翱翔在Linux的天空

HumJb & HaHa

humjb_1983

博客访问：9501
博文数量：80
博客积分：0
博客等级：民兵
技术积分：685
用 户 组：普通用户
注册时间：2014-02-20 08:27

加关注 短消息
论坛 加好友

KVM基本原理及架构八-KVM内核模块重要流程分析

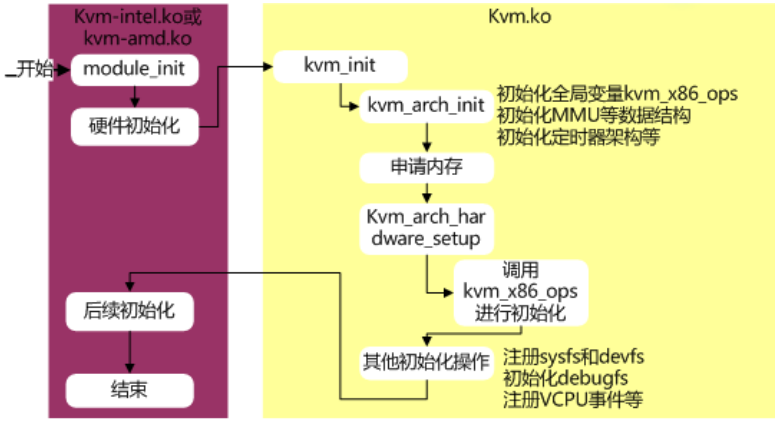
2014-07-18 20:01:08

分类：LINUX

KVM内核模块重要流程分析

8.1 初始化流程

如之前描述，KVM中主要包括：kvm.ko，kvm-intel.ko和kvm-amd.ko 3个模块，主要的初始化流程如下：



主要过程包括两部分：

ü 通过module_init宏执行平台相关的模块(kvm-intel.ko或kvm-amd.ko)初始化代码

ü 进入kvm_init执行kvm核心初始化工作

8.2 虚拟机创建

内核函数kvm_create_vm(linux/virt/kvm/kvm_main.c)用于创建虚拟机，用户态Qemu-kvm通过如下过程，最终进入内核的此函数中，由该函数完成虚拟机的创建。

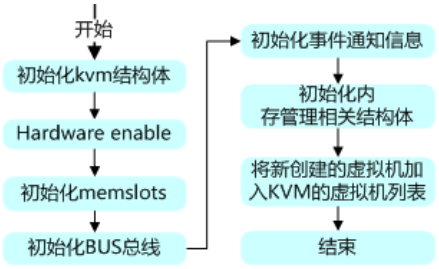
ioctl(KVM_CREATE_VM..)

à kvm_dev_ioctl

à kvm_dev_ioctl_create_vm

à kvm_create_vm

kvm_create_vm主要完成KVM虚拟机结构体的创建、KVM的MMU操作接口的安装、KVM的IO总线、事件通道的初始化等操作，主要流程如下：



1、 kvm_arch_create_vm()初始化kvm结构体

2、 hardware_enable_all()，针对每一个CPU，调用kvm_x86_ops中硬件相关的函数进行硬件使能，主要设置相关寄存器和标记，使CPU进入虚拟化相关模式中(如Intel VMX)。

3、 初始化memslots结构体信息

4、 初始化BUS总线结构体信息

5、 初始化事件通知信息和内存管理相关结构体信息

6、 将新创建的虚拟机加入KVM的虚拟机列表

8.3 VCPU创建

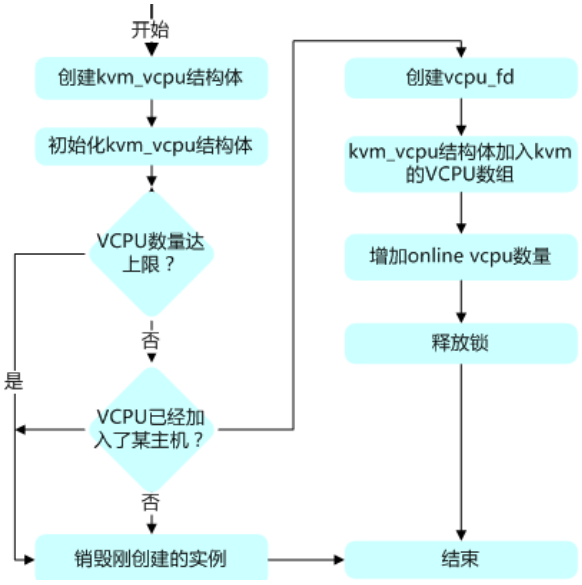
在通过ioctl(KVM_CREATE_VM…)创建虚拟机并获得相应的fd后，可以通过ioctl(KVM_CREATE_VCPU…)创建VCPU(相关操作通常在Qemu-kvm用户态进程中进行)。

ioctl(KVM_CREATE_VCPU..)

