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checkpoi

nt和restore的方法，从而实现进程或者容器的迁移。请利用C

RIU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

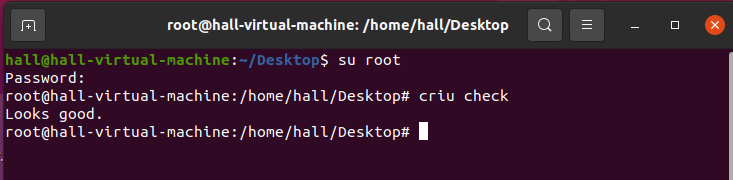
执行该行命令从CRIU官网下载CRIU的第三方软件包

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



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

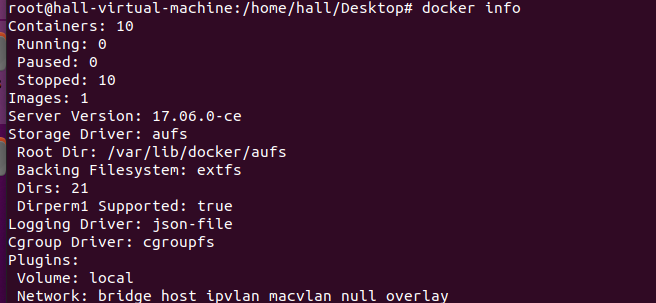
apt install apt-transport-https ca-certificates curl software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -

add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu 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：



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;

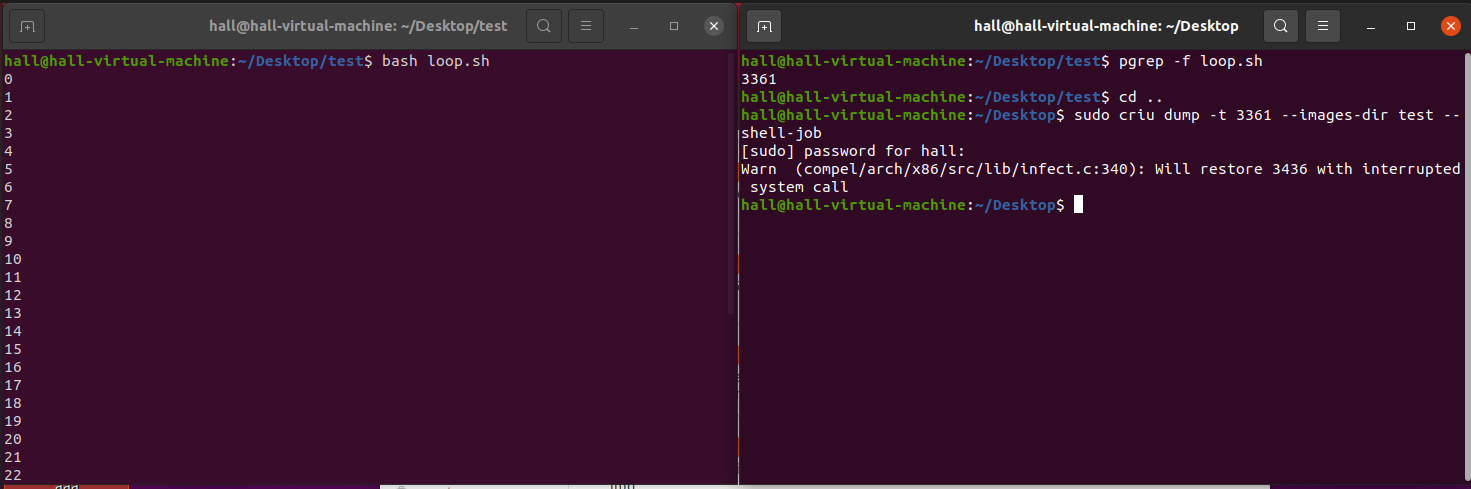
i=$(expr $i + 1);

