

目 录

1	测试第一种情况	2
1.1	第一种情况的描述	2
1.2	测试的思路	2
1.3	测试需要的环境	3
1.4	测试的代码	6
2	测试第二种情况	7
2.1	第二种情况的描述	7
2.2	测试的思路	8
2.3	测试代码	8

1 测试第一种情况

1.1 第一种情况的描述

openstack 上如果一台虚拟机从来没有创建过快照，那么它的快照链为 `base<-overlay1`。我们现在想测试的是，如果此时对虚拟机做动态快照，是否会对虚拟机上进程的执行产生影响，是否会对在处理中的数据的完整性造成影响。

所以我们现在想要测试的是，如果虚拟机上运行着一个脚本，这个脚本的功能是向 `num.txt` 文件中写入 1 ~ 4000000 个数字，每个数字一行。此时我们对这台虚拟机进行动态快照，查看脚本是否正常运行，然后查看新建的增量文件的磁盘上数据的完整性，并且查看原来的磁盘文件上的数据是否中断。

1.2 测试的思路

首先我介绍一下当前环境: `libvirt` 中定义了两个虚拟机 `overlay1` 和 `overlay2`。`overlay1` 的磁盘文件是 `overlay1.qcow2`，`overlay2` 的磁盘文件也是 `overlay1.qcow2`。`overlay1.qcow2` 的后端镜像是 `base.qcow2`。

`overlay1.qcow2` 上存放这 `test_in_vm.sh`、`monitor.sh` 和 `judge.sh` 这三个脚本文件。这个三脚本文件的功能分别是：

```
1 test_in_vm.sh: 将1~4000000这些数字写入num.txt文件中，每个数字一行。
2 monitor.sh: 这个文件用于监测test_in_vm.sh这个进程是否运行结束。
3 judge.sh: 这个文件用于判断num.txt是否有4000000，用于判断数据的完整性。
```

接下来我说一下测试方案：

1. 首先启动虚拟机 `overlay1`，注意它的磁盘文件为 `overlay1.qcow2`。
2. 在宿主机环境中，`ssh` 登录到 `overlay1` 上，执行上面的 `test_in_vm.sh` 脚本。
3. 动态创建快照，此时虚拟机 `overlay1` 的磁盘文件就变成了 `overlay2.qcow2`。
4. 等待 `overlay1` 上的脚本执行结束。随后查看 `overlay1` 上的数据是否完整，也就是查看 `overlay2.qcow2` 磁盘上的数据是否完整。
5. 启动虚拟机 `overlay2`，注意它的磁盘文件为 `overlay2.qcow2`。随后查看它上面的数据是否不完整，也就是查看 `overlay1.qcow2` 磁盘上的数据是否不完整。
6. 关闭虚拟机 `overlay1` 和 `overlay2`。
7. 清除之前的改动，返回到最初的环境，也就是要把 `overlay2.qcow2` 这个增量文件删除，将虚拟机 `overlay1` 的磁盘文件更换为原来的 `overlay1.qcow2`，将虚拟机 `overlay1` 的磁盘文件更换为 `overlay1.qcow2`，并且删除虚拟机 `overlay1` 的快照数据。

1.3 测试需要的环境

首先创建一个 base.qcow2 镜像：

```
1 qemu-img create -f qcow2 base.qcow2 50G
```

随后在 base.qcow2 镜像上安装 ubuntu 系统，随后在这个系统上安装 ssh：

```
1 sudo apt install openssh-server
```

然后在这个镜像的基础上创建增量镜像 overlay1.qcow2：

```
1 qemu-img create -f qcow2 -b base.qcow2 overlay1.qcow2
```

