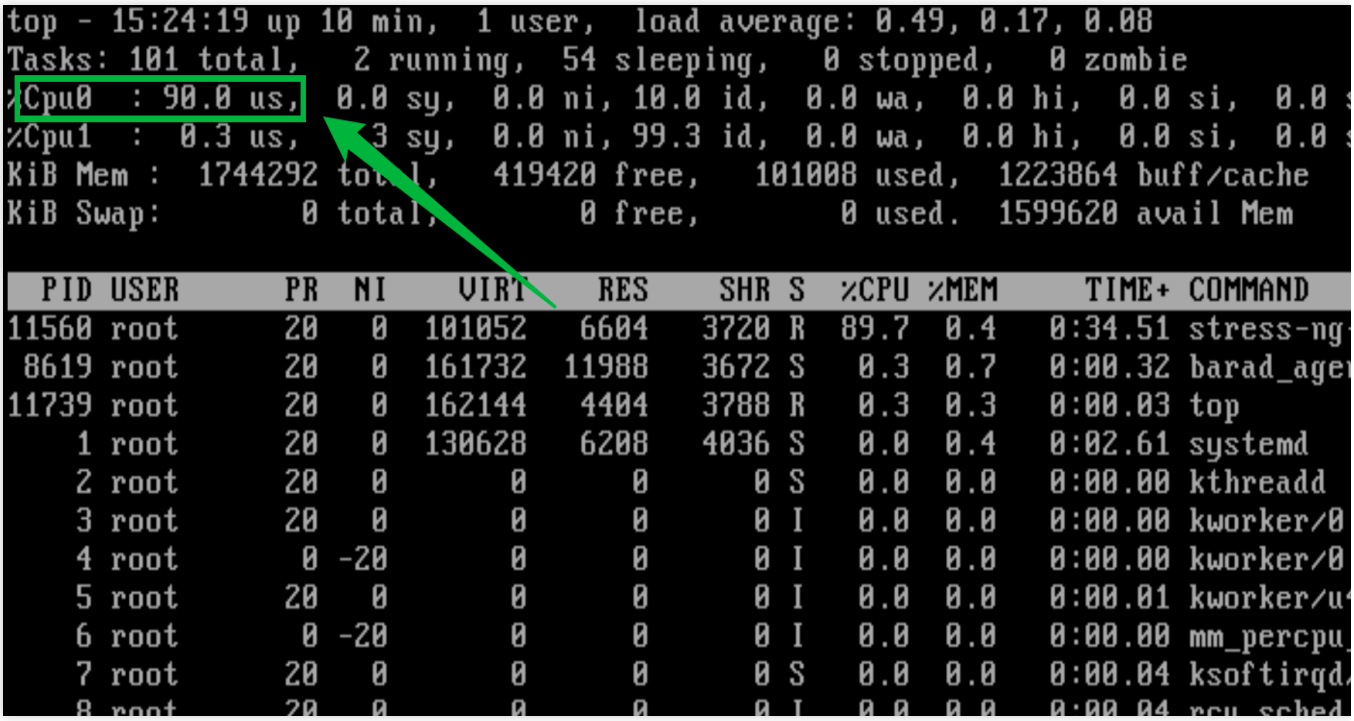
```
percentage={{percentage}}
timeout={{timeout}}

stress-ng -c 1 -l $percentage --timeout $timeout

if [[ !$? -eq 0 ]]
then
    echo -e "["`date +"%Y-%m-%d %H:%M:%S"`"] \\c"
    echo "Failed"
    exit 1
else
    echo -e "["`date +"%Y-%m-%d %H:%M:%S"`"] \\c"
    echo "Completed"
    exit 0
fi
```