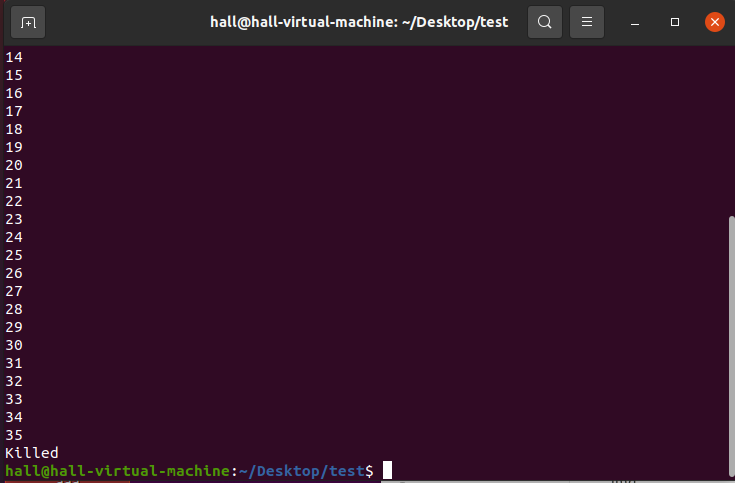
sleep 2;

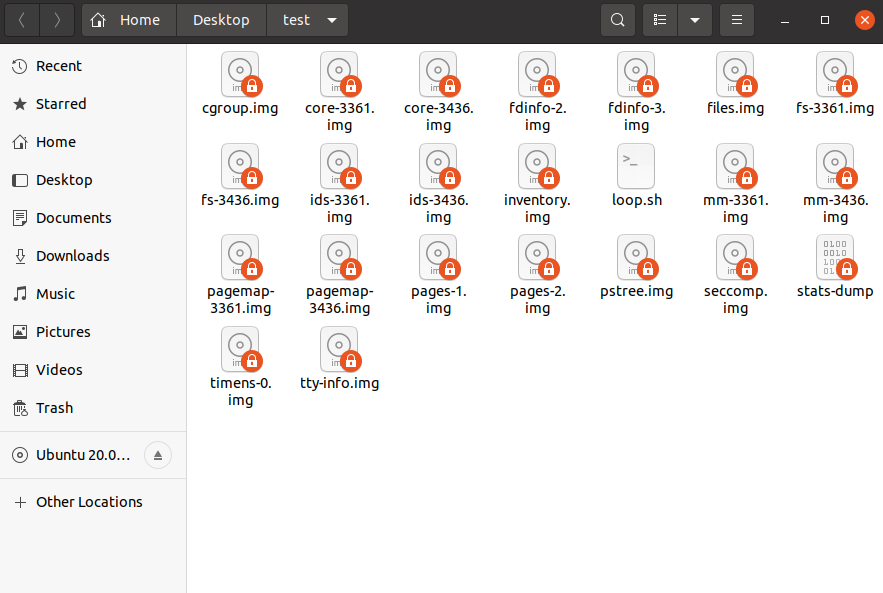
done

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：







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：

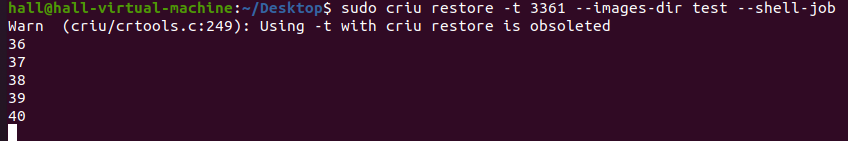
sudo tar -cvf test.tar.gz test

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



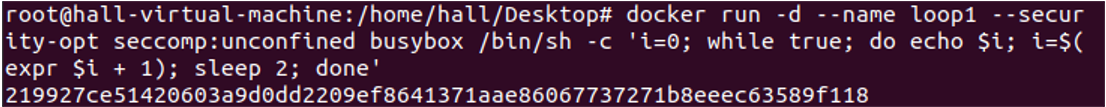
发现进程继续从36恢复运行，所以进程热迁移成功。

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

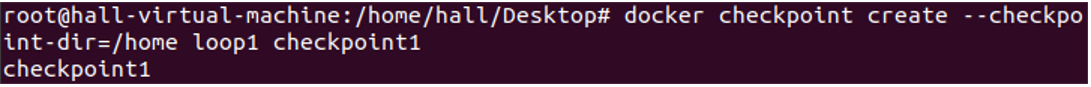
/bin/sh -c 'i=0; while true; do echo $i; i=$(expr $i + 1); sleep 2; done'



