



实验课程: 分布式系统

实验名称: 第二次作业

专业名称: 计算机科学与技术

学生姓名: 吴臻

学生学号: 21307371

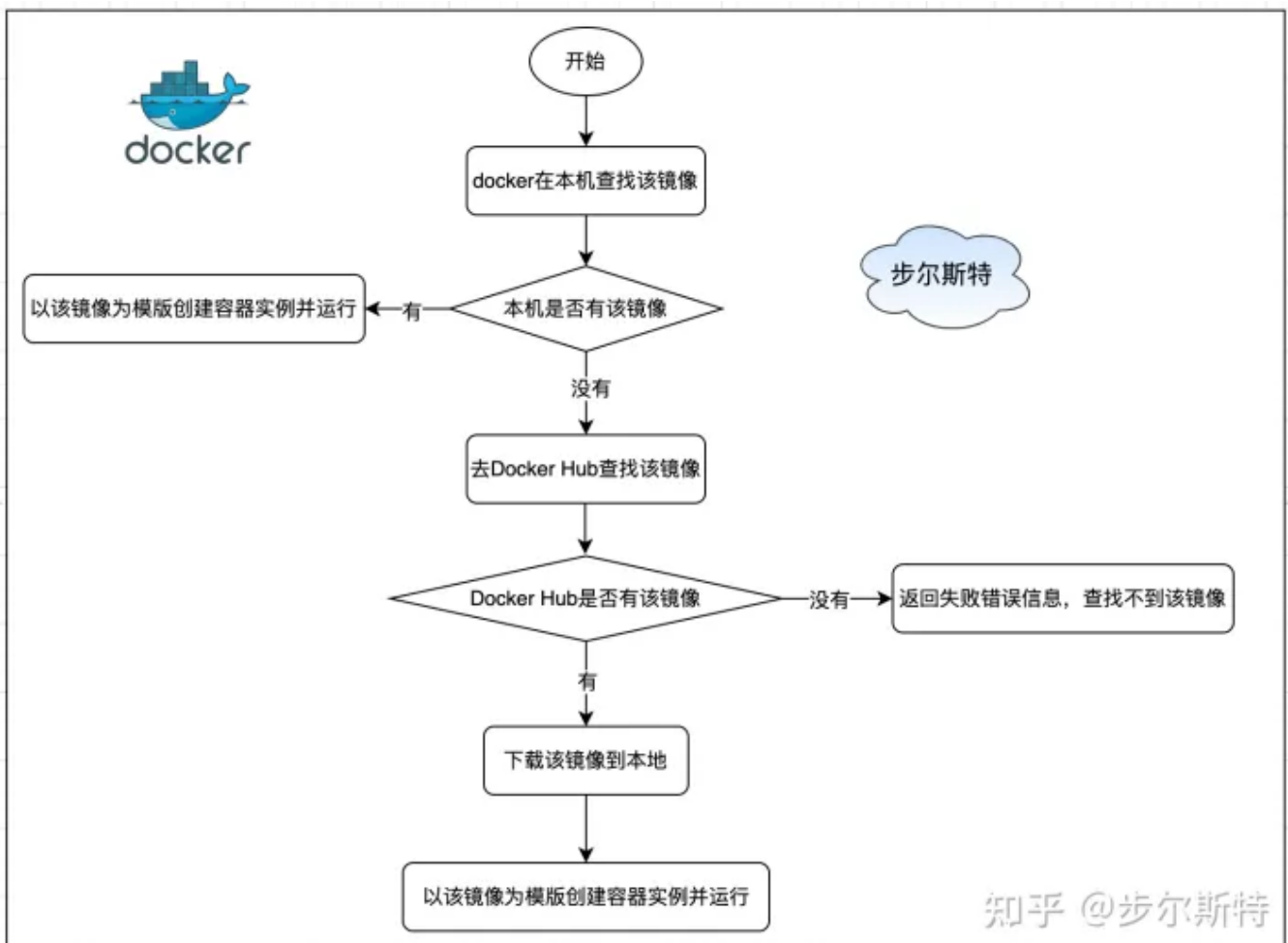
班级: 计科3班

- 问题描述
- ▼ 解决方案
 - 搭建环境
 - 进程热迁移

- 容器热迁移
- 实验结果
- 遇到的问题及解决方法
- 相关资料

问题描述

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，并利用样例程序如Web程序等测试迁移过程中的性能损耗（如延迟变化或者请求错误率等）、观察发现，并撰写报告



