Linux Wind

Talk is cheap, Lead by example!

kvm 学习笔记–KVM架构与原理

1. KVM架构

kvm基本结构有2个部分构成：

- kvm 驱动，现在已经是linux kernel的一个模块了。其主要负责虚拟机的创建，虚拟内存的分配，VCPU寄存器的读写以及VCPU的运行。
- 另个组成是Qemu，用于模拟虚拟机的用户空间组件，提供I/O设备模型，访问外设的途径。

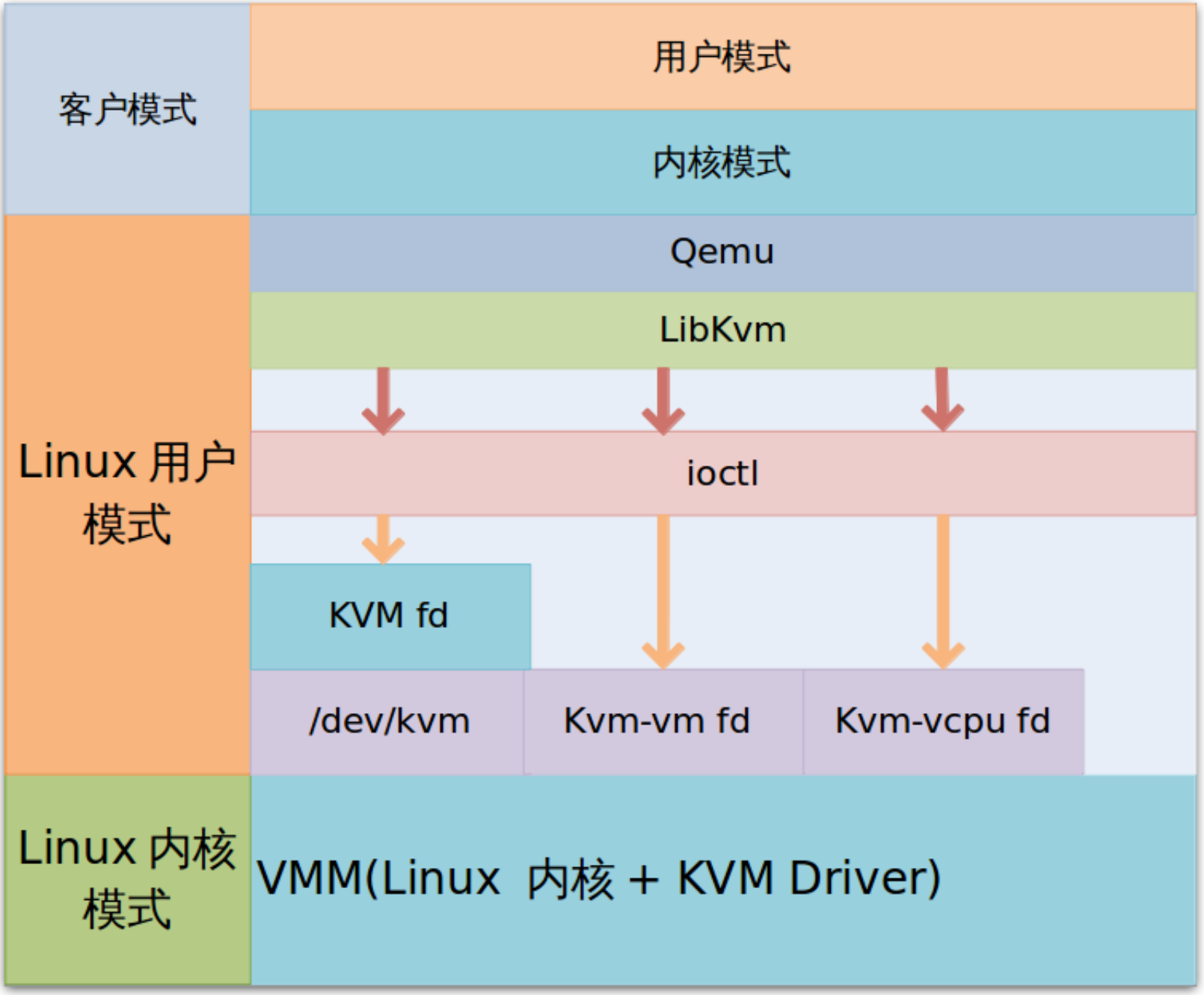
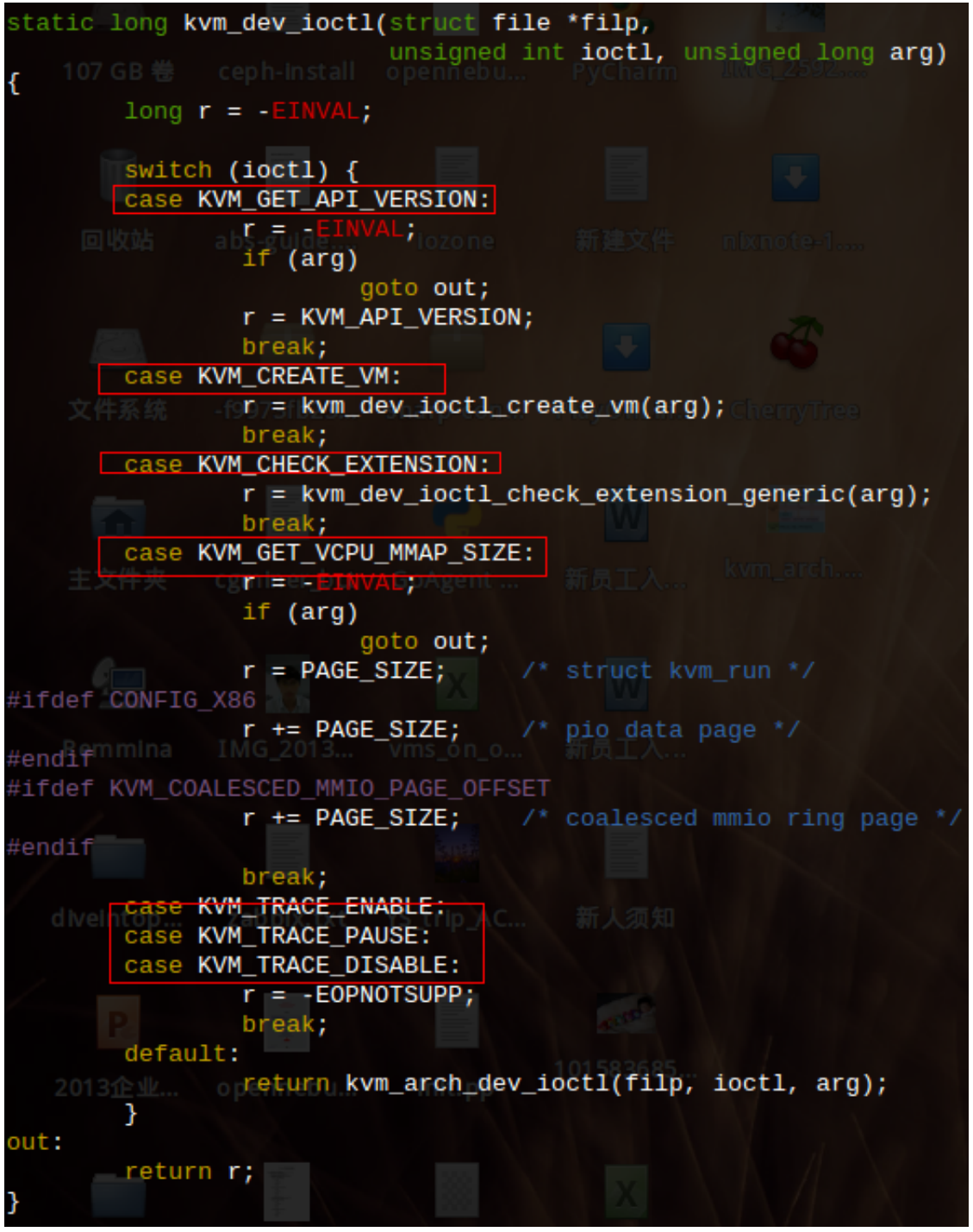

客户模式
用户模式
内核模式
Linux 用户模式
Qemu
LibKvm
ioctl
KVM fd
/dev/kvm
Kvm-vm fd
Kvm-vcpu fd
Linux 内核模式
VMM(Linux 内核 + KVM Driver)

