


导航

博客园
首页
新随笔
联系
订阅 
管理

<	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	

公告

昵称： 钱小小
园龄： 5年5个月
粉丝： 0
关注： 1
[+加关注](#)

搜索

<input type="text"/>	<input type="button" value="找找看"/>
<input type="text"/>	<input type="button" value="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

我的标签

linux基础(3)
网络(3)
转载(3)
GRO(2)
linux内核(2)
nova(1)
open-falcon(1)
OpenStack(1)
iptables(1)
jsonrpc(1)
[更多](#)

【翻译】QEMU内部机制：顶层概览

系列文章：

1. [【翻译】QEMU内部机制：宏观架构和线程模型](#)
2. [【翻译】QEMU内部机制：vhost的架构](#)
3. [【翻译】QEMU内部机制：顶层概览\(本文\)](#)
4. [【翻译】QEMU内部机制：内存](#)

原文地址：<http://blog.vmsplice.net/2011/03/qemu-internals-big-picture-overview.html>

原文时间：2011年3月9日

作者介绍：Stefan Hajnoczi来自红帽公司的虚拟化团队，负责开发和维护QEMU项目的block layer, network subsystem和tracing subsystem。

目前工作是multi-core device emulation in QEMU和host/guest file sharing using vsock，过去从事过disk image formats, storage migration和I/O performance optimization

QEMU内部机制：顶层概览

在上一篇[【翻译】QEMU内部机制：宏观架构和线程模型](#)中，我直接介绍了QEMU的线程模型，并没有提到顶层的架构是什么样的。

本文将对QEMU顶层架构进行介绍，以帮助理解其线程模型。

vm的运行形式

vm是通过qemu程序创建的，一般就是qemu-kvm或kvm程序。在一台运行有3个vm的宿主机上，将看到3个对应的qemu

随笔分类

Apache全景分析(1)
QT(1)
安全
编程基础(27)
技术调研(19)
数据库(1)
虚拟化(13)

随笔档案

2020年2月(1)
2019年7月(14)
2019年6月(16)
2019年5月(1)
2019年4月(6)
2016年3月(2)
2015年12月(2)
2015年3月(12)
2014年12月(2)
2014年11月(9)

wxWidgets相关

wxWidgets简明教程
wxWidgets的linux安装配置
wxWidgets在线帮助

最新评论

1. Re:【转】extern "C"的含义和用法
nm test.so可以查看符号表

--钱小小

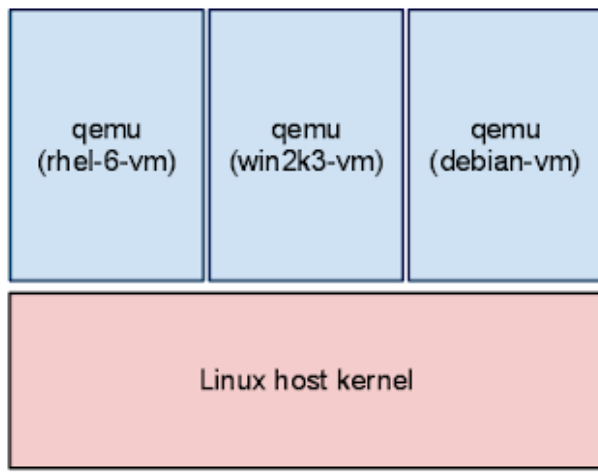
2. Re:“云端融合”思想的自我摸索（很不靠谱）
目前，云计算技术的发展日新月异，它可以促进信息技术和数据资源充分合理利用，是信息化发展的必然趋势。下面是我对于云计算与操作系统的简单想法，其中内容不乏失实及误导之陈述，请领导批评指正。 经过将近十年的...

--钱小小

3. Re:debian包的补丁管理工具：quilt
来自ubuntu的
dh_quilt_patch命令帮助：
NAME dh_quilt_patch -
apply patches listed in
debian/patches/seriesSY
NOP...

--钱小小

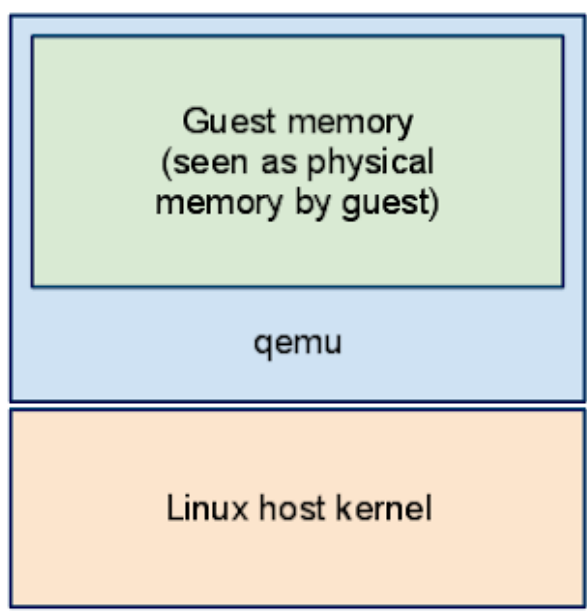
进程：



当vm关机时，qemu进程就会退出。vm重启时，为了方便，qemu进程不会被重启。不过先关闭并重新开启qemu进程也是完全可以的。

vm的内存

当qemu启动时，vm的内存就会被分配。-mem-path参数可以支持使用基于文件的内存系统，比如说hugetlbfs等。无论如何，内存会映射至qemu进程的地址空间，从vm的角度来看，这就是其物理内存。



qemu支持大端和小端字节序架构，所以从qemu代码访问vm内存时要小心。字节序的转换不是直接访问vm的内存进行的，而是由辅助函数负责。这使得可以从宿主机上运行具有不同字节序的vm。

kvm虚拟化

kvm是linux内核中的虚拟化特性，它使得类似于qemu的程序能够安全的在宿主cpu上直接执行vm的代码。但是必须得到宿

4. Re:在Linux下开发多语言软件(gettext解决方案)

附图bug修复报告:

--钱小小

阅读排行榜

1. golang类型断言的使用 (Type Assertion) (3770)
2. 进程占用过高cpu的排查(977)
3. 【转】QEMU Monitor机制实例分析(792)
4. 【转】理解qemu对设备的模拟机制(618)
5. 【翻译】QEMU内部机制: 宏观架构和线程模型(402)

评论排行榜

1. 在Linux下开发多语言软件(gettext解决方案)(1)
2. debian包的补丁管理工具: quilt(1)
3. “云端融合”思想的自我摸索(很不靠谱) (1)
4. 【转】extern "C"的含义和用法(1)

主机cpu的支持。当前, kvm在x86, ARMv8, ppc, s390和MIPS等CPU上都被支持。

为了使用kvm执行vm的代码, qemu进程会访问/dev/kvm设备并发起KVM_RUN ioctl调用。kvm内核模块会借助Intel和AMD提供的硬件虚拟化扩展功能执行vm的代码。当vm需要访问硬件设备寄存器、暂停访问cpu或执行其他特殊操作时, kvm就会把控制权交给qemu。此时, qemu会模拟操作的预期输出, 或者只是在暂停的vm CPU的场景下等待下一次vm的中断。

vm的cpu执行任务的基本流程如下:

```
open("/dev/kvm")
ioctl(KVM_CREATE_VM)
ioctl(KVM_CREATE_VCPU)
for (;;) {
    ioctl(KVM_RUN)
    switch (exit_reason) {
        case KVM_EXIT_IO: /* ... */
        case KVM_EXIT_HLT: /* ... */
    }
}
```

