昵称： Jessica程序猿
园龄： 5年10个月
粉丝： 544
关注： 27
+加关注

< 2020年4月 >

日	一	二	三	四	五	六
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

搜索

找找看

谷歌搜索

常用链接

我的随笔

我的评论

我的参与

最新评论

我的标签

随笔分类 (988)

C++(79)

C++ template(5)

C++ 容器(28)

C++构造函数(6)

C++面向对象编程(7)

C++训练(71)

C++重载与类型转换(9)

careercup(87)

dubbo(2)

Effective C++(3)

ext2文件系统系列(6)

flashcache(2)

KVM的初始化过程

之前打算整理一下在Guest VM, KVM, QEMU中IO处理的整个流程,通过查阅资料和阅读源码,已经大致知道IO在Guest KVM中的处理流程.当想要整理IO在KVM和QEMU中的处理时,发现很难理清楚QEMU和KVM之间的跳转和交互的过程,于是促使自己去了解QEMU和KVM启动的过程.(本文展示的代码中,qemu版本为1.6.0, linux内核版本为3.7.10)

为了介绍qemu和kvm的交互过程,我首先介绍一下kvm给用户提供的接口.kvm是一个内核模块,它实现了一个/dev/kvm的字符设备来与用户进行交互,通过调用一系列ioctl函数可以实现qemu和kvm之间的切换.当要创建一个新的虚拟机时,首先打开/dev/kvm设备,在其上调用ioctl函数:

[cpp] view plain copy 在CODE上查看代码片 派生到我的代码片

```
1. system_fd = open("/dev/kvm", ORDWR);
2. vm_fd = ioctl(system_fd, KVM_CREATE_VM, 0);
```

ioctl函数在kvm中的实现为virt/kvm/kvm_main.c中kvm_dev_ioctl函数,当传入的参数为KVM_CREATE_VM时,该函数会创建一个VM,并且返回一个fd,通过该fd可以操作虚拟机.

创建完虚拟机之后,需要在该虚拟机上面创建vcpu,调用的接口也是ioctl,只是此时对应的fd为创建虚拟机时返回的fd.

[cpp] view plain copy 在CODE上查看代码片 派生到我的代码片

```
1. vcpu_fd = ioctl(vm_fd, VM_CREATE_VCPU, 0)
```

此时ioctl函数对应的实现为virt/kvm/kvm_main.c中kvm_vm_ioctl函数,当传入的参数为VM_CREATE_VCPU时,与KVM_CREATE_VM过程类似,它创建一个vcpu并且返回可以操作该vcpu的fd.

创建完vcpu后,可以在该vcpu上面调用ioctl函数进入guest vm.

[cpp] view plain copy 在CODE上查看代码片 派生到我的代码片

```
1. ret = ioctl(vcpu_fd, KVM_RUN, 0);
```

此时ioctl函数对应的实现为virt/kvm/kvm_main.c中kvm_vcpu_ioctl函数,若传入的参数为KVM_RUN,它最终会调用vcpu_enter_guest函数进入guest vm.

qemu作为一个user mode的程序,其入口为main函数,该main函数定义在vl.c文件中.main函数比较长,其中跟KVM初始化相关的主要有两个函数:configure_accelerator()和machine->init(&args).cofigure_accelerator()函数选择运用哪一种虚拟化方案,其应用到的数据结构为accel_list,会调用accel_list[i].init函数.accel_list的初始化如下所示,当使用kvm虚拟化解决方案时,accel_list[i].init对应的函数即为kvm_init.