解决方案

搭建环境

两台虚拟机（运行在同一台电脑，文件可直接互传）：

主机：wz-ubuntu

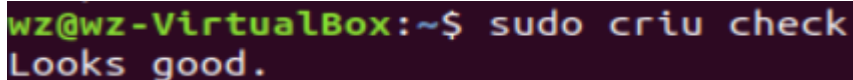
客户端：wz2-ubuntu

1. 首先将docker（和CRIU）所需要的支持软件全部安装好

```
#build dependent
sudo apt install build-essential -f -y
sudo apt install pkg-config -f -y
sudo apt install libnet-dev python-yaml libaio-dev -f -y
sudo apt install libprotobuf-dev libprotobuf-c-dev protobuf-c-compiler protobuf-compiler |
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common -y -f
```

2. criu安装 (3.14)

```
# criu install
curl -O -sSL http://download.openvz.org/criu/criu-3.14.tar.bz2
tar xjf criu-3.14.tar.bz2
cd criu-3.14
make
sudo cp ./criu/criu /usr/local/bin
# check
sudo criu check
```



```
wz@wz-VirtualBox:~$ sudo criu check
Looks good.
```

3. docker(24.0.2)

```
wz@wz-VirtualBox:~$ docker version
Client: Docker Engine - Community
 Version:           24.0.2
 API version:       1.43
 Go version:        go1.20.4
 Git commit:        cb74dfc
 Built:             Thu May 25 21:52:13 2023
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:           24.0.2
  API version:       1.43 (minimum version 1.12)
  Go version:        go1.20.4
  Git commit:        659604f
  Built:             Thu May 25 21:52:13 2023
  OS/Arch:           linux/amd64
  Experimental:      true
 containerd:
  Version:           1.6.21
  GitCommit:         3dce8eb055cbb6872793272b4f20ed16117344f8
 runc:
  Version:           1.1.7
  GitCommit:         v1.1.7-0-g860f061
 docker-init:
  Version:           0.19.0
  GitCommit:         de40ad0
```

进程热迁移

思路：首先要在服务器上创建要被迁移的进程 `test.sh`（从0开始递增输出整数），在服务器上暂停进程，将进程相关的文件打包到客户端，之后在客户端上使用CRIU恢复进程，观察恢复的进程是否会在之前状态下继续输出

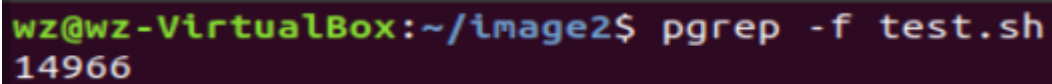
- 在服务器上

1. `test.sh`

```
#!/bin/bash
i=0;
while true;
do echo $i;
i=$(expr $i + 1);
sleep 1;
done
```

2. 运行该程序，并得到进程编号 (pid)

```
$ bash test.sh
$ pgrep -f test.sh
```



```
wz@wz-VirtualBox:~/image2$ pgrep -f test.sh
14966
```

3. 使用CRIU的dump执行暂停进程(\$id=14966)

```
criu dump --tree $id --images-dir $PWD --shell -job
#等价于
criu dump --tree 14966 --images-dir $PWD --shell -job
```

```
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
已杀死
wz@wz-VirtualBox:~/image2$
```

4. 将跟该进程有关的文件打包移植到客户端上

- 客户端

5. 在客户端恢复进程

```
$ criu restore -t 14966 --images-dir $PWD --shell -job
```

结果展示

客户端

```
wz@wz2:~/image2$ sudo criu restore -t 14966 --i
[sudo] wz 的密码:
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
```