在 overlay1.qcow2 上启动虚拟机 overlay1,overlay1 的 xml 配置文件内容如下：

```
1 <domain type='kvm'>
2   <name>overlay1</name>
3   <memory>1048576</memory>
4   <currentMemory>1048576</currentMemory>
5   <vcpu>4</vcpu>
6   <os>
7     <type arch='x86_64' machine='pc'>hvm</type>
8     <boot dev='cdrom' />
9   </os>
10  <features>
11    <acpi/>
12    <apic/>
13    <pae/>
14  </features>
15  <clock offset='localtime' />
16  <on_poweroff>destroy</on_poweroff>
17  <on_reboot>restart</on_reboot>
18  <on_crash>destroy</on_crash>
19  <devices>
20    <emulator>/usr/bin/qemu-system-x86_64</emulator>
21    <disk type='file' device='disk'>
22      <driver name='qemu' type='qcow2' />
23      <source file='/home/pengsida/kvm/openstack/overlay1.qcow2' />
24      <target dev='hda' bus='ide' />
25    </disk>
26    <interface type='network'>
27      <source network='default' />
28    </interface>
29    <interface type='network'>
30      <source network='default' />
31    </interface>
32    <input type='mouse' bus='ps2' />
33    <graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0' keymap='
34      en-us' />
35    <channel type='unix'>
36      <source mode='bind' path='/var/lib/libvirt/qemu/fl6x86_64.agent' />
37      <target type='virtio' name='org.qemu.guest_agent.0' />
38    </channel>
```

```

38     </devices>
39 </domain>

```

在宿主机上安装 qemu-guest-agent:

```

1  sudo apt install qemu-guest-agent

```

随后启动虚拟机 overlay1:

```

1  # 在overlay1.xml目录下
2  sudo virsh define overlay1.xml
3  sudo virsh start overlay1

```

在虚拟机中安装 qemu-guest-agent:

```

1  sudo apt install qemu-guest-agent

```

然后在 overlay1 上创建两个 test_in_vm.sh、monitor.sh 和 judge.sh 这三个脚本文件。

test_in_vm.sh 脚本内容如下:

```

1  #!/bin/bash
2
3  times=2000
4
5  rm num.txt
6
7  num=1
8
9  for ((i=0;i<=times;i++))
10 do
11     for ((j=0;j<=times;j++))
12     do
13         echo $num >> num.txt
14         num=$((num+1))
15     done
16 done

```

monitor.sh 脚本内容如下:

```

1  #!/bin/bash
2
3  test='ps aux | grep ./test.sh'
4  num='echo $test | awk -v RS="/bin/bash ./test.sh" 'END {print —NR}''
5
6  while (( $num >= 1 ))
7  do
8      echo "./test.sh is running"
9      sleep 5
10     test='ps aux | grep ./test.sh'
11     num='echo $test | awk -v RS="/bin/bash ./test.sh" 'END {print —NR}''
12 done

```

```

13
14 echo "./test.sh is done"

```

judge.sh 脚本内容如下:

```

1  #!/bin/bash
2
3  num='awk 'END {print NR}' num.txt '
4
5  if (( num == 4000000))
6  then
7      echo "data is complete"
8  else
9      echo "data is not complete"
10 fi

```

随后在宿主机上创建 address.sh 脚本文件, 这个脚本文件可以根据虚拟机的名字返回虚拟机的 ip 地址:

```

1  #!/bin/bash
2
3  MAC='awk '/virbr0/ { print $4 }' /proc/net/arp '
4  VM=$1
5
6  line='virsh dumpxml $VM | grep "mac address"'
7  address='echo $line | awk -F " " '
8  {
9      for (i=1; i<=NF; i++)
10     {
11         if ($i ~ /\'\'/)
12             break
13     }
14     i++
15     str=""
16     for (; i<=NF; i++)
17     {
18         if ($i ~ /\'\'/)
19             break
20         temp=$i
21         str=(str temp)
22     }
23     print str
24 }',
25
26 ip='awk '/'$address'/ {print $1}' /proc/net/arp '
27
28 echo $ip

```

首先 ssh 登录到虚拟机 overlay1:

```

1  # 使用address.sh脚本获得overlay1虚拟机的ip地址
2  overlay1='./address.sh overlay1 '
3  echo $overlay1 # 如果输出为空, 需要重复上面的命令
4  ssh pengsida@$overlay1 # 假设虚拟机的用户名是pengsida

```

随后关闭虚拟机 overlay1:

```
1 sudo virsh destroy overlay1
```

随后开启虚拟机 overlay2:

```
1 sudo virsh define overlay2.xml
2 sudo virsh start overlay2
```

同样是 ssh 登录到虚拟机 overlay2:

```
1 overlay2='./address.sh overlay2'
2 echo $overlay2 # 如果输出为空, 需要重复上面的命令, 直到获得overlay2的ip
3 ssh pengsida@$overlay2
```

