在 QEMU 上跑 arm/mips 架构的 Debian

2020-11-12 | 2021-10-08 | 环境搭建

前言

对于分析 IOT 的安全工程师来说，模拟调试一个非本机架构的 binary/rootfs 是很常见的需求，今天介绍一种我本人觉得虽然稍微有点扭曲但是效果最好的办法：用 qemu-system 模拟运行异构的 Debian，并且相比常见的 network install 的方法有很大的改进。

本文主要介绍用 qemu-system-arm 来安装运行 arm Debian，对于 mips Debian 步骤基本是一样的。

For English version of this post, see here.

原理

以前的安装方法主要是通过 netboot 的 kernel 和 initrd 启动 installer，然后再通过 network install 安装系统，最后把新的 kernel 和 initrd 抽出来跑。总体上是可行的，但是问题是 QEMU 本身模拟异构的系统的时候性能就慢一些，再加上模拟网卡的性能较差，基本上没个几个小时是不可能装完的。考虑到这一点，一个更好的办法应该是把 offline installer 跑起来，但是目前还没有见到有类似的文章，经过简单的改进和一些探索，我成功的实现了 offline install，把安装时间基本缩短到 30 分钟左右。

准备

环境准备

我用的是 KUbuntu 20.04，理论上直接安装 qemu 即可。

```
apt install qemu-system qemu-utils bridge-utils libguestfs-tools
```

此外只需要下载 Debian 的 DVD 即可，以我用的 iso 为例。

```
wget -c "https://mirrors.ustc.edu.cn/debian-cd/10.6.0/armhf/iso-dvd/debian-1
```

制作系统盘

制作一个空白的系统盘镜像。

```
qemu-img create -f qcow2 debian.qcow2 8G
```

安装

抽取 kernel 和 initrd

qemu-system 跑 Linux 需要 kernel 和 initrd，这部分其实就是 grub 的工作，因此如果想把 installer 跑起来，我们直接看看 installer 的 grub 怎么写的。

```
mkdir mnt
sudo mount -o loop debian-10.6.0-armhf-DVD-1.iso ./mnt
cat mnt/boot/grub/grub.cfg
```

可以看到正常启动 installer 的选项。

```
menuentry 'Install' {
    set background_color=black
    linux    /install.ahf/vmlinuz  --- quiet
    initrd   /install.ahf/initrd.gz
}
```

于是从上述位置取出 vmlinuz 和 initrd。

```
cp mnt/install.ahf/vmlinuz ./
cp mnt/install.ahf/initrd.gz ./
```

到这里准备工作就完成了。

光盘挂载

如果熟悉 Debian Installer 的工作方式的话我们知道 installer 干的事情就是把 `/cdrom` 当做 apt 源然后 `apt install` 整个系统而已，因此接下来理论上我们只要把光盘交给 QEMU 然后带着 kernel 和 initrd 跑起来就行了，但是有一个坑点是 qemu-system-arm 的 `-cdrom` 选项是不工作的，我不太清楚原因，可能也是驱动的问题。但是这里可以通过另一种方法绕过这个坑点：把 cd 作为硬盘接入 QEMU，然后挂载到 `/cdrom`，实际上用 U 盘安装的时候 installer 会自动帮你做这件事，只不过我们等会儿需要手动做一下。

网络驱动

另外一个坑点是 qemu-system-arm 并不支持 PCI 网卡，从一些说法中我看到似乎是 arm Debian 不支持，总之如果要给 arm Debian 分配网卡的话需要用 `-netdev` 和 `-device` 来指定 `virtio-blk-device`，但是注意 `-nic` 的 `model` 选项不能指定。

此外如果要以 tap 模式分配主机网卡，还需要手动操作一下，以我的机器为例。

```
brctl addbr br0
brctl addif br0 ens33 # ens33 是物理网卡对应的接口
brctl addif br0 vmnic0
ip tuntap add dev vmnic0 mode tap
ip link set vmnic0 up
ip link set br0 up
ip addr add dev br0 192.168.4.158/24 # 和 ens33 同一网段
systemd-resolve --interface=br0 --set-dns=192.168.4.1 # 添加 DNS
ip ro del default
ip ro add default via 192.168.4.1 dev br0
echo 1 > /proc/sys/net/ipv4/ip_forward
```

安装系统

最后完整的 qemu-system 命令如下，注意如果用了 tap 网卡需要以 root 身份执行。

```
sudo qemu-system-arm -machine virt -cpu cortex-a15 -smp cpus=4,maxcpus=4 -no
        -kernel ./vmlinuz -initrd ./initrd.gz -m 1024 \
        -netdev user,id=n0 -device virtio-net-device,netdev=n0 \
        -drive file=debian.qcow2,if=none,format=qcow2,id=hd0 -device virtio-
        -drive file=debian-10.6.0-armhf-DVD-1.iso,if=none,format=raw,id=hd
```

这里注意我们把 `debian-10.6.0-armhf-DVD-1.iso` 是作为 hdd 挂载的，因此首先就会遇到找不到 cdrom 的报错。

这里连续选择 `No`，然后会回到主菜单。

这里选择 `Execute a shell` 就可以进入 busybox 了。当然本来是可以用 `Alt + F2` 来切换 Shell 的，但是在 `nographics` 下我试了下没法发送 `Alt` 所以只能这样了。

