虛擬化容錯技術

Virtualization-based fault tolerance system

安裝與開發指南說明書

工研院 | 資訊與通訊研究所

中華民國 105 年 11 月

一、 技術項目簡介

Virtualization-based fault tolerance system is shown in Fig. 1. It can provide uninterrupted VM services when VM stops unexpectedly. Consistent states of virtual machine are repeatedly synchronized to backup physical machine. We name the running VM as master VM, the backup on backup physical machine as slave VM. When unexpected events happened, the slave VM can take over all ongoing jobs of the master VM because virtualization-based fault tolerance system provides a consistent view of service for clients. Any results on master VM without synchronizing to slave VM will not be exposed to master VM. If the hardware failure happened, the slave VM can take over the master VM's job and provide uninterrupted services for clients.

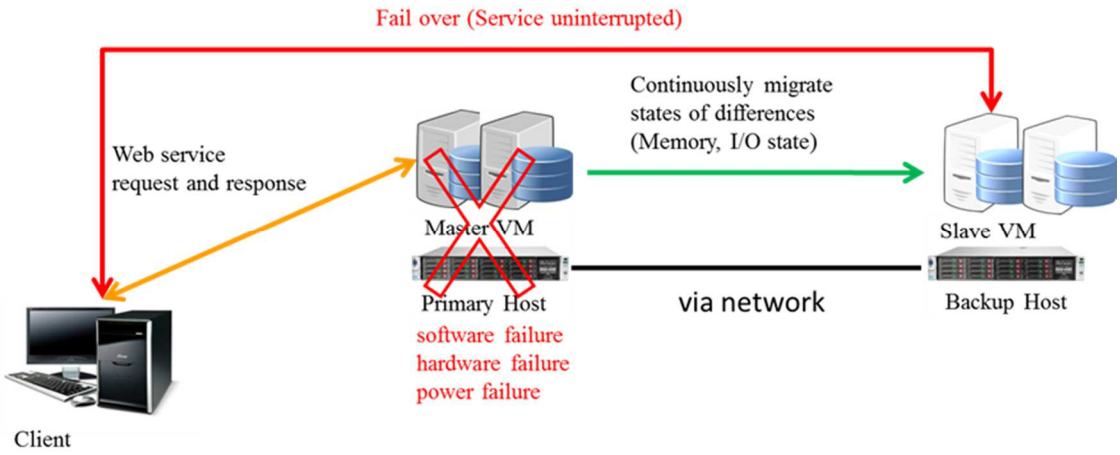

Fig. 1. Virtualization-based fault tolerance system

二、 安裝指南說明

1.Install Ubuntu 12.04

- [ubuntu-12.04.1-desktop-amd64.iso]

select English install

- sudo without password on Ubuntu

sudo visudo

Add this line at the end

[your username] ALL=(ALL) NOPASSWD: ALL

Ctrl-X to leave, save your changes, and you're done!

2.apt-get related tools

sudo apt-get update
sudo apt-get install vim gcc make gdb fakeroot build-essential kernel-package libncurses5 \
libncurses5-dev zlib1g-dev libglib2.0-dev qemu xorg bridge-utils openvpn vncviewer

3.Install Linux kernel 3.5

[linux-3.5.tar.gz]

tar jxf linux-3.5.tar.gz
cd linux-3.5
cp /boot/config-`uname -r` ./.config
make menuconfig
Disable following item:
Processor type and features
  [ ] x86 PAT support

Kernel hacking
    [ ] Filter access to /dev/mem

make-kpkg clean
fakeroot make-kpkg --initrd kernel_image kernel_headers -j8
cd ~
sudo dpkg -i linux-image-3.5.0_3.5.0-10.00.Custom_amd64.deb
sudo dpkg -i linux-headers-3.5.0_3.5.0-10.00.Custom_amd64.deb

4.Set grub

/etc/default/grub

GRUB_DEFAULT='Ubuntu, with Linux 3.5.0'

GRUB_HIDDEN_TIMEOUT_QUIET=true

GRUB_TIMEOUT=10

sudo update-grub

reboot

5.Set nfs

- In NFS server

sudo apt-get install nfs-kernel-server

/etc/exports

/home/[your username]/nfsfolder *(rw,no_root_squash,no_subtree_check)

put [Ubuntu20G-1604.tar.gz] or [Ubuntu20G32bits.tar.gz] in /home/[your username]/nfsfolder and

tar zxf Ubuntu20G32bits.tar.gz

put your qemu source code on /home/[your username]/nfsfolder

- In Primary and Backup side /etc/fstab

[your NFS server IP]:/home/[your username]/nfsfolder /mnt/nfs nfs auto 0 0

- mount

mount -a

6.Primary and Backup side configure

- /etc/qemu-ifup, /etc/qemu-ifdown add exit on first line to avoid to use it

#!/bin/sh
exit
...