Step 3: Experiment orchestration

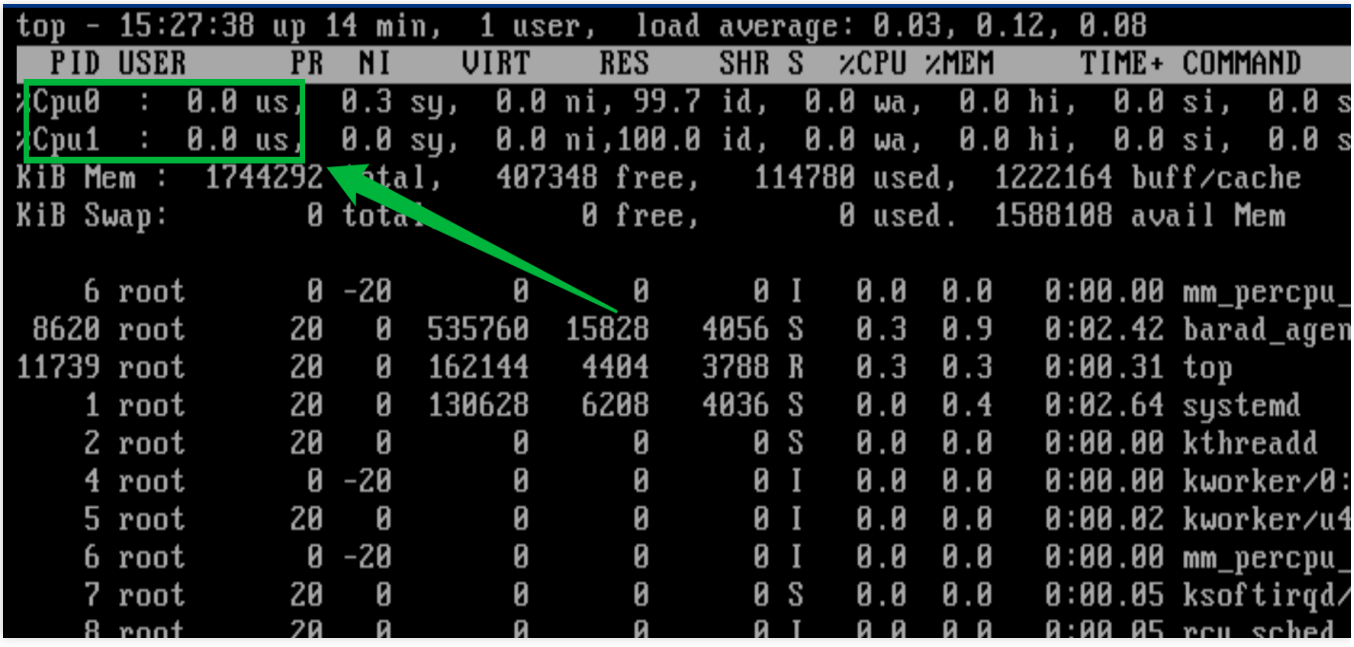
1. Enter the **Experiment Management** page and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment**, fill in the experiment details, and select the objective CVM instance for the experiment.
3. Add experiment actions. Click **Add Now** , select **Shell Script** , select the created **Custom action**, and click **Next** .
4. Leave the action parameters as default and click **Confirm**.
5. After configuring action parameters, click **Next**. After confirming all configurations, click **Submit** to complete experiment creation.

Step 4: Execute the experiment

1. Execute the experiment and observe the execution results.
2. Use the top command to view CPU. The CPU usage has reached the configured value.



3. The action will last for 60 seconds. After completing experiment execution, view the CPU usage. It has been recovered to normal.



Implementing CPU Accumulation Faults with Custom Actions

Last updated：2024-09-26 15:49:18

Background

The CFG allows users to create custom fault actions for chaotic experiments. Users can configure action parameters and execute script commands based on specific scenarios.

Practical Examples

This section demonstrates how to configure custom actions to perform chaos engineering experiments for CVM CPU accumulation faults.