[cpp] view plain copy 在CODE上查看代码片 派生到我的代码片

```
1. static struct {
2.     const char *opt_name;
3.     const char *name;
4.     int (*available)(void);
5.     int (*init)(void);
6.     bool *allowed;
7. } accel_list[] = {
8.     { "tcg", "tcg", tcg_available, tcg_init, &tcg_allowed },
9.     { "xen", "Xen", xen_available, xen_init, &xen_allowed },
10.    { "kvm", "KVM", kvm_available, kvm_init, &kvm_allowed },
```

```
11.     { "qtest", "QTest", qtest_available, qtest_init, &qtest_allowed },
12. };
```

kvm_init函数定义在kvm-all.c文件中,其主要功能是打开/dev/kvm设备,创建一个虚拟机.

machine->init(&arg)函数主要初始化硬件设备,并且调用qemu_init_vcpu为每一个vcpu创建一个线程,线程执行的函数为qemu_kvm_cpu_thread_fn.从qemu main到qemu_init_vcpu之间函数调用关系涉及到一些函数指针的赋值源码比较难于读懂,以下是使用gdb调试打出其调用关系.

[cpp] view plain copy 在CODE上查看代码片 派生到我的代码片

```
1.  #0 qemu_init_vcpu (cpu=0x55555681ea90) at /home/dashu/kvm/qemu/qemu-dev-
    zwu/cpus.c:1084
2.  #1 0x0000555555909f1e in x86_cpu_realizefn (dev=0x55555681ea90, errp=0x7fffffff
    d8f8) at /home/dashu/kvm/qemu/qemu-dev-zwu/target-i386/cpu.c:2399
3.  #2 0x00005555556c768a in device_set_realized (obj=0x55555681ea90, value=true, e
    rr=0x7fffffffda88) at hw/core/qdev.c:699
4.  #3 0x000055555580b93f in property_set_bool (obj=0x55555681ea90, v=0x5555565bab2
    0, opaque=0x5555565375a0, name=0x555555a01f88 "realized", errp=0x7fffffffda88)
    at qom/object.c:1300
5.  #4 0x000055555580a484 in object_property_set (obj=0x55555681ea90, v=0x5555565ba
    b20, name=0x555555a01f88 "realized", errp=0x7fffffffda88) at qom/object.c:788
6.  #5 0x000055555580bbea in object_property_set_qobject (obj=0x55555681ea90, value
    =0x555556403e40, name=0x555555a01f88 "realized", errp=0x7fffffffda88) at qom/qo
    m-qobject.c:24
7.  #6 0x000055555580a770 in object_property_set_bool (obj=0x55555681ea90, value=tr
    ue, name=0x555555a01f88 "realized", errp=0x7fffffffda88) at qom/object.c:851
8.  #7 0x00005555558a7de0 in pc_new_cpu (cpu_model=0x555555a0200b "qemu64", apic_id
    =0, icc_bridge=0x55555655b2c0, errp=0x7fffffffdac8) at /home/dashu/kvm/qemu/qem
    u-dev-zwu/hw/i386/pc.c:922
9.  #8 0x00005555558a7fed in pc_cpus_init (cpu_model=0x555555a0200b "qemu64", icc_b
    ridge=0x55555655b2c0) at /home/dashu/kvm/qemu/qemu-dev-zwu/hw/i386/pc.c:978
10. #9 0x00005555558a923b in pc_init1 (system_memory=0x5555562a7240, system_io=0x55
    55562a7f60, ram_size=1073741824, boot_device=0x555555a0248a "cad", kernel_filen
    ame=0x0, kernel_cmdline=0x5555559f85be "",
11. initrd_filename=0x0, cpu_model=0x0, pci_enabled=1, kvmclock_enabled=1) at /home
    /dashu/kvm/qemu/qemu-dev-zwu/hw/i386/pc_piix.c:105
12. #10 0x00005555558a9a36 in pc_init_pci (args=0x7fffffffdf10) at /home/dashu/kvm/
    qemu/qemu-dev-zwu/hw/i386/pc_piix.c:245
13. #11 0x00005555558a9a7f in pc_init_pci_1_6 (args=0x7fffffffdf10) at /home/dashu/
    kvm/qemu/qemu-dev-zwu/hw/i386/pc_piix.c:255
14. #12 0x00005555558584fe in main (argc=10, argv=0x7fffffffe148, envp=0x7fffffffe1
    a0) at vl.c:4317
```

qemu_kvm_cpu_thread_fn函数创建vcpu,然后调用kvm_cpu_exec函数.kvm_cpu_exec函数调用ioctl进入kvm并最终进入guest vm.