- vim /etc/network/interfaces

```
auto br0
#iface br0 inet dhcp
iface br0 inet static
bridge_ports eth0
bridge_maxwait 0
address [your eth0 IP]
netmask [your eth0 netmask]
gateway [your eth0 gateway]
dns-nameservers [your eth0 dns-server]

auto eth0
iface eth0 inet static
address 0.0.0.0

auto eth1
iface eth1 inet static
address [192.168.x.x]
netmask [255.255.255.0]
```

and execute

```
# sudo /etc/init.d/networking restart
```

- Every time boot host need to execute

./pre.sh

```
#!/bin/sh
WORKSPACE=/home/[your username]

sudo $WORKSPACE/reinsmodkvm.sh
sudo $WORKSPACE/set-net2.sh
sudo $WORKSPACE/set-vmft.sh
```

The following script is: (compile kvm first, compile method on next section)

- reinsmodkvm.sh

```
#!/bin/sh
```

```
cd /mnt/nfs/qemu/kvm
lsmod | grep kvm
sudo rmmod kvm_intel
sudo rmmod kvm
sudo insmod ./x86/kvm.ko
sudo insmod ./x86/kvm-intel.ko
lsmod | grep kvm
```

- set-net2.sh

```
#!/bin/sh

/usr/sbin/openvpn --mktun --dev tap0 --user `id -un`
#ip tuntap add tap0 mode tap
ifconfig tap0 promisc up
brctl addif br0 tap0

/usr/sbin/openvpn --mktun --dev tap1 --user `id -un`
#ip tuntap add tap1 mode tap
ifconfig tap1 promisc up
brctl addif br0 tap1
```

- set-vmft.sh

```
#!/bin/sh

mkdir /dev/cgroup
mount -t cgroup -o cpuset cpuset /dev/cgroup/
mkdir /dev/cgroup/vmft/
echo 7 > /dev/cgroup/vmft/cpuset.cpus
echo 1 > /dev/cgroup/vmft/cpuset.cpu_exclusive
echo 0 > /dev/cgroup/vmft/cpuset.mems
```

7.Compile Chiyun

- qemu

```
# cd /mnt/nfs/qemu
# ./configure --disable-pie --enable-kvm
# make clean
```

make -j8

- kvm

cd /mnt/nfs/qemu/kvm
./configure
make clean
make -j8

and use reinsmodkvm.sh to insmod kvm

8.Boot guest

cd /mnt/nfs/qemu

- Primary side

run_ft

sudo gdb -ex 'set confirm off' --args x86_64-softmmu/qemu-system-x86_64 \
-drive
if=none,id=drive0,cache=none,aio=native,format=raw,file=/mnt/nfs/Ubuntu20G-1604.img -device virtio-blk,drive=drive0,scsi=off \
-m 2G -enable-kvm -net tap,ifname=tap0 -net
nic,model=virtio,vlan=0,macaddr=ae:ae:00:00:00:20 -ft-join-port 5000 -vga std \
-chardev socket,id=mon,path=/mnt/nfs/chiyun-nfs/vm1.monitor,server,nowait -mon chardev=mon,id=monitor,mode=readline -gft-id 0 -smp 1
./run_ft
(gdb) r

Boot VM use boot.sh

sudo echo " " | sudo nc -U /mnt/nfs/vm1.monitor
# ./boot.sh

wait the guest boot, you can use ssh login or

vncviewer :5900

ID:root

PW:root

If you want to boot 2nd VM run_ft2

```
sudo gdb -ex 'set confirm off' -ex run --args x86_64-softmmu/qemu-system-x86_64 \
-drive
if=none,id=drive1,cache=none,aio=native,format=raw,file=/mnt/nfs/Ubuntu20G-1604-2.img -device virtio-blk,drive=drive1,scsi=off \
-m 2G -enable-kvm -net tap,ifname=tap1 -net
nic,model=virtio,vlan=0,macaddr=ae:ae:00:00:00:32 -ft-join-port 5003 -vga std \
-chardev socket,id=mon,path=/mnt/nfs/chiyun-nfs/vm2.monitor,server,nowait -mon
chardev=mon,id=monitor,mode=readline -gft-id 1 -smp 1
```

The different part is

```
id=drive1
file=/mnt/nfs/Ubuntu20G-1604-2.img
ifname=tap1
macaddr=ae:ae:00:00:00:21
-ft-join-port 5003
path=/mnt/nfs/chiyun-nfs/vm2.monitor
-gft-id 1
```

9.Start FT mode

- In Backup side

recv

```
sed -e 's/mode=readline/mode=readline -incoming tcp\:0\:4441,ft_mode/g' -e
's/vm1.monitor/vm1r.monitor/g' ./run_ft > tmp.sh
chmod +x ./tmp.sh
./tmp.sh
# ./recv
(gdb) r
```

In Primary side Host

- start ft mode

ftmode.sh

sudo echo "migrate -k tcp:[backup side ip address]:4441," | sudo nc -U /home/[your username]/vm1.monitor
# ./ftmode.sh

according to the backup side ip address and port

Guest grub

/etc/default/grub

GRUB_CMDLINE_LINUX_DEFAULT="video=efifb:off"

sudo update-grub

Remote Storage Server

- Start Remote Storage Server script on NFS server

sudo gdb -ex 'set confirm off' --args x86_64-softmmu/qemu-system-x86_64 -drive if=none,id=drive0,cache=none,aio=native,format=raw,file=[PATH of Disk Image] -device virtio-blk,drive=drive0,scsi=off -m 2G -
enable-kvm -net tap,ifname=tap0 -net nic,model=virtio,vlan=0,macaddr=ae:ae:00:00:00:50 -ft-join-port 5000 -vga std -chardev socket,id=mon,path=[PATH of Monitor]vm1.monitor,server,nowait -mon chardev=mon,id=monitor,mode=readline -gft-id 0 -blk-server-listen :5000

Just like start a VM but Add **-blk-server-listen :5000** at the end of start script

- Start VM on primary Host

Add **-blk-server 192.168.XXX.XXX:5000** at the end of start VM (run_ft) script