

分布式系统作业

第 2 次作业

姓名：郝裕玮

班级：计科 1 班

学号：18329015

1、什么是三层客户-服务器体系结构？

答：二层结构：客户端直接访问服务器为两层结构；

三层结构：在客户端与数据库之间加入了一个中间层，也叫组件层。即应用程序将业务规则、数据访问、合法性校验等工作放到了中间层进行处理。通常情况下，客户端不直接与数据库进行交互。

三层结构一般把各个功能模块划分为表示层（UI）、业务逻辑层（BLL）和数据访问层（DAL）三层架构，各层之间采用接口相互访问。

三层结构的优点：

（1）三层结构适合群体开发，每人可以有不同的分工，协同工作使效率倍增；

（2）可以更好的支持分布式计算环境。逻辑层的应用程序可以有多个机器上运行，充分利用网络的计算功能；

（3）避免了表示层直接访问数据访问层，表示层只和业务逻辑层有联系，提高了数据安全性。

2、在点对点的网络中，并不是每个节点都能成为超级节点，满足超级对等节点的合理要求是什么？

答：（1）该节点必须高度可用，因为其他节点需要依赖该节点；

（2）该节点的性能必须足够高，高到可以处理来自各个节点的不同的大量请求。

3、通过生成进程来构建并发服务器与使用多线程服务器相比有优点也有缺点。给出部分优点和缺点。

答：优点：对每个进程的安全性都有保障；

缺点：（1）创建/撤销进程的代价较高；

（2）同时若进程间需要通信，则需要内核参与控制，没有线程间通信那么简单。

4、维护到客户的 TCP/IP 链接的服务器是状态相关的还是状态无关的？说明理由。

答：（1）若服务器的传输层没有在客户端上保存消息，则服务器状态无关；

（2）所以本地操作系统所跟踪的状态与消息和服务器无关，与其传输层有关。

实验题：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迁移。请利用 CRIU 实现进程和容器的迁移（冷迁移和热迁移），并利用样例程序如循环计数程序等测试迁移过程中的性能损耗、观察发现，并撰写报告。

一、配置 CRIU 和 Docker

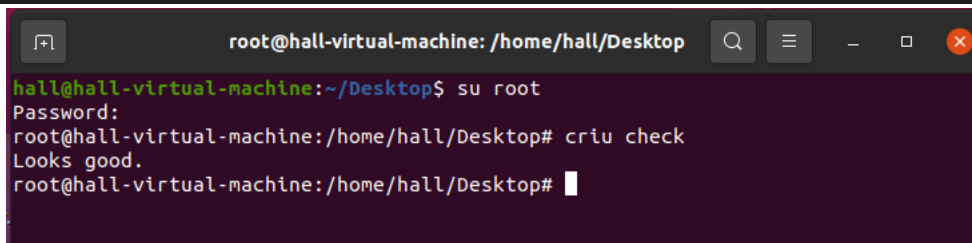
(1) 配置 CRIU

执行该行命令从 CRIU 官网下载 CRIU 的第三方软件包

```
sudo add-apt-repository ppa:criu/ppa
```

执行接下来 2 条命令进行安装，并进入 root 模式检验 CRIU 是否安装成功。

```
sudo apt-get update  
sudo apt install criu
```



A terminal window titled 'root@hall-virtual-machine: /home/hall/Desktop' showing the following commands and output:

```
hall@hall-virtual-machine:~/Desktop$ su root  
Password:  
root@hall-virtual-machine:/home/hall/Desktop# criu check  
Looks good.  
root@hall-virtual-machine:/home/hall/Desktop#
```