Step 1: Experiment preparation

A CVM instance used for fault injection (The example instance runs TencentOS Server 2.6 (Final).)
Fault action script file

Step 2: Create custom actions

1. Log in to Tencent Smart Advisor > Chaotic Fault Generator, enter the Action Library Management page, and select **Create Custom Action.**
2. Configure the custom actions. Configure command content and command parameters. Click **Save** after completing the configuration.
The example command content is as follows, where the percentage, timeout, and step parameters can be passed during execution.

```
#!/bin/bash
user=$(whoami)
if [ !$user == 'root' ]
then
    sudo -i
fi
echo -e "["`date +"%Y-%m-%d %H:%M:%S"`"] \\c"
echo "installing stress-ng..."
yum install stress-ng -y

percentage={{percentage}}
```

```
timeout={{timeout}}
step={{step}}

for (( i = $step; i < ${timeout} + $step; i=(i+step) )); do
    val=`expr $percentage \\* $i / $timeout`
    echo "Pre = $val"
    stress-ng -c 0 -l $val --timeout $step
done

if [[ !$? -eq 0 ]]
then
    echo -e "["`date +"%Y-%m-%d %H:%M:%S"`"] \\c"
    echo "Failed"
    exit 1
else
    echo -e "["`date +"%Y-%m-%d %H:%M:%S"`"] \\c"
    echo "Completed"
    exit 0
fi
```

Step 3: Experiment orchestration

1. Enter the **Experiment Management** page and click **Create a New Experiment**.
2. Click **Skip and create a blank experiment**, fill in the experiment details, and select the objective CVM instance for the experiment.
3. Add experiment actions. Click **Add Now**, select **Shell Script** , select the created **Custom action**, and click **Next** .
4. Leave the action parameters as default and click **Confirm**.
5. After configuring action parameters, click **Next**. After confirming all configurations, add a monitoring metric to observe the action's effect, then click **Submit.**
6. Click **Submit** to complete the experiment creation.

Step 4: Execute the experiment

1. Execute the experiment and observe the execution results.
2. During execution, users can monitor CPU data changes via monitoring metrics. After waiting for 90 seconds, once the action is completed, you can see from the monitoring metrics that the experiment results meet the expectations.

Implementing CRS Connection Count Increase with Custom Actions

Last updated：2024-09-26 15:49:18

Background

CFG allows users to customize fault actions for chaos engineering experiments. In fault scenarios where stress test is executed on MySQL or CRS, a CVM instance within the same VPC as MySQL or CRS can be used to execute stress test scripts.

Practical Examples

The following demonstrates how to use custom actions to implement CRS connection count increase.

Step 1: Experiment preparation

A CVM instance used for fault injection with a Redis client installed, used to connect to the CRS instance (Ensure the CVM can be connected to the CRS instance.)
A CRS instance that is in the same VPC with the CVM
Fault action script file

Step 2: Create custom actions

1. Log in to Tencent Smart Advisor > Chaotic Fault Generator, enter the **Action Library Management** page, and select **Create Custom Action**.
2. Configure the custom actions. Configure command content and command parameters. Click **Save** after you complete the configuration.
The example command content includes parameters such as request_num, host, port, password, and client_count, which can be passed during execution.

```
redis-benchmark -q -n {{request_num}} -h {{host}} -p {{port}} -a '{{password}}' -c
```

Step 3: Experiment orchestration

1. Enter the **Experiment Management** page, click **Create a New Experiment**, fill in the experiment details, and select the objective CVM instance for the experiment.
2. Click **Add Experiment Action** in the lower-left corner, select **Shell Script**, select the custom action you created, and click **Next**.
3. Modify the default parameters of the action according to the stress test requirements.
4. After configuring action parameters, click **Next**. After filling in the experiment global configuration, click **Submit**.

Step 4: Execute the experiment

1. Execute the experiment and observe the execution results.
2. During execution, you can monitor data on the CRS console.

Injecting PowerShell Scripts for Windows Systems

Last updated：2024-09-26 15:49:18

Background

CFG supports multi-environment and multi-system instances. In addition to Linux systems using shell scripts, PowerShell scripts can also be used to inject commands for Windows systems.

Directions

Step 1: Create an experiment

1. Log in to the Tencent Smart Advisor > Chaotic Fault Generator, enter the **Experiment Management** page, click **Create a New Experiment**, and fill in the experiment details.
2. Select **Compute** as the resource type, select **CVM** as the resource object, and add the instance.

Step 2: Select the action

Click **Add Now** to add fault action. Select **Shell Script** from the experiment action category, select **Custom Action** as the platform action, and then click **Next.**

Step 3: Fill in custom action parameters

Select **powerShell** as the command type, and input the custom command text.

