kernel hacker修炼之道之内核虚拟化KVM/QEMU——Guest OS, Qe...

2012-06-19 Liucw2012 阅 1304 转 7

内核虚拟化KVM/QEMU——Guest OS, Qemu, KVM工作流程

作者 李万鹏

这里主要介绍基于x86平台的Guest Os, Qemu, Kvm工作流程，如图，通过KVM APIs可以将qemu的command传递到kvm：

1.创建VM

vm_fd = open("/dev/kvm", xxx);

vm_fd = ioctl(system_fd, KVM_CREATE_VM, xxx);

2.创建VCPU

vcpu_fd = kvm_vm_ioctl(vm_fd, VM_CREATE_VCPU, xxx);

3.运行KVM

status = kvm_vcpu_ioctl(vcpu_fd, KVM_RUN, xxx);

Qemu通过KVM APIs进入KVM后，KVM会切入Guest OS，假如Guest OS运行运行，需要访问IO等，也就是说要访问physical device，那么Qemu与KVM就要进行emulate。 如果是KVM emulate的则由KVM emulate，然后切回Guest OS。如果是Qemu emulate的，则从KVM中进入Qemu，等Qemu中的device model执行完emulate之后，再次在Qemu中调用kvm_vcpu_ioctl(vcpu_fd, KVM_RUN, xxx)进入KVM运行，然后再切回Guest OS.

(图片勘误，如果KVM can emulate那么emulate之后应该层层返回到kvm_x86_ops->run(vcpu)，然后才切入guest os，不是直接切入，图画完了，不好修改)

Qemu是一个应用程序，所以入口函数当然是main函数，但是一些被type_init修饰的函数会在main函数之前运行。这里分析的代码是emulate x86 的一款i440板子。main函数中会调用在main函数中会调用kvm_init函数来创建一个VM(virtual machine)，然后调用机器硬件初始化相关的函数，对PCI，memory等进行emulate。然后调用qemu_thread_create创建线程，这个函数会调用

Liucw2012
关注 对话

TA的最新馆藏 (共307篇)
[转] bind源码解析(一)
[转] netfilter中对多连接协议跟踪和NA…
[转] pci 学习笔记
[转] ptmalloc,tcmalloc和jemalloc内…
[转] Linux内核虚拟文件系统
[转] [转载]完整的技术交易策略分析图

喜欢该文的人也喜欢 更多
刘嘉玲：脾气不好的人，请记住这…
中药贴敷儿科配方大全
人民日报发布:生活、工作27个沟通…
炸串也停业了师傅说技术我就不保…
法国艺术家James Tissot 宫廷人物…
不瞒你讲：缘分躲不掉，情人在一…
小宝宝“闹独立”，是他的第一逆…
验证一个男人有多爱你：不联系就…
《百战奇略》：一本通俗而又丰富…