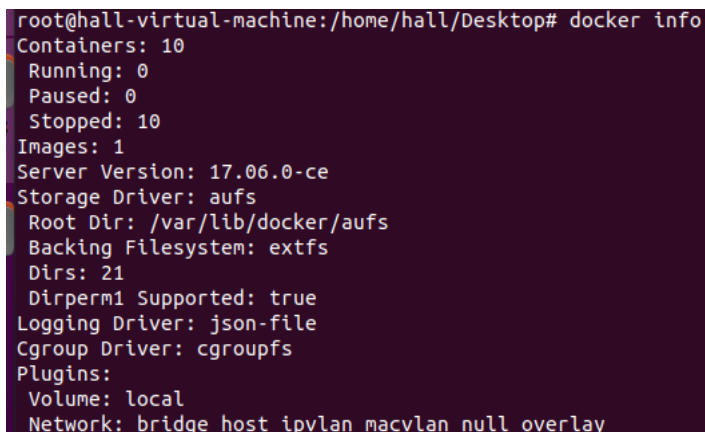
输出“Looks good.”，CRIU 配置成功。

(2) 配置 Docker (17.06 版本)

仍然进入 root 模式执行以下指令：

```
apt update  
apt install apt-transport-https ca-certificates curl software-properties  
-common  
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -  
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/u  
buntu xenial stable"  
apt install docker-ce=17.06.0~ce-0~ubuntu
```

执行 docker info 可查看是否安装成功：



A terminal window titled 'root@hall-virtual-machine: /home/hall/Desktop' showing the output of the 'docker info' command:

```
root@hall-virtual-machine:/home/hall/Desktop# docker info  
Containers: 10  
Running: 0  
Paused: 0  
Stopped: 10  
Images: 1  
Server Version: 17.06.0-ce  
Storage Driver: aufs  
Root Dir: /var/lib/docker/aufs  
Backing Filesystem: extfs  
Dirs: 21  
Dirperm1 Supported: true  
Logging Driver: json-file  
Cgroup Driver: cgroupfs  
Plugins:  
Volume: local  
Network: bridge host ipvlan macvlan null overlay
```

同时首先需要打开 Docker 实验性能：

```
sudo nano /etc/docker/daemon.json
```

在 daemon 中加入以下代码（将实验性能设置为 true）：

```
{  
  "experimental":true  
}
```

之后由于 Docker 服务器在国外，所以我们需要为 docker 设置国内阿里云的镜像加速器，否则在后面实验中无法正常拉取镜像。

在 daemon 中加入以下代码：

```
{  
  "registry-mirrors": ["https://alzgoonw.mirror.aliyuncs.com"]  
}
```

重启 Docker 即可：

```
systemctl restart docker
```

二、进程热迁移

首先编写脚本文件 loop，并将其放在 test 文件夹中：

输入该行命令创建脚本文件 loop：

```
touch loop.sh
```

脚本文件内容如下所示：

```
i=0;  
while true;  
do echo $i;  
i=$(expr $i + 1);  
sleep 2;  
done
```