以上即为qemu调用kvm的接口初始化kvm的过程.后续我会整理出IO在kvm和qemu之间执行过程,同时描述kvm和qemu之间如何协同工作的.

参考资料:

1. qemu-kvm的初始化与客户系统的执行:http://blog.csdn.net/lux_veritas/article/details/9383643

2. 内核虚拟化kvm/qemu----guest os,kvm,qemu工作流程:http://www.360doc.com/content/12/0619/13/7982302_219186951.shtml

转载：http://blog.csdn.net/dashulu/article/details/17074675

分类: 虚拟化

Jessica程序猿
关注 - 27
粉丝 - 544
+加关注

« 上一篇： linux删除文件未释放空间问题处理
» 下一篇： KVM虚拟机IO处理过程(一) ----Guest VM I/O 处理过程

posted @ 2015-07-30 16:20 Jessica程序猿 阅读(820) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

软件安装(31)
设计模式(12)
深度探索C++对象模型(17)
深入理解Linux内核(1)
数据结构与算法(60)
搜索引擎(2)
算法 每日一练(7)
算法导论(27)
文件系统(20)
虚拟化(11)
杂项(23)

随笔档案 (974)

2020年1月(1)
2019年2月(2)
2019年1月(1)
2018年8月(1)
2018年7月(1)
2018年6月(6)
2018年5月(7)
2018年4月(2)
2018年2月(3)
2017年12月(3)
2017年11月(1)
2017年9月(2)
2017年5月(2)
2017年4月(2)
2017年3月(3)
2017年2月(3)
2016年5月(4)
2016年4月(12)
2016年3月(3)
2016年2月(2)
2016年1月(6)

文章分类 (0)

linux

算法牛人

酷壳

C++11 中值得关注的几大变化（详解）

iTech's Blog

Not Only Algorithm，不仅仅是算法，关注数学、算法、数据结构、程序员笔试面试以及一切涉及计算机编程之美的内容 。

最新评论

1. Re:编写一个程序，从标准输入中读取若干string对象并查找连续重复出现的单词。所谓连续重复出现的意思是：一个单词后面紧跟着这个单词本身。要求记录连续重复出现的最大次数以及对应的单词

输入how cow，两个不同的单词，统计会出现问题。结果应该是没有连续出现的单词。

--想撸串的红杉树

2. Re:linux内核内存管理(zone_dma zone_normal zone_highmem)

你好，我想咨询您一个问题，32位系统中，用户进程可以访问物理内存的大小为3G，那么这3G有具体的空间范围吗？是物理内存的0-3G还是什么，因为我看0-896M是被内核线性映射，所以肯定表示0-3G，所...

--CV学习者

3. Re:spring中bean配置和bean注入

tql

--那么小晨呐

4. Re:如何使用jstack分析线程状态

thanks

--andyFly2016

5. Re:迪杰斯特拉算法介绍

不过有图挺好的，但我看到那里看不懂了。

--何所倚

阅读排行榜

1. spring中bean配置和bean注入(138538)

2. 如何使用jstack分析线程状态(89973)

3. maven快照版本和发布版本(29243)

4. C++ stringstream介绍，使用方法与例子(28518)

5. Linux用户空间与内核空间（理解高端内存）(27348)

评论排行榜

1. KVM虚拟机IO处理过程(一) ----Guest VM I/O 处理过程(9)

2. careercup-递归和动态规划 9.10(7)

3. 如何使用jstack分析线程状态(6)

4. spring中bean配置和bean注入(6)

5. 蜻蜓fm面试(6)

推荐排行榜

1. 如何使用jstack分析线程状态(20)

2. spring中bean配置和bean注入(19)

3. 数据库索引原理及优化(12)

4. maven快照版本和发布版本(8)

5. linux下的僵尸进程处理SIGCHLD信号(8)