VCPU(virtual CPU)，然后调用kvm_vcpu_ioctl，参数KVM_RUN，这样就进入KVM中了。进入KVM中第一个执行的函数名字相同，也叫kvm_vcpu_ioctl，最终会调用到kvm_x86_ops->run()进入到Guest OS，如果Guest OS要写某个端口，会产生一条IO instruction，这时会从Guest OS中退出，调用kvm_x86_ops->handle_exit函数，其实这个函数被赋值为vmx_handle_exit，最终会调用到kvm_vmx_exit_handlers[exit_reason](vcpu)，kvm_vmx_exit_handlers是一个函数指针，会根据产生事件的类型来匹配使用那个函数。这里因为是ioport访问产生的退出，所以选择handle_io函数。

```
5549static int (*kvm_vmx_exit_handlers[])(struct kvm_vcpu *vcpu) = {
5550     [EXIT_REASON_EXCEPTION_NMI]        = handle_exception,
5551     [EXIT_REASON_EXTERNAL_INTERRUPT]    = handle_external_interrupt,
5552     [EXIT_REASON_TRIPLE_FAULT]         = handle_triple_fault,
5553     [EXIT_REASON_NMI_WINDOW]           = handle_nmi_window,
5554     [EXIT_REASON_IO_INSTRUCTION]       = handle_io,
5555     [EXIT_REASON_CR_ACCESS]            = handle_cr,
5556     [EXIT_REASON_DR_ACCESS]            = handle_dr,
5557     [EXIT_REASON_CPUID]                = handle_cpuid,
5558     [EXIT_REASON_MSR_READ]             = handle_rdmsr,
5559     [EXIT_REASON_MSR_WRITE]            = handle_wrmsr,
5560     [EXIT_REASON_PENDING_INTERRUPT]    = handle_interrupt_window,
5561     [EXIT_REASON_HLT]                  = handle_halt,
5562     [EXIT_REASON_INVD]                 = handle_invd,
5563     [EXIT_REASON_INVLPG]               = handle_invlpg,
5564     [EXIT_REASON_VMCALL]               = handle_vmcall,
5565     [EXIT_REASON_VMCLEAR]              = handle_vmclear,
5566     [EXIT_REASON_VMLAUNCH]             = handle_vmlaunch,
5567     [EXIT_REASON_VMPTRLD]              = handle_vmptrld,
5568     [EXIT_REASON_VMPTRST]              = handle_vmptrst,
5569     [EXIT_REASON_VMREAD]               = handle_vmread,
5570     [EXIT_REASON_VMRESUME]             = handle_vmresume,
5571     [EXIT_REASON_VMWRITE]              = handle_vmwrite,
5572     [EXIT_REASON_VMOFF]                = handle_vmoff,
5573     [EXIT_REASON_VMON]                 = handle_vmon,
5574     [EXIT_REASON_TPR_BELOW_THRESHOLD]  = handle_tpr_below_threshold,
5575     [EXIT_REASON_APIC_ACCESS]          = handle_apic_access,
5576     [EXIT_REASON_WBINVD]               = handle_wbinvd,
5577     [EXIT_REASON_XSETBV]               = handle_xsetbv,
5578     [EXIT_REASON_TASK_SWITCH]          = handle_task_switch,
5579     [EXIT_REASON_MCE_DURING_VMENTRY]   = handle_machine_check,
5580     [EXIT_REASON_EPT_VIOLATION]        = handle_ept_violation,
5581     [EXIT_REASON_EPT_MISCONFIG]        = handle_ept_misconfig,
5582     [EXIT_REASON_PAUSE_INSTRUCTION]    = handle_pause,
5583     [EXIT_REASON_MWAIT_INSTRUCTION]    = handle_invalid_op,
5584     [EXIT_REASON_MONITOR_INSTRUCTION]  = handle_invalid_op,
5585};
```

如果KVM中的handle_io函数可以处理，那么处理完了再次切入Guest OS。如果是在Qemu中emulate，那么在KVM中的代码执行完后，会再次回到Qemu中，调用Qemu中的kvm_handle_io函数，如果可以处理，那么再次调用kvm_vcpu_ioctl，参数KVM_RUN，进入KVM，否则出错退出。

http://www.linuxso.com/linuxrumen/20534.html

类似文章

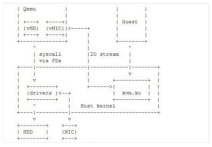
更多

与vmcs初始化有关

一个虚拟机由qemu来加载，大致上qemu的执行流程如 下： 1. KVM ioctl KVM_SET_MEMORY_REGION （kvm_dev_ioctl_set_memory_region）， 设置guest内存。这些内存不能被swap out. 2. ioctl KVM_CREATE_VCP...

qemu 安装

qemu 安装 一、QEMU简介。QEMU发起ioctrl来调用KVM接口，KVM则利用硬件扩展直接将虚拟机代码运行于主机之上，一旦vCPU需要操作设备寄...



淘宝核心系统团队博客 | 硬件虚拟化技术浅析

2 KVM的内部实现概述2.1 KVM的抽象对象2.2 KVM的vcpu2.3 KVM的IO虚拟化 2.3.1 IO的虚拟化2.3.2 VirtIO.KVM同应用程序(Qemu)的交互接口为/...

一个人的逆袭，必须要抓住这三个时机

一个人的逆袭，必须要抓住这三个时机。她第一次演讲摸不到头绪，不会讲，又紧张，在勉强晋级之后，她研究了市面上几乎所有教人演讲的书...

Centos 6安装KVM

通过以下命令安装虚拟机 virt-install \ --name vm3 \ --os-variant=rhel6 \ --vcpus=1 \ --ram 2048 \ --network bridge=br0 \ --disk path=/vm/images/vm3.img,size=50 \ --cdrom /vm/iso/Ce...

KVM基础功能

在QEMU/KVM中，qemu提供对cpu的模拟，展现给客户机一定的cpu数目和cpu特性；CPU的过载简单的理解就是vcpus的总是大于物理服务器的cpu个数，当虚拟机不是cpu满载的时候，是不会对整体的性能造成影响的。...

KVM虚拟机的创建、管理与迁移

amd.ko[root@target ~]# modprobe -ls | grep kv...

使用libvirt管理kvm（virsh篇）

使用libvirt管理kvm（virsh篇） 使用libvirt管理kvm（virsh篇）(2013-08-28 13:49:07)转载▼。libvirt（包括virsh）使用xml文件对虚拟机进行配置，其中包括虚拟机名称、分配内存、vcpu等多种信息。我编...

libvirt/qemu特性之cpu hotplug