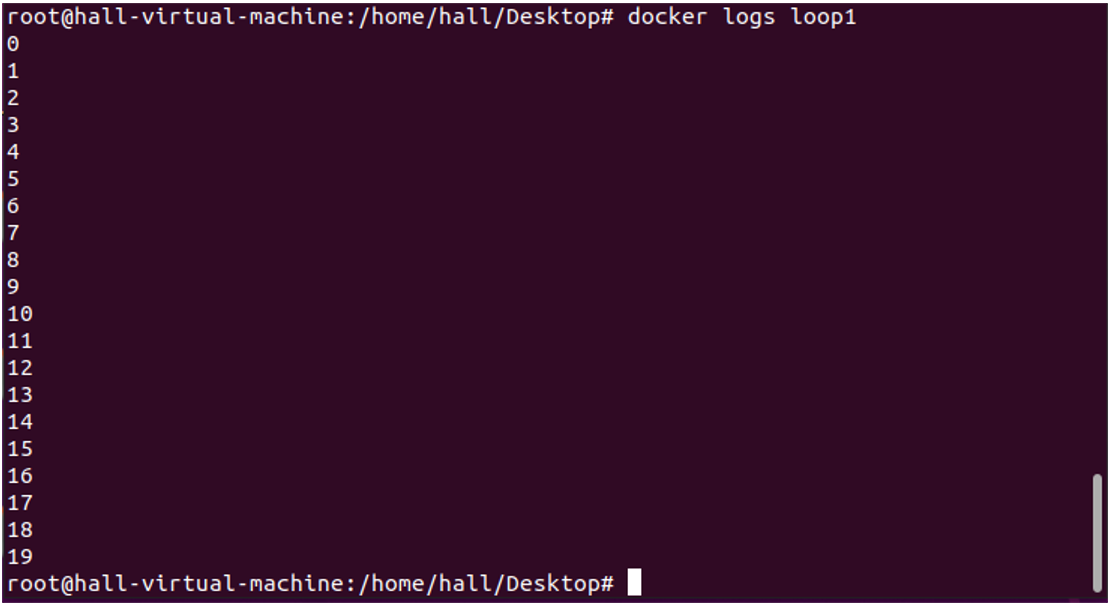
并记录下其生成的ID。

用checkpoint保存当前状态：

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



使用docker log指令查看程序中断前的运行状态（见下页）：



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：

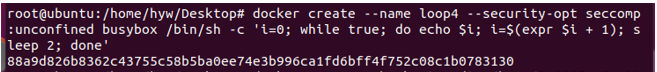
tar -xvf loop1.tar.gz

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'

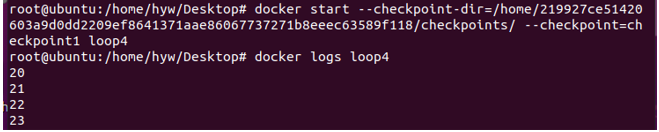


之后让新容器loop4执行checkpoint1，并执行docker log获取loop4的输出。

docker start --checkpoint-dir=/home/219927ce51420603a9d0dd2209ef8641371a

ae86067737271b8eeec63589f118/checkpoints/ --checkpoint=checkpoint1 loop4

docker logs loop4



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；



（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；

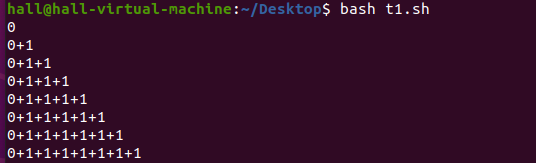


（3）忘记在编辑daemon文件后重启docker；

（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…



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

