



本科生实验报告

学生姓名： 丁晓琪

学生学号： 22336057

专业名称： 计科

一：实验任务

实验题1：CRIU是一种在用户空间实现的进程或者容器checkpoint和restore的方法，从而实现进程或者容器的保存和恢复。请利用CRIU实现进程和容器的迁移（迁移种类不限），并测试迁移过程中的性能损耗（如进程停止时间、网络传输时间等）

二：实验过程

1. 进程迁移

- 利用 nfs 将虚拟机1的 /home 目录，挂载在虚拟机2的 /home 目录
 - 目的：在用criu生成进程迁移相关文件后，不需要从虚拟机1复制粘贴到虚拟机2，而是在虚拟机2远程访问运行虚拟机1中文件，从而复原进程
 - 注意：在网上教程中是将虚拟机1的 /home 目录挂载在了虚拟机2的 /nfs/home 目录，但是后面进程迁移时会发现找不到 /home 目录的报错，所有挂载在同一个地址

```
test2@test2-virtual-machine:/$ cd home
test2@test2-virtual-machine:/home$ cd test1
test2@test2-virtual-machine:/home/test1$ ls
checkpoint_file  nfs      test.sh
criu-3.14       test.log  Untitled 1.odt
examples.desktop test.out  公共的
```

- （虚拟机1）在后台启动一个新的会话进程运行 test.sh 脚本
 - test.sh 脚本如下：打印时间

```
1  #!/bin/bash
2  while (true)
3  do
4      time=$(date +%s.%N)
5      echo $time
6      sleep 10e-4
7  done
```

- 执行并且查看输出：|
 - setid: 让脚本在新的会话进行，脱离当前终端会话，不会收到终端关闭影响。

- `</dev/null`:脚本无法从用户处接收输入
- `&>$PWD/test.out`:脚本运行的输出被重定向到当前目录的 `test.out` 下面

```
test1@test1-virtual-machine:~$ sudo setsid ./test.sh </dev/null &>$PWD/test.out
^[[A^[[B^C^C^C^C^C已杀死
test1@test1-virtual-machine:~$ tail test.out
1728484212.127702858
1728484212.135880297
1728484212.145354654
1728484212.153787007
1728484212.165464634
1728484212.178420179
1728484212.186824142
1728484212.195421070
1728484212.204080824
1728484212.219608091
test1@test1-virtual-machine:~$
```

- 查看进程相关信息:

`ps -ef | grep test`:脚本进程pid为10539

```
test1 10376 3242 0 22:18 pts/21 00:00:00 bash
test1 10508 2427 0 22:21 ? 00:00:02 gedit /home/test1/test.out
root 10538 4226 0 22:22 pts/2 00:00:00 sudo setsid ./test.sh
root 10539 10538 14 22:22 ? 00:00:28 /bin/bash ./test.sh
test1 11536 3242 0 22:22 pts/22 00:00:00 bash
test1 52468 3344 1 22:25 ? 00:00:00 /usr/lib/firefox/firefox -contentproc -childI
test1 61173 11536 0 22:25 pts/22 00:00:00 ps -ef
```

- 虚拟机1设置检查点并且终端进程:为进程10539创建检查点并且将进程检查点文件保存在 `checkpoint_file` 目录

```
1 mkdir checkpoint_file
2 criu dump -D checkpoint_file -j -t 10539
```

- 在虚拟机2挂载虚拟1/home目录后,检查是否存在 `checkpoint_file` 文件,并且恢复进程 `criu restore -D checkpoint_file -j`:恢复 `checkpoint_file` 中的进程