图1 kvm基本结构

kvm基本结构如上图。kvm已经是内核模块，被看作是一个标准的linux 字符集设备（/dev/kvm）。Qemu通过libkvm应用程序接口，用fd通过ioctl向设备驱动来发送创建，运行虚拟机命令。设备驱动kvm就会来解析命令（kvm_dev_ioctl函数在kvm_main.c文件中）,如下图：



```
static long kvm_dev_ioctl(struct file *filp,
                          unsigned int ioctl, unsigned long arg)
{
        long r = -EINVAL;

        switch (ioctl) {
        case KVM_GET_API_VERSION:
                r = -EINVAL;
                if (arg)
                        goto out;
                r = KVM_API_VERSION;
                break;
        case KVM_CREATE_VM:
                r = kvm_dev_ioctl_create_vm(arg);
                break;
        case KVM_CHECK_EXTENSION:
                r = kvm_dev_ioctl_check_extension_generic(arg);
                break;
        case KVM_GET_VCPU_MMAP_SIZE:
                r = -EINVAL;
                if (arg)
                        goto out;
                r = PAGE_SIZE;     /* struct kvm_run */
#ifdef CONFIG_X86
                r += PAGE_SIZE;    /* pio data page */
#endif
#ifdef KVM_COALESCED_MMIO_PAGE_OFFSET
                r += PAGE_SIZE;    /* coalesced mmio ring page */
#endif
                break;
        case KVM_TRACE_ENABLE:
        case KVM_TRACE_PAUSE:
        case KVM_TRACE_DISABLE:
                r = -EOPNOTSUPP;
                break;
        default:
                return kvm_arch_dev_ioctl(filp, ioctl, arg);
        }
out:
        return r;
}
```

图2 kvm_dev_ioctl函数

kvm 模块让Linux主机成为一个虚拟机监视器（VMM），并且在原有的Linux两种执行模式基础上，新增加了客户模式，客户模式拥有自己的内核模式和用户模式。在虚拟机运行时，三种模式的工作各为：

- 客户模式： 执行非I/O的客户代码，虚拟机运行在这个模式下。
- 用户模式：代表用户执行I/O指令，qemu运行在这个模式下。
- 内核模式：实现客户模式的切换，处理因为I/O或者其他指令引起的从客户模式退出（VM_EXIT）。kvm 模块工作在这个模式下。

在kvm的模型中，每一个Gust OS都是作为一个标准的linux进程，都可以使用linux进程管理命令管理。

这里假如qemu通过ioctl发出KVM_CREATE_VM 指令，创建了一个VM后，qemu需要需要发送一些命令给VM，如KVM_CREATE_VCPU。这些命令当然也是通过ioctl发送的，用户程序中用ioctl发送KVM_CREATE_VM得到的返回值就是新创建的VM对应的fd(kvm_vm)，fd是创建的指向特定虚拟机实例的文件描述符，之后利用这个fd发送命令给VM进行访问控制。kvm解析这些命令的函数是kvm_vm_ioctl。

2. KVM 工作原理

kvm基本工作原理概述：

用户模式的qemu利用libkvm通过ioctl进入内核模式，kvm模块未虚拟机创建虚拟内存，虚拟CPU后执行VMLAUCH指令进入客户模式。加载Guest OS并执行。如果Guest OS 发生外部中断或者影子页表缺页之类的情况，会暂停Guest OS的执行，退出客户模式出行异常处理，之后重新进入客户模式，执行客户代码。如果发生I/O事件或者信号队列中有信号到达，就会进入用户模式处理。(如下图)