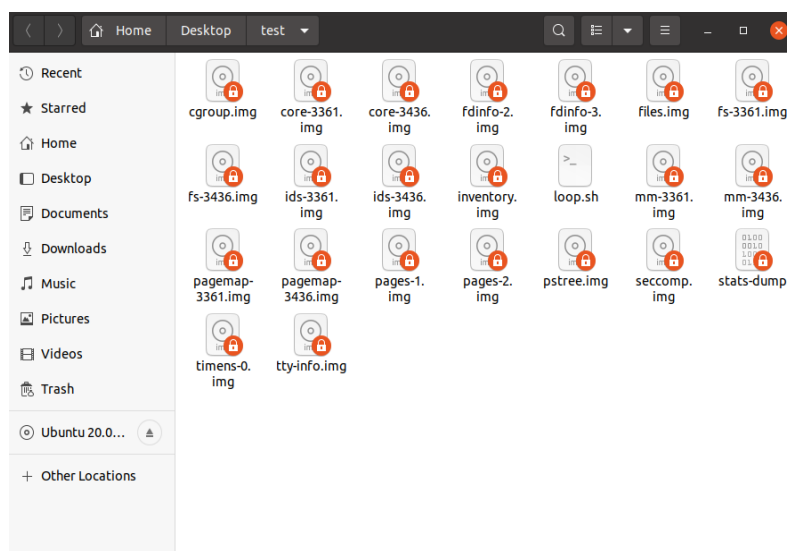
功能为每 2 秒输出一个从 0 开始递增 1 的数。

在 test 文件夹中同时打开 2 个终端窗口，分别执行图中的指令：

```
hall@hall-virtual-machine: ~/Desktop/test
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
hall@hall-virtual-machine:~/Desktop/test$ bash loop.sh

hall@hall-virtual-machine:~/Desktop/test$ pgrep -f loop.sh
3361
hall@hall-virtual-machine:~/Desktop/test$ cd ..
hall@hall-virtual-machine:~/Desktop$ sudo criu dump -t 3361 --images-dir test --shell-job
[sudo] password for hall:
Warn (compel/arch/x86/src/lib/infect.c:340): Will restore 3436 with interrupted system call
hall@hall-virtual-machine:~/Desktop$
```

```
hall@hall-virtual-machine: ~/Desktop/test
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
Killed
hall@hall-virtual-machine:~/Desktop/test$
```



发现进程在运行到 35 时被 Killed 且 test 文件夹中生成许多用于保存状态的 img 镜像文件。

执行以下指令对 test 文件夹进行压缩：

```
sudo tar -cvf test.tar.gz test
```

同时使用 U 盘将其移动到另一台虚拟机上并解压：

```
sudo tar -xvf test.tar.gz
```

执行 restore 命令对文件进行恢复：

```
sudo criu restore -t 3361 --images-dir test --shell-job
```

```
hall@hall-virtual-machine:~/Desktop$ sudo criu restore -t 3361 --images-dir test --shell-job
Warn (criu/crttools.c:249): Using -t with criu restore is obsoleted
36
37
38
39
40
█
```

发现进程继续从 36 恢复运行，所以进程热迁移成功。

三、容器热迁移

首先创建并运行一个容器 loop1，容器内部代码功能为每 2 秒输出一个从 0 开始递增 1 的数。

```
docker run -d --name loop1 --security-opt seccomp:unconfined busybox
/bin/sh -c 'i=0; while true; do echo $i; i=$((expr $i + 1)); sleep 2;
done'
```

```
root@hall-virtual-machine:/home/hall/Desktop# docker run -d --name loop1 --security-opt seccomp:unconfined busybox /bin/sh -c 'i=0; while true; do echo $i; i=$((expr $i + 1)); sleep 2; done'
219927ce51420603a9d0dd2209ef8641371aae86067737271b8eeec63589f118
```

并记录下其生成的 ID。

用 checkpoint 保存当前状态：

```
docker checkpoint create --checkpoint-dir=/home loop1 checkpoint1
```

```
root@hall-virtual-machine:/home/hall/Desktop# docker checkpoint create --checkpoint-dir=/home loop1 checkpoint1
checkpoint1
```

使用 docker log 指令查看程序中断前的运行状态（见下页）：

```
root@hall-virtual-machine:/home/hall/Desktop# docker logs loop1
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
root@hall-virtual-machine:/home/hall/Desktop#
```

可知运行到 19 终止。

将以容器 ID 为名的文件夹压缩，并用 U 盘存储。

```
cd /home
tar -cvf loop1.tar.gz 219927ce51420603a9d0dd2209ef8641371aae86067737271b8eeec63589f118
```

将其移动到另一台虚拟机上并解压，同时将文件夹移动到 home 文件夹下：

```
tar -xvf loop1.tar.gz
mv 219927ce51420603a9d0dd2209ef8641371aae86067737271b8eeec63589f118 /home
```