- 问题1:有一些文件的大小和预期不符合

解决:用 `truncate -s size file` 缩小文件file到预期大小size

```
test2@test2-virtual-machine:/$ cd home
test2@test2-virtual-machine:/home$ cd test1
test2@test2-virtual-machine:/home/test1$ ls
checkpoint_file  nfs      test.sh      模板  文档  桌面
criu-3.14        test.log  Untitled 1.odt  视频  下载
examples.desktop test.out  公共的      图片  音乐
test2@test2-virtual-machine:/home/test1$ sudo criu restore -D checkpoint_file -j
10539: Error (criu/files-reg.c:1674): File home/test1/test.out has bad size 136
6575 (expect 820680)
10539: Error (criu/files.c:1191): Unable to open fd=1 id=0x9
Error (criu/cr-restore.c:2171): Restoring FAILED.
test2@test2-virtual-machine:/home/test1$ truncate -s 820680 test.out
test2@test2-virtual-machine:/home/test1$ sudo criu restore -D checkpoint_file -j
10539: Error (criu/files-reg.c:1674): File usr/lib/locale/locale-archive has bad
size 4795408 (expect 10219008)
```

- 问题2:两台虚拟机的链接库版本不一样报错

```
test1@test1-virtual-machine:~$ ls -l /lib/x86_64-linux-gnu/libc.so.6
lrwxrwxrwx 1 root root 12 10月  9 15:34 /lib/x86_64-linux-gnu/libc.so.6 -> libc-
2.23.so

test2@test2-virtual-machine:/home/test1$ ls -l /lib/x86_64-linux-gnu/libc.so.6
lrwxrwxrwx 1 root root 12 5月  3 2022 /lib/x86_64-linux-gnu/libc.so.6 -> libc-
2.27.so
```

解决:重新创建虚拟机3和虚拟机1的链接库版本一致

```
test3@test3-virtual-machine:~/桌面/vmware-tools-distrib$ sudo mount 192.168.91.1
r35:/home /home
```

- 在虚拟机3重新恢复进程:

```
test3@test3-virtual-machine:/home/test1
test3@test3-virtual-machine:/home$ cd test1
test3@test3-virtual-machine:/home/test1$ tail test.out
1728528421.066471845
1728528421.078010318
1728528421.088385295
1728528421.100438301
1728528421.112442730
1728528421.122917681
1728528421.134512414
1728528421.144364526
1728528421.155994509
1728528421.237779998
test3@test3-virtual-machine:/home/test1$ tail test.out
1728528430.478811410
1728528430.488919245
1728528430.499246307
1728528430.509991102
1728528430.520286337
1728528430.530942456
1728528430.539909890
1728528430.549489270
1728528430.560115867
1728528430.577099729
test3@test3-virtual-machine:/home/test1$

ration not permitted
Error (crui/cr-restore.c:1890): Restoring FAILED.
test3@test3-virtual-machine:/home/test1$ crui restore -D checkpoint_file -j
Error (crui/utll.c:666): exited, status=3
Error (crui/utll.c:666): exited, status=3
nodprobe: ERROR: could not insert 'lp_tables': Operation not permitted
iptables v1.6.0: can't initialize iptables table 'filter': Table does not exist
(do you need to insmod?)
Perhaps iptables or your kernel needs to be upgraded.
Warn (crui/sk-tcp.c:842): TCP_REPAIR isn't available to unprivileged users
Error (crui/cr-restore.c:922): 10539: Write 10538 to sys/kernel/ns_last_pid: Operation not permitted
Error (crui/cr-restore.c:1890): Restoring FAILED.
test3@test3-virtual-machine:/home/test1$ sudo crui restore -D checkpoint_file -j

Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
```

查看虚拟机3中恢复后的进程的pid, 也是10539

```
test3 5511 1436 0 10:41 ? 00:00:00 /usr/bin/speech-otspatcher
pawn --communication-method unix_socket --socket-path /run/user/1000/speech-otspatcher/speechd.sock
root 10539 6440 8 10:46 ? 00:00:11 /bin/bash ./test.sh
test3 14810 2197 0 10:48 pts/2 00:00:00 ps -ef
test3 14811 2197 0 10:48 pts/2 00:00:00 grep --color=auto test
test3@test3-virtual-machine:/home/test1$
```

2. 容器迁移:

- 在虚拟机1创建容器:

- 在国内的镜像网站拉取 busybox 的镜像:

```
docker pull swr.cn-north-4.myhuaweicloud.com/ddn-k8s/registry.k8s.io/e2e-test-images/busybox:1.29-4
```

- 查看镜像是否拉取成功:

```
docker images
```

```
test3@test3-virtual-machine:/home$ docker pull swr.cn-north-4.myhuaweicloud.com/ddn-k8s/registry.k8s.io/e2e-test-images/busybox:1.29-4
1.29-4: Pulling from ddn-k8s/registry.k8s.io/e2e-test-images/busybox
b4a6e23922dd: Pull complete
Digest: sha256:7dddb083e44b8f0e2f1fccac9562f5600b71e7717515ebd2131bdb94ad8034c
Status: Downloaded newer image for swr.cn-north-4.myhuaweicloud.com/ddn-k8s/registry.k8s.io/e2e-test-images/busybox:1.29-4
test3@test3-virtual-machine:/home$ docker ps
CONTAINER ID        IMAGE               COMMAND
test3@test3-virtual-machine$ docker run -d --name loop2 --security-opt seccomp:unconfined busybox /bin/sh -c 'i=0; while true; do echo $i; i=$((expr $i + 1)); sleep 1; done'
Unable to find image 'busybox:latest' locally
test3@test3-virtual-machine$ docker run -d --name loop2 --security-opt seccomp:unconfined busybox /bin/sh -c 'i=0; while true; do echo $i; i=$((expr $i + 1)); sleep 1; done'
Unable to find image 'busybox:latest' locally
test3@test3-virtual-machine$ docker images
REPOSITORY          TAG                IMAGE ID           CREATED            SIZE
swr.cn-north-4.myhuaweicloud.com/ddn-k8s/registry.k8s.io/e2e-test-images/busybox 1.29-4             d59c675982db      5 years ago       1.15MB
test3@test3-virtual-machine$
```

- 在docker容器中启动一个进程:

docker run -D: 从给定的镜像启动新的容器, 并且容器在后台运行

--name loop2: 容器的名称为loop2

swr.cn-north-4.myhuaweicloud.com/ddn-k8s/registry.k8s.io/e2e-test-images/busybox:1.29-4: 创建容器的镜像

/bin/sh -c 'i=0; while true; do echo \$i; i=\$((expr \$i + 1)); sleep 1; done': 容器启动后要执行的命令。用/bin/sh (一个 shell) 执行一个无限循环的脚本。脚本的内容是初始化一个变量 i 为 0, 进入一个无限循环, 每次循环都会打印当前的 i 值, 将 i 的值加 1, 然后休眠 1 秒。

```
test3@test3-virtual-machine$ docker run -d --name loop2 --security-opt seccomp:unconfined busybox /bin/sh -c 'i=0; while true; do echo $i; i=$((expr $i + 1)); sleep 1; done'
Unable to find image 'busybox:latest' locally
test3@test3-virtual-machine$ docker images
REPOSITORY          TAG                IMAGE ID           CREATED            SIZE
swr.cn-north-4.myhuaweicloud.com/ddn-k8s/registry.k8s.io/e2e-test-images/busybox 1.29-4             d59c675982db      9 years ago       1.15MB
swr.cn-north-4.myhuaweicloud.com/ddn-k8s/registry.k8s.io/e2e-test-images/busybox:1.29-4 /bin/sh -c 'i=0; while true; do echo $i; i=$((expr $i + 1)); sleep 1; done'
3889b4dabbb71a46022b4baaf7711ce49afcd1b5e48754ee0ee403e
test3@test3-virtual-machine$
```