然后直接利用 `dmesg` 查看硬盘挂载的情况，以我的为例。

```
[    9.713510] virtio_blk virtio0: [vda] 9117960 512-byte logical blocks (4.
[    9.741109]  vda: vda1 vda2
[    9.755204] virtio_blk virtio1: [vdb] 16777216 512-byte logical blocks (8
[    9.764864]  vdb: vdb1 vdb2 vdb3 < vdb5 >
```

这里显然可以看出来 `/dev/vda` 就是 iso 了，因此 `/dev/vda1` 就是安装分区，我们只要把它挂载到 `/cdrom` 就行了。

```
modprobe isofs
mount /dev/vda1 /cdrom
```

接下来返回主菜单选择 `Detect and mount CD-ROM` 即可继续安装，剩下流程和正常 Debian 的安装基本完全一致。

但是最后安装 Grub 的时候会提示错误导致无法安装 grub。

不过这也无所谓，因为 grub 的工作会由 qemu-system 来完成。

启动

抽取新 kernel, initrd

接下来我们要启动 Debian 了，因此之前的 kernel 和 initrd 已经不能使用了，所以我们需要从既有的磁盘镜像里抽取。

这里用到的工具是 `virt-ls` 和 `virt-copy-out`，注意需要 root 权限。

```
sudo virt-ls -l debian.qcow2 /boot
```

可以看到 vmlinuz 和 initrd 的符号链接。

```
drwxr-xr-x  3 0 0     1024 Nov 12 11:55 .
drwxr-xr-x 18 0 0     4096 Nov 12 11:19 ..
-rw-r--r--  1 0 0  3212152 Sep 17 21:42 System.map-4.19.0-11-armmp-lpae
-rw-r--r--  1 0 0  3212349 Oct 18 08:43 System.map-4.19.0-12-armmp-lpae
-rw-r--r--  1 0 0   210638 Sep 17 21:42 config-4.19.0-11-armmp-lpae
-rw-r--r--  1 0 0   210638 Oct 18 08:43 config-4.19.0-12-armmp-lpae
lrwxrwxrwx  1 0 0       31 Nov 12 11:51 initrd.img -> initrd.img-4.19.0-12-
-rw-r--r--  1 0 0 20587326 Nov 12 11:33 initrd.img-4.19.0-11-armmp-lpae
-rw-r--r--  1 0 0 20590187 Nov 12 11:55 initrd.img-4.19.0-12-armmp-lpae
lrwxrwxrwx  1 0 0       31 Nov 12 11:24 initrd.img.old -> initrd.img-4.19.0
drwx------  2 0 0    12288 Nov 12 11:02 lost+found
lrwxrwxrwx  1 0 0       28 Nov 12 11:51 vmlinuz -> vmlinuz-4.19.0-12-armmp-
-rw-r--r--  1 0 0  4403712 Sep 17 21:42 vmlinuz-4.19.0-11-armmp-lpae
-rw-r--r--  1 0 0  4403712 Oct 18 08:43 vmlinuz-4.19.0-12-armmp-lpae
lrwxrwxrwx  1 0 0       28 Nov 12 11:24 vmlinuz.old -> vmlinuz-4.19.0-11-ar
```

然后把 vmlinuz 和 initrd 拷贝出来。

```
sudo virt-copy-out -a debian.qcow2 /boot/initrd.img-4.19.0-12-armmp-lpae ./
sudo virt-copy-out -a debian.qcow2 /boot/vmlinuz-4.19.0-12-armmp-lpae ./
```

启动

最后就是用新的 kernel 和 initrd 启动系统了。

```
sudo qemu-system-arm -machine virt -cpu cortex-a15 -smp cpus=4,maxcpus=4 -no
        -kernel ./vmlinuz-4.19.0-12-armmp-lpae -initrd ./initrd.img-4.19.0-1
        -netdev tap,ifname=vmnic1,id=n0,script=no,downscript=no -device virt
        -drive file=debian.qcow2,if=none,format=qcow2,id=hd0 -device virtio-
```

到这里就完美搞定了。

总结

当然除了 qemu-system 的方法以外还可以选择 qemu-usermode，但是 qemu-usermode 有很多问题，比如不支持多线程调试，相对来说 qemu-system 要灵活的多，可以自定义硬件，甚至可以直接在模拟的 Debian 中调试。另外利用本文中的安装方法，在断网的情况下基本 30-60 min 就可以安装完毕，比以往的 network install 快很多。

一些小细节：

- 对于链接的 libc 不是 glibc 的程序，最新的 openwrt 19 使用的 musl，旧一点的 openwrt 15 用的 uclibc，基本够用。
- arm64 的 Debian/Ubuntu 不能直接运行部分 arm binary。
- 实机也是一个选择，比如树莓派是 arm 架构，很多路由器是 mips 架构。

参考

本文参考了非常多的文章，这里列出几篇主要的。

- 构建嵌入式 qemu.md
- 在 X86 Linux 下透過 Qemu 安裝 ARM 的 Debian 系統
- Installing Debian on QEMU’s 32-bit ARM “virt” board

相关文章

- C# 逆向入门：Celeste 拆包

逆向 # iot # 模拟 # 虚拟 # qemu # qemu system # qemu-system # arm # mips # debian # emulation

‹ 用麒麟框架深入分析实模式二进制文件 把闭包变成函数指针 ——libffi 闭包原理解析 ›