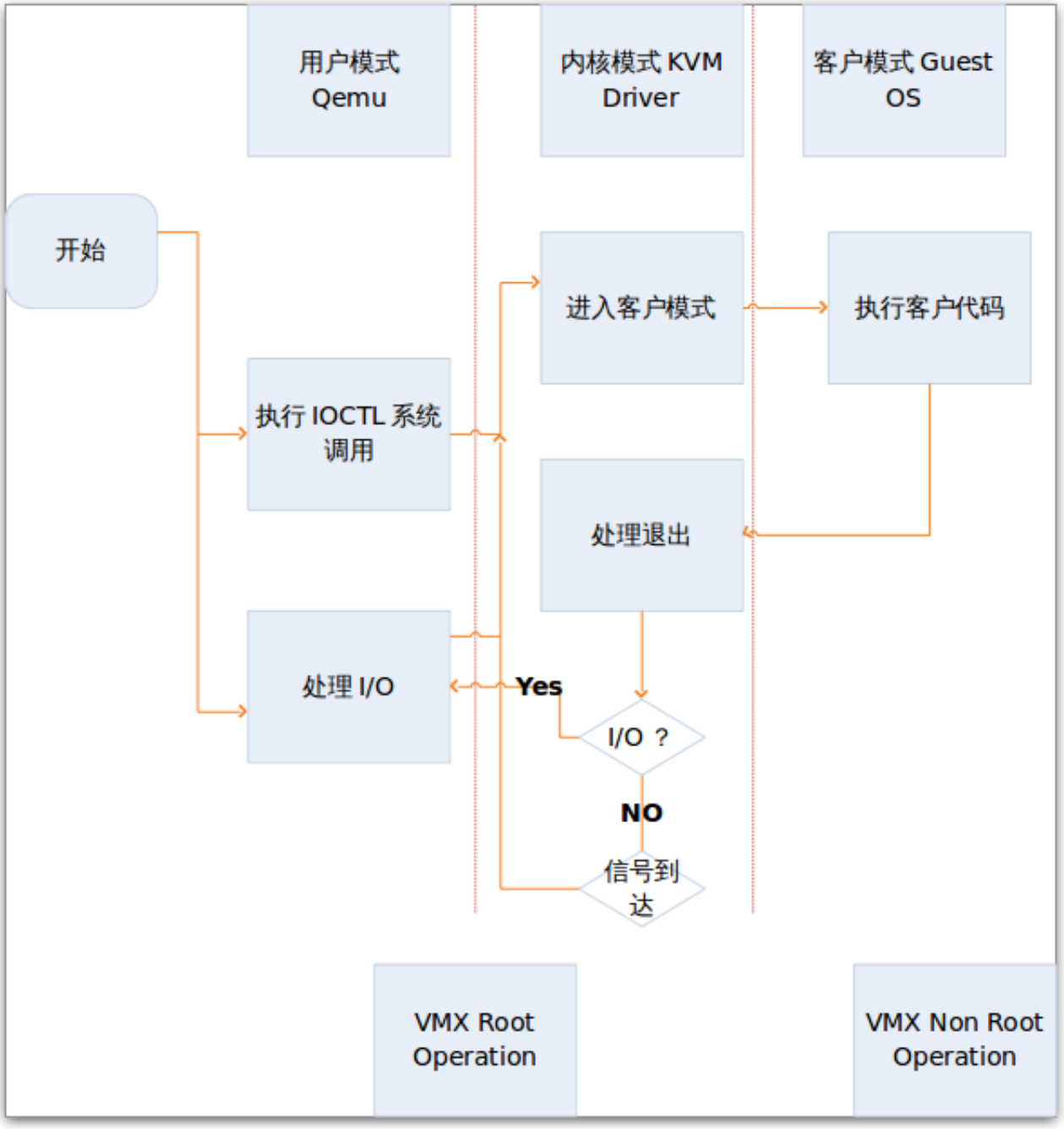

用户模式 Qemu
内核模式 KVM Driver
客户模式 Guest OS
开始
进入客户模式
执行客户代码
执行 IOCTL 系统调用
处理退出
处理 I/O
Yes
I/O ？
NO
信号到达
VMX Root Operation
VMX Non Root Operation

图3 KVM工作原理流程图

[参考资料]

http://zhangjun2915.blog.163.com/blog/static/38086236201041175424137/

kvm源代码分析

本条目发布于2013/07/09 [https://www.linuxwind.org/html/learning-the-kvm-01.html]。属于Cloud Computing分类，被贴了 kvm、kvm原理、kvm结构、qemu 标签。

《kvm 学习笔记-KVM架构与原理》上有2条评论

V_v
2013/07/0919:24

改用HTTPS访问，会不会影响博客访问速度？

疯狂小蜗牛 文章作者
2013/07/1010:26

会有一点点，不过可以忽略。