可以看到，当客户端恢复进程后，进程的输出是接着服务器暂停前的输出继续的，说明进程热迁移成功

容器热迁移

思路：基本的思路是在服务器上创建容器，然后暂停容器，查看运行的输出。在客户端恢复容器之后查看输出，如果运行的输出是紧接服务器输出即说明热迁移成功。

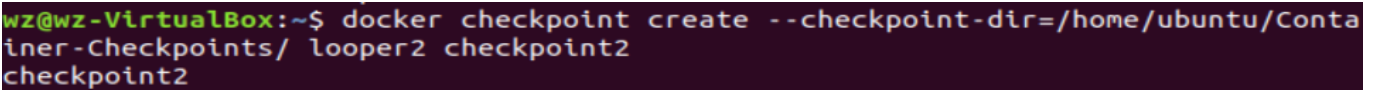
- 在服务器上

```
$ docker run -d --name looper2 --security-opt seccomp:unconfined busybox \  
    /bin/sh -c 'i=0; while true; do echo $i; i=$(expr $i + 1); sleep 1; done'  
  
# wait a few seconds to give the container an opportunity to print a few lines, then  
$ docker checkpoint create --checkpoint-dir= \  
/home/ubuntu/Container-Checkpoints/ looper2 checkpoint2  
  
# check your container & print log file  
$ docker ps  
$ docker logs looper2
```

代码解释

创建了一个名为"looper2"的docker容器，该容器在后台运行，并且周期性地打印数字。然后，创建了一个名为"checkpoint2"的检查点，以便稍后可以恢复容器的状态。最后，可以使用docker ps来检查容器状态，并使用docker logs来查看容器的日志输出。

结果展示

A terminal window with a dark background and light-colored text. The prompt is 'wz@wz-VirtualBox:~\$'. The command entered is 'docker checkpoint create --checkpoint-dir=/home/ubuntu/Container-Checkpoints/ looper2 checkpoint2'.

```
wz@wz-VirtualBox:~$ docker checkpoint create --checkpoint-dir=/home/ubuntu/Container-Checkpoints/ looper2 checkpoint2
```



```

42
43
44
45
46
47
48
49
50
51
52
53
54
55
wz@wz-VirtualBox:~$ docker logs looper2
0
1
2
3
4
5
6
7
8

```

```

wz@wz-VirtualBox:~$ docker start looper2
looper2
wz@wz-VirtualBox:~$ docker ps

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
49378b29b52f	busybox	"/bin/sh -c 'i=0; wh..."	54 minutes ago	Up 5 seconds

```

looper2

```

- 在客户端

```
$ docker create --name looper-clone --security-opt seccomp:unconfined busybox \
    /bin/sh -c 'i=0; while true; do echo $i; i=$((expr $i + 1)); sleep 1; done'
```

```
$ docker start --checkpoint-dir=/nfs/home/ubuntu/Container-Checkpoints/ --checkpoint=check
```

```
# check your container
```

```
$ docekr ps
```

```
$ docker logs looper-clone
```

代码解释

创建了一个名为 "looper-clone" 的 docker 容器，并使用之前创建的检查点 "checkpoint2" 恢复了容器的状态。最后，使用 docker ps 检查了容器的状态，并使用 docker logs 查看了容器的日志输出。

结果展示

```
wz@wz-VirtualBox:~$ docker start --checkpoint=checkpoint2 looper-clone
wz@wz-VirtualBox:~$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS   NAMES
2041946661c7   busybox    "/bin/sh -c 'i=0; wh..." 17 minutes ago Up 9 seconds           looper-clone
wz@wz-VirtualBox:~$ docker logs looper-clone
56
57
58
59
60
61
62
63
64
65
66
67
68
69
```

可以看到客户端运行的输出是紧接服务器输出，说明热迁移成功

实验结果

遇到的问题及解决方法

- 进程迁移后restore时权限不够

解决方法：更改文件权限: `sudo chmod -R 775 * /` `sudo chmod -R 775 image2`

- 容器迁移restore时 —— checkpoint-dir 不支持 (docker版本不同)

解决方法：手动把 checkpoint 文件目录加入到对应容器的默认位置

/var/lib/docker/containers/<CONTAINER_ID>/checkpoints/, 参考：

<https://github.com/moby/moby/issues/37344#issuecomment-450782189>

相关资料

1. https://zhuanlan.zhihu.com/p/269552404?utm_id=0

2. <https://www.jianshu.com/p/ed88fe8a446d>

进程迁移:

<https://blog.8086k.cn/archives/37/>