Step 4: Execute the action

Execute the experiment and observe the execution results.

Cloud Streaming Services (CSS) Stream Push Interruption

Last updated：2024-09-26 15:49:18

The stream push interruption fault scenario is used to verify situations where CSS stream push is unexpectedly interrupted. By observing the performance post-fault, you can verify if the stream push client has a retry mechanism and assess whether downstream playback is affected.

Experiment Preparation

For CSS online streams, go to CSS > Stream Management > Online Streaming to view.

Note:

1. Stream push interruption will **genuinely disconnect the live stream**. Please carefully evaluate the impact when conducting experiments in a production environment.
2. The stream push interruption fault is equivalent to manually operating **online stream disconnection on the CSS console.**
3. For primary-secondary streams, stream push interruption will **disconnect both primary and secondary streams** simultaneously. To disconnect one stream only, see the **Primary-secondary stream single path disconnection** fault.
4. Stream push interruption **does not have a recovery action** and requires the client to actively recover stream push.

Experiment Steps

Step 1: Create an experiment

1. Log in to Tencent Smart Advisor > Chaotic Fault Generator, and follow the steps in Quick Start Guide to create an experiment. In the action group configuration, select **Audio & Video** for **Instance Type**, and **CSS (Stream Push Domain Name)** for **Instance Object**. Click **Add Instance** to add the stream push domain name corresponding to the experiment stream.
2. After adding the domain name, click **Experiment Actions** and then **Add Now** to add the experiment action. Select the **Stream push interruption** fault action and configure the action parameters, where the **StreamName** field is the stream you will inject the fault into (supporting up to 20 different streams for concurrent operations).
3. After confirmation, click **Next** to enter the **Global Configuration** page, select **Execution Method** for the action, and configure **Guardrail Policy** and **Monitoring Metrics** .

Note:

Fault actions for CSS (stream push domain name) currently do not support configuring monitoring metrics.

4. Click **Submit** when it is completed, then click **Experiment Details**. The system will automatically pre-check the resources you added to determine if they meet the experiment execution conditions (Pre-check does not execute the experiment actions).

Stream push interruption will perform the following verification. Ensure the environment passes the pre-check before starting the experiment, or it may fail. After issues are fixed, you can click **Double-Check** to initiate the pre-check again.

Check whether the selected domain name is enabled and is the stream push domain name.

Confirm whether the specified stream is an online stream.

Step 2: Execute the experiment

Click **Execute** on the action to start executing the fault action.

Step 3: Observe results

During the execution of fault actions, you can go to CSS Console > Stream Management to confirm whether the stream is online.

If your client has a retry policy, the CSS stream will quickly recover. If the recovery is too fast, you might not notice the interruption effect. You can go to CSS > Monitoring > Stream Interruption Records to determine the specific disconnection time.

Additionally, you can view the impact on **Live Playback**, **User distribution**, **Top playbacks**, and **Origin Server** via CSS > Monitoring > Operation Analysis. For details, see Operation Analysis.

Stream Push Disabled

Last updated：2024-09-26 15:49:18

The stream push disabled scenario can verify scenarios where an online stream is blocked due to violations or other reasons. By observing the fault performance, you can verify whether **downstream playback** can be properly disconnected after the online stream is disabled. If you have configured Stream slate, you can also verify whether the CSS can switch to the slate after the stream is banned.

Experiment Preparation

For CSS online streams, you can go to CSS > Stream Management > Live Streaming to view.

Note:

1. Stream push disabled will **realistically disable CSS stream push**. Please carefully evaluate the impact when conducting experiments in a production environment.
2. Stream push disabled is equivalent to manually **disabling the online stream on the CSS console.**

Experiment Steps

Step 1: Create an experiment

1. Log in to Tencent Cloud Smart Advisor > Chaotic Fault Generator, and follow the steps in Quick Start Guide to create an experiment. In the action group configuration, select **Audio and Video** for **Instance Type**, and **CSS (Stream Push Domain Name)** for **Instance Object**. Click **Add Instance** to add the push domain name corresponding to the experiment stream.
2. After adding the domain name, click **Experiment Actions** and then **Add Now** to add the experiment action. Select the **Stream push disabled** fault action and configure the action parameters, where the **StreamName** field is the stream you will inject the fault into (supporting up to 20 different streams for concurrent operations).
3. After confirmation, click **Next** to enter the **Global Configuration** page, select **Execution Method** for the action, and configure **Guardrail Policy** and **Monitoring Metrics** .