随后关闭虚拟机 overlay2:

```
1 sudo virsh destroy overlay2
```

这里之所以要 ssh 登录到虚拟机 overlay1 和 overlay2 的原因是需要信任未连接过的 ip 地址, 如下图所示:

```
pengsida@scholar:~$ ssh pengsida@$overlay1
The authenticity of host '192.168.122.234 (192.168.122.234)' can't be established.
ECDSA key fingerprint is SHA256:iEDukuHZ+pvJ+VrmS2mst3b1+C0eP20ZxW55WSUPgXw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.122.234' (ECDSA) to the list of known hosts.
```

1.4 测试的代码

在宿主机上创建 test.sh 脚本, 内容如下:

```
1 #!/bin/bash
2
3 # 步骤一
4 # 创建虚拟机overlay1, 磁盘文件为overlay1.qcow2
5 sudo virsh start overlay1
6 # 等待虚拟机启动
7 sleep 30
8
9
10 # 步骤二
11 # 获取虚拟机ip地址
12 overlay1='./address.sh overlay1'
13 # ssh登录到overlay1上, 执行上面的test.sh脚本
14 sshpass -p p1111111 ssh pengsida@$overlay1 ./test_in_vm.sh &
15
16 # 先让./test.sh脚本运行5秒
17 sleep 5
18
19
20 # 步骤三
```

```

21 # 动态创建快照，此时虚拟机overlay1的磁盘文件变为overlay2.qcow2
22 sudo virsh snapshot--create--as overlay1 snap snap--desc --disk-only --diskspec
    hda,snapshot=external,file=/home/pengsida/kvm/openstack/overlay2.qcow2 --
    atomic
23
24 sleep 5
25
26
27 # 步骤四
28 # 等待test.sh脚本执行结束
29 sshpass -p p1111111 ssh pengsida@$overlay1 ./monitor.sh
30 # 判断overlay1.qcow2上的数据是否完整
31 sshpass -p p1111111 ssh pengsida@$overlay1 ./judge.sh
32
33
34 # 步骤五
35 # 创建虚拟机overlay2，磁盘文件为overlay1.qcow2
36 sudo virsh start overlay2
37 # 等待虚拟机启动
38 sleep 60
39 # 获取虚拟机ip地址
40 overlay2='./address.sh overlay2'
41 # 判断overlay1.qcow2上的数据是否完整
42 sshpass -p p1111111 ssh pengsida@$overlay2 ./judge.sh
43
44
45 # 步骤六
46 sudo virsh destroy overlay1
47 sudo virsh destroy overlay2
48
49
50 # 步骤七
51 sudo virsh snapshot--delete overlay1 snap --metadata
52 sudo rm /home/pengsida/kvm/openstack/overlay2.qcow2
53
54 # 将虚拟机overlay1的磁盘文件更换为overlay1.qcow2
55 sudo virsh detach--disk overlay1 hda --persistent
56 sudo virsh attach--disk overlay1 /home/pengsida/kvm/openstack/overlay1.qcow2 hda
    --subdriver qcow2 --persistent

```

2 测试第二种情况

2.1 第二种情况的描述

openstack 上一台虚拟机被动态创建快照以后，它的快照链为 base.qcow2<-overlay1.qcow2<-overlay2.qcow2。我们现在想测试的是，如果此时对虚拟机做动态快照，并且缩短快照链，是否会对虚拟机上进程的执行产生影响，是否会对在处理中的数据的完整性造成影响。

所以我们现在想要测试的是，如果虚拟机上运行着一个脚本，这个脚本的功能是向

num.txt 文件中写入 1 ~ 4000000 个数字，每个数字一行。此时我们对这台虚拟机进行动态快照，并且缩短这台虚拟机的快照链，查看脚本是否正常运行，然后查看新建的增量文件的磁盘上数据的完整性，并且查看原来的磁盘文件上的数据是否中断。