- 确认容器状态: docker ps -a

检查容器中是否有计数进程: docker top loop2

```
test3@test3-virtual-machine$ sudo systemctl restart docker
test3@test3-virtual-machine$ docker unpause loop2
Error response from daemon: Container 3889b4dabbb71a46022b4baaf7711ce49afcd1b5e48754ee0ee403e is not paused
test3@test3-virtual-machine$ docker ps -a
CONTAINER ID        IMAGE               COMMAND
3889b4dabbb71a46022b4baaf7711ce49afcd1b5e48754ee0ee403e swr.cn-north-4.myhuaweicloud.com/ddn-k8s/registry.k8s.io/e2e-test-images/busybox:1.29-4 /bin/sh -c 'i=0; while true; do echo $i; i=$((expr $i + 1)); sleep 1; done'
test3@test3-virtual-machine$ docker top
"docker top" requires at least 1 argument(s).
See 'docker top --help'.

Usage: docker top CONTAINER [ps OPTIONS]

Display the running processes of a container.
test3@test3-virtual-machine$ docker top loop2
UID          PID         PPID        C         TIME          TTY          TIME          CMD
root          9545        0           0         11:40         ?            00:00:00     /bin/sh -c i=0; while true; do echo $i; i=$((expr $i + 1)); sleep 1; done
root          974         0           0         11:41         ?            00:00:00     sleep 1
test3@test3-virtual-machine$
```

- (虚拟机1) 设置 checkpoint:

docker checkpoint create: 创建新的检查点

`--checkpoint-dir=/home/container/`: 检查点保存目录

创建检查点的容器是 `looper2`

检查点名称为 `checkpoint2`

查看容器的输出: `docker logs looper2`

```
test1@test1-virtual-machine:~$ sudo docker checkpoint create --checkpoint-dir=/home/container/ looper2 checkpoint2
checkpoint2
test1@test1-virtual-machine:~$ sudo docker logs looper2
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
```

注意设置检查点后容器中断,最后输出为91

```
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
test1@test1-virtual-machine:~$ S
```

- (虚拟机3) 恢复容器:
 - 恢复前准备: 拉取busybox镜像和创建克隆容器 `looper-clone` (在克隆容器中恢复容器)

```
test3@test3-virtual-machine:~$ sudo usermod -aG docker test3
test3@test3-virtual-machine:~$ systemctl restart docker
test3@test3-virtual-machine:~$ chmod 666 /var/run/docker.sock
chmod: 更改 '/var/run/docker.sock' 的权限: 不允许的操作
test3@test3-virtual-machine:~$ sudo chmod 666 /var/run/docker.sock
test3@test3-virtual-machine:~$ docker pull swr.cn-north-4.myhuaweicloud.com/ddn-k8s/registry.k8s.io/e2e-test-images/busybox:1.29-4
1.29-4: Pulling from ddn-k8s/registry.k8s.io/e2e-test-images/busybox
04a6e23922dd: Pull complete
Digest: sha256:7ddd6b83e44b8f6e2f1fccace9562f5600b71e7717515ebd2131bdb94ad8634c
Status: Downloaded newer image for swr.cn-north-4.myhuaweicloud.com/ddn-k8s/registry.k8s.io/e2e-test-images/busybox:1.29-4
test3@test3-virtual-machine:~$
```

```
test3@test3-virtual-machine:~$ sudo docker create --name looper-clone --security-opt seccomp:unconfined swr.cn-north-4.myhuaweicloud.com/ddn-k8s/registry.k8s.io/e2e-test-images/busybox:1.29-4 /bin/sh -c 'l=0; while true; do echo $l; l=$((expr $l + 1)); sl
op 1; done'
f3e0e080f932a052e0a4d3594fbc9d07417b96c74db88a05a
test3@test3-virtual-machine:~$
```

- 在克隆容器中恢复容器:

`docker start`:

```
test3@test3-virtual-machine:~$ sudo docker start --checkpoint-dir=/home/container/3889b4edabb713a46622b46baaf1711ce49afc5ad1b5e540754e60ee4d03e/checkpoints/ --checkpoint=checkpoint2 looper-clone
```

- [illegible]

由于不熟悉进程迁移和容器迁移的过程，创建检查点后将时间大部分花在解决恢复进程和容器的报错以及一些环境问题上。在网络传输上，时间主要花费在挂载检查文件所在的目录上。成功执行恢复命令时也需要耗费一定时间等待进程/容器启动。