Note:

Fault actions for CSS (stream push domain name) do not currently support configuring monitoring metrics.

4. Click **Submit** when it is completed, then click **Experiment Details**. The system will automatically pre-check the resources you added to determine if they meet the experiment execution conditions (Pre-check does not execute the experiment actions).

Stream push disabled will perform the following verification. Ensure the environment passes the pre-check before starting the experiment, or it may fail. After issues are fixed, you can click **Double-Check** to initiate the pre-check

again.
Check whether the selected domain name is enabled and is a stream push domain name.
Confirm whether the specified stream is an online stream.

Step 2: Execute the experiment

Click **Execute** on the action to start executing the fault action.

Step 3: Observe results

1. After the stream push is banned, you will not see the disabled online streams on the CSS Console > Stream Management > Live Streaming page.
2. On the **Disabled Streams** page, you can see the disabled online streams.
3. The downstream playback will be interrupted immediately. If you have configured a stream slate, the CSS screen will immediately switch to the stream slate.
4. You can view the impact on **Live Playback**, **User distribution**, **Top playbacks**, and **Origin Server** via CSS > Business Monitoring > Operational Analysis. For details, see Operation Analysis.

Step 4: Fault recovery

In the CFG platform, click **Recover Action** to cancel the disablement. After successful execution, check if the items observed in **Step 3** have been recovered.

Note:

You may need to **restart stream push** to recover the online stream.

Primary-Secondary Stream Switch

Last updated：2024-09-26 15:49:18

The primary-secondary stream switch fault scenario can simulate primary-secondary stream switch caused by stream push anomalies. By observing the post-fault behavior, you can verify whether **downstream playback** is normal after the primary-secondary stream switch and help evaluate the impact duration of primary-secondary switch on downstream playback.

Experiment Preparation

For CSS online streams, you can go to CSS > Stream Management > Online Streaming to view.

Note:

1. Primary-secondary stream switch will **genuinely switch the primary-secondary streams**. Please carefully evaluate the impact when conducting experiments in a production environment.
2. The primary-secondary stream switch fault is equivalent to manually operating the **primary-secondary stream switch** on the CSS console.
3. The primary-secondary stream switching fault requires the primary-secondary streams to have **downstream playback**, otherwise **the switching will fail**.

Experiment Steps

Step 1: Create an experiment

1. Log in to Tencent Smart Advisor > Chaotic Fault Generator, and follow the steps in Quick Start Guide to create an experiment. In the action group configuration, select **Audio and Video** for **Instance Type**, and **CSS (Stream Push Domain Name)** for **Instance Object**. Click **Add Instance** to add the push domain name corresponding to the experiment stream.
2. After adding the domain name, click **Experiment Actions** and then **Add Now** to add the experiment action. Select the **Primary-secondary stream switch** fault action and configure the action parameters, where the **StreamName** field is the stream you will inject the fault into (supporting up to 20 different streams for concurrent operations).
3. After confirmation, click **Next** to enter the **Global Configuration** page, select the **Execution Method** for the action, configure **Guardrail Policy**, and set up **Monitoring Metrics**.

Note:

Fault actions for CSS (stream push domain name) do not currently support configuring monitoring metrics.

4. Click **Submit** when it is completed, then click **Experiment details**. The system will automatically pre-check the resources you added to determine if they meet the experiment execution conditions (Pre-check does not execute the

experiment actions).

Stream push disabled will perform the following verification. Ensure the environment passes the pre-check before starting the experiment, or it may fail. After issues are fixed, you can click **Double-Check** to initiate the pre-check again.

Pre-check items for the primary-secondary stream switch:

Check whether the selected domain name is enabled and is a stream push domain name.

Confirm whether the specified stream is an online stream.

Step 2: Execute the experiment

Click **Execute** on the action to start executing the fault action.

Step 3: Observe results

1. After performing primary-secondary stream switch, enter CSS Console > Stream Management, and click **Primary-secondary Stream** to observe the current status of the primary-secondary streams.
2. In CSS Console > Business Monitoring > Exception Events, click **Primary-secondary Stream Events** to view **Proactive Switchover** abnormal events.
3. After switching, you can simultaneously observe the changes in downstream playback, which typically switches within 1 to 3 seconds.
4. You can view the impact on **Live Playback**, **User distribution**, **Top playbacks**, and **Origin Server** via CSS > Business Monitoring > Operational Analysis. For details, see Operation Analysis.

Step 4: Fault Recovery

In the CFG platform, click **Recover Action** to switch back the primary-secondary streams. After successful execution, check if the items observed in **Step 3** have been recovered.

Primary-secondary Stream Single Path Interruption

Last updated：2024-09-26 15:49:18

Primary-secondary stream single path interruption simulates the scenario where the primary stream is interrupted unexpectedly in the primary-secondary stream push scenario. By observing the post-fault behavior, you can verify whether the secondary stream can be switched normally and the impact on downstream playback.

Experiment Preparation

For CSS online streams, you can go to CSS > Stream Management > Online Streaming to view.

Note:

1. Primary-secondary stream single path interruption will **genuinely disconnect one of the streams** in the primary-secondary stream line. Please carefully evaluate the impact when conducting experiments in a production environment.
2. Stream push disabled is equivalent to manually operating **primary-secondary stream single path disconnection on the CSS console.**
3. Primary-secondary stream single path interruption **does not have a recovery action** and requires the client to actively recover stream push.

Experiment Steps

Step 1: Create an experiment

1. Log in to Tencent Cloud Smart Advisor > Chaotic Fault Generator, and follow the steps in Quick Start Guide to create an experiment. In the action group configuration, select **Audio and Video** for **Instance Type**, and **CSS (Stream Push Domain Name)** for **Instance Object**. Click **Add Instance** to add the push domain name corresponding to the experiment stream.
2. After adding the domain name, click **Experiment action** and then **Add Now** to add the experiment action. Select the **Primary-secondary stream single path interruption** fault action and configure the action parameters, where the **StreamName** field is the stream you will inject the fault into (supporting up to 20 different streams for concurrent operations).
3. After confirmation, click **Next** to enter the **Global Configuration** page, select the **Execution Method** for the action, and configure **Guardrail Policy** and **Monitoring Metrics**.

Note:

Fault actions for CSS (stream push domain name) do not currently support configuring monitoring metrics.
4. Click **Submit** when it is completed, and after creating the experiment, click **Experiment details**. The system will automatically pre-check the resources you added to determine if they meet the experiment execution conditions (Pre-check does not execute the experiment actions). **Primary-secondary stream single path interruption** will perform the following verification. Ensure the environment passes the pre-check before starting the experiment, or it may fail. After issues are fixed, you can click **Double-Check** to initiate the pre-check again.
Pre-check items for primary-secondary stream single path interruption:
Check whether the selected domain name is enabled and is a stream push domain name.
Confirm whether the specified stream is an online stream.

Step 2: Execute the experiment

Click **Execute** on the action to start executing the fault action.

Step 3: Observe results

1. After the primary-secondary stream disconnection, enter CSS Console > Stream Management page, click **Primary and Secondary Stream**, and you can see the **Primary and Secondary Stream** relationship has changed.
2. In CSS Console > Business Monitoring > Exception Events, click **Primary and Secondary Stream Events** to view **Proactive Switchover** abnormal events.
3. **Downstream Playback** will automatically switch to the replica stream after the primary stream disconnects, which typically switches within 1 to 3 seconds.
4. You can view the impact on **Live Playback**, **User distribution**, **Top playbacks**, and **Origin Server** via CSS > Business Monitoring > Operation Analysis. For details, see Operation Analysis.