2.2 测试的思路

测试方案如下：

1. 首先启动虚拟机 overlay2, 注意它的磁盘文件为 overlay2.qcow2, 虚拟机的快照链为 base.qcow2<-overlay1.qcow2<-overlay2.qcow2。
2. 在宿主机环境中, ssh 登录到 overlay2 上, 执行上面的 test_in_vm.sh 脚本。
3. 动态创建快照, 此时虚拟机 overlay2 的磁盘文件就变成了 overlay3.qcow2, 此时它的快照链为 base.qcow2<-overlay1.qcow2<-overlay2.qcow2<-overlay3.qcow2。然后缩短虚拟机的快照链, 此时它的快照链为 base.qcow2<-overlay1.qcow2<-overlay3.qcow2。
4. 等待 overlay2 上的脚本执行结束。查看 overlay2 上的数据是否完整, 也就是查看 overlay3.qcow2 磁盘上的数据是否完整。
5. 启动虚拟机 overlay3, 注意它的磁盘文件为 overlay2.qcow2。随后查看它上面的数据是否不完整, 也就是查看 overlay2.qcow2 磁盘上的数据是否不完整。
6. 关闭虚拟机 overlay2 和 overlay3。
7. 清除之前的改动, 返回到最初的环境, 也就是将新生成的增量文件 overlay3.qcow2 重命名为 overlay2.qcow2。随后删除之前生成的快照元数据。因为动态创建快照以后虚拟机 overlay2 的磁盘文件变成了 overlay3.qcow2, 我们还需要将虚拟机 overlay2 的磁盘文件更换为 overlay2.qcow2。

2.3 测试代码

```
1  #!/bin/bash
2
3  # 步骤一
4  # 创建虚拟机 overlay2, 磁盘文件为 overlay2.qcow2, 它的镜像链为 base.qcow2<-
   overlay1.qcow2<-overlay2.qcow2
5  virsh start overlay2
6  # 等待虚拟机启动
7  sleep 30
8
9
10 # 步骤二
11 # 获取虚拟机 ip 地址
```



```

12 overlay1='./address.sh overlay2 '
13 # ssh登录到overlay1上, 执行上面的test.sh脚本
14 sshpass -p p1111111 ssh pengsida@$overlay2 ./test_in_vm.sh &
15
16 # 先让脚本运行3秒
17 sleep 3
18
19
20 # 步骤三
21 # 动态创建快照, overlay2的磁盘文件变为overlay3.qcow2
22 virsh snapshot-create-as overlay2 snap snap-desc --disk-only --diskspec hda,
    snapshot=external, file=/home/pengsida/kvm/openstack/overlay3.qcow2 --atomic
23
24 # 缩短快照链, 这里只能动态缩短快照链
25 virsh blockcommit --domain overlay2 hda --base overlay1.qcow2 --top overlay2.
    qcow2 --wait --verbose
26
27 sleep 5
28
29
30 # 步骤四
31 # 等待test.sh脚本执行结束
32 sshpass -p p1111111 ssh pengsida@$overlay2 ./monitor.sh
33 # 判断overlay1.qcow2上的数据是否完整
34 sshpass -p p1111111 ssh pengsida@$overlay2 ./judge.sh
35
36
37 # 步骤五
38 # 创建虚拟机overlay3, 磁盘文件为overlay2.qcow2
39 virsh start overlay3
40 # 等待虚拟机启动
41 sleep 60
42 # 获取虚拟机ip地址
43 overlay3='./address.sh overlay3 '
44
45 # 判断overlay1.qcow2上的数据是否完整
46 sshpass -p p1111111 ssh pengsida@$overlay3 ./judge.sh
47
48 # 步骤六
49 # 关闭虚拟机overlay2和overlay3
50 virsh destroy overlay2
51 virsh destroy overlay3
52
53 # 步骤七
54 # 因为之前的overlay2.qcow2在合并中被删除了, 所以现在直接将新生成的overlay3.
    qcow2重命名为overlay2.qcow2, 它的后端镜像仍然是overlay1.qcow2
55 sudo mv /home/pengsida/kvm/openstack/overlay3.qcow2 /home/pengsida/kvm/
    openstack/overlay2.qcow2
56 # 删除之前生成的快照元数据
57 virsh snapshot-delete overlay2 snap --metadata
58 # 因为动态创建快照以后overlay2的磁盘文件变成了overlay3.qcow2
59 # 将虚拟机overlay2的磁盘文件更换为overlay2.qcow2
60 virsh detach-disk overlay2 hda --persistent
61 virsh attach-disk overlay2 /home/pengsida/kvm/openstack/overlay2.qcow2 hda --
    subdriver qcow2 --persistent

```