现在我们先在新虚拟机上创建一个新容器 loop4 但不运行它：

```
docker create --name loop4 --security-opt seccomp:unconfined busybox
/bin/sh -c 'i=0; while true; do echo $i; i=$(expr $i + 1); sleep 2;
done'
```

```
root@ubuntu:/home/hyw/Desktop# docker create --name loop4 --security-opt seccomp
:unconfined busybox /bin/sh -c 'i=0; while true; do echo $i; i=$(expr $i + 1); s
leep 2; done'
88a9d826b8362c43755c58b5ba0ee74e3b996ca1fd6bff4f752c08c1b0783130
```

之后让新容器 loop4 执行 checkpoint1，并执行 docker log 获取 loop4 的输出。

```
docker start --checkpoint-dir=/home/219927ce51420603a9d0dd2209ef8641371a
ae86067737271b8eeec63589f118/checkpoints/ --checkpoint=checkpoint1 loop4
```

```
docker logs loop4
```



```
root@ubuntu:/home/hyw/Desktop# docker start --checkpoint-dir=/home/219927ce51420603a9d0dd2209ef8641371aae86067737271b8eeec63589f118/checkpoints/ --checkpoint=checkpoint1 loop4
root@ubuntu:/home/hyw/Desktop# docker logs loop4
20
21
22
23
```

我们可发现在新容器 loop4 上，它会接着之前的容器 loop1 的输出中断点 19 继续运行，所以容器热迁移成功。

四、遇到的问题及解决方法

(1) 一开始在配置 CRIU 过程中照着老师 PDF 的参考网站无法完成，总是卡在 make 步骤（无法 make），最后自己通过查阅资料找到了最简单的安装方法并完成了 CRIU 的安装；

ubuntu安装criu

原创 原来是木斯 2021-03-20 17:55:59 160 收藏 2

版权

[criu官网](#)

添加criu的第三方软件包

```
sudo add-apt-repository ppa:criu/ppa
sudo apt-get update
```

然后

```
sudo apt install criu
```

最后检查sudo criu check

如果输出 Looks Good! 即安装成功

(2) 在创建 Docker 时报错，说无法拉取镜像，查阅资料后发现需要更改 Docker 的服务器至国内；

docker 安装完成后测试hello-world出现问题 (Unable to find image 'hello-world:latest' locally)

原创 来自丹麦的天籁 2019-04-03 10:56:51 53213 收藏 48 版权

分类专栏: docker 文章标签: docker



docker 专栏收录该内容

0 订阅

5 篇文章

订阅专栏

1 | 安装docker之后, 测试hello-world镜像, 终端卡在Unable to find image 'hello-world:latest' locally位置

```
weihe@ubuntu:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
```

```
1 | docker在本地没有找到hello-world镜像, 也没有从docker仓库中拉取镜像, 出项这个问题的原因: 是应为docker服务器再国外, 我们在国内
2 | 无法正常拉取镜像, 所以就需要我们为docker设置国内阿里云的镜像加速器;
3 | 需要修改配置文件 /etc/docker/daemon.json 如下
4 |
5 | {
6 |   "registry-mirrors": ["https://alzgoonw.mirror.aliyuncs.com"]
7 | }
```

(3) 忘记在编辑 daemon 文件后重启 docker;

(4) 创建容器时脚本代码出现错误, 导致输出不对:

```
'i=0; while true; do echo $i; i=$(expr $i+1); sleep 2; done'
```

这样写是不对的, 实际上 i+1 中间应有空格, 否则无法顺序输出

0 1 2 3...

```
hall@hall-virtual-machine:~/Desktop$ bash t1.sh
0
0+1
0+1+1
0+1+1+1
0+1+1+1+1
0+1+1+1+1+1
0+1+1+1+1+1+1
0+1+1+1+1+1+1+1
```

```
'i=0; while true; do echo $i; i=$(expr $i + 1); sleep 2; done'
```

```
hall@hall-virtual-machine:~/Desktop$ bash t1.sh
0
1
2
3
4
5
```