à kvm_vm_ioctl

à kvm_dev_ioctl_create_vcpu

VCPU的创建过程只要是创建VCPU描述符，即kvm_vcpu结构体，该结构体内容较多，包括硬件相关的内容。具体实现由内核函数kvm_dev_ioctl_create_vcpu()完成，主要流程如下：



1、 kvm_arch_vcpu_create()创建kvm_vcpu结构体，具体实现跟架构相关，直接调用kvm_x86_ops中的create_cpu方法执行，主要完成相关寄存器和CPUID的初始化，为调度运行做准备。

2、 kvm_arch_vcpu_setup()初始化kvm_vcpu结构体

3、 判断当前VCPU数量是否达到上限，如果是，则销毁刚创建的实例

4、 判断当前VCPU是否已经加入了某个KVM主机，如果是，则销毁刚创建的实例

5、 create_vcpu_fd()创建vcpu_fd

6、 将创建的kvm_vcpu结构体加入kvm的VCPU数组中

7、 增加online vcpu数量

8、 释放锁，结束。

8.4 VCPU运行

在VM和VCPU创建好并完成初始化后，就可以调度该VCPU运行了。VCPU(虚拟机)的运行主要任务是要进行上下文切换，主要就是相关寄存器、APIC状态、TLB等，通常上下文切换的过程如下：

1、 保存当前的上下文

2、 使用kvm_vcpu结构体中的上下文信息，加载到物理CPU中

3、 执行kvm_x86_ops中的run_vcpu函数，调用硬件相关的指令(如VMLAUNCH)，进入虚拟机运行环境中

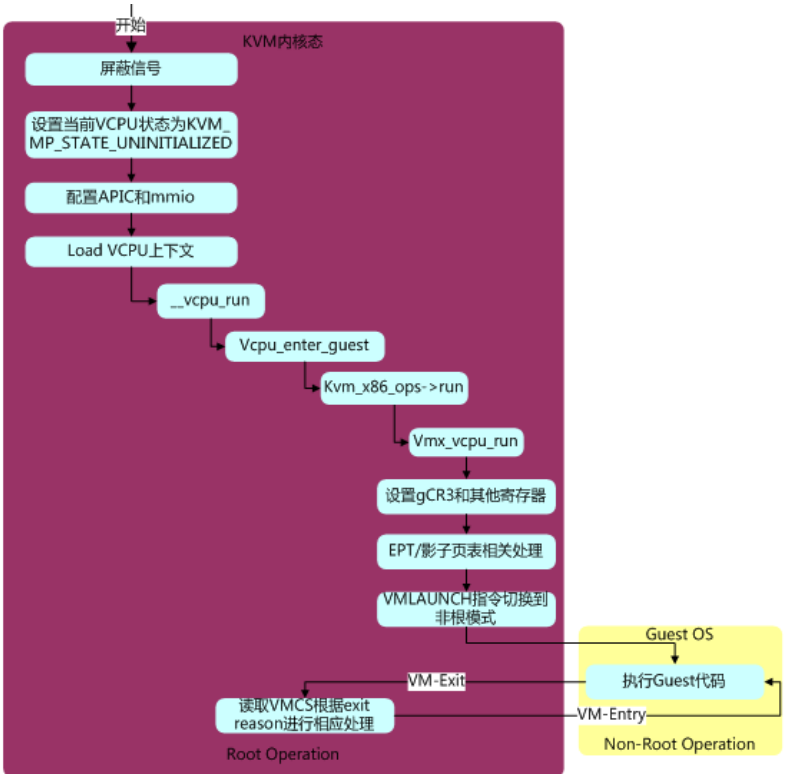
Qemu-kvm可以通过ioctl(KVM_RUN…)使虚拟机运行。过程如下：

ioctl(KVM_CREATE_VCPU..)

à kvm_vcpu_ioctl

à kvm_arch_vcpu_ioctl_run

具体由内核函数kvm_arch_vcpu_ioctl_run完成相关工作。主要流程如下：



1、 Sigprocmask()屏蔽信号，防止在此过程中受到信号的干扰。

2、 设置当前VCPU状态为KVM_MP_STATE_UNINITIALIZED

3、 配置APIC和mmio相关信息

4、 将VCPU中保存的上下文信息写入指定位置

5、 然后的工作交由__vcpu_run完成

6、 __vcpu_run最终调用vcpu_enter_guest，该函数实现了进入Guest，并执行Guest OS具体指令的操作。

7、 vcpu_enter_guest最终调用kvm_x86_ops中的run函数运行。对应于Intel平台，该函数为vmx_vcpu_run(设置Guest CR3和其他寄存器、EPT/影子页表相关设置、汇编代码VMLAUNCH切换到非根模式，执行Guest目标代码)。

8、 Guest代码执行到敏感指令或因其他原因(比如中断/异常)，VM-Exit退出非根模式，返回到vcpu_enter_guest函数继续执行。

9、 vcpu_enter_guest函数中会判断VM-Exit原因，并进行相应处理。

10、处理完成后VM-Entry到Guest重新执行Guest代码，或重新等待下次调度。

上一篇：KVM基本原理及架构七-KVM内核模块中重要的数据结构
下一篇：没有了

给主人留下些什么吧！~~

评论热议

请登录后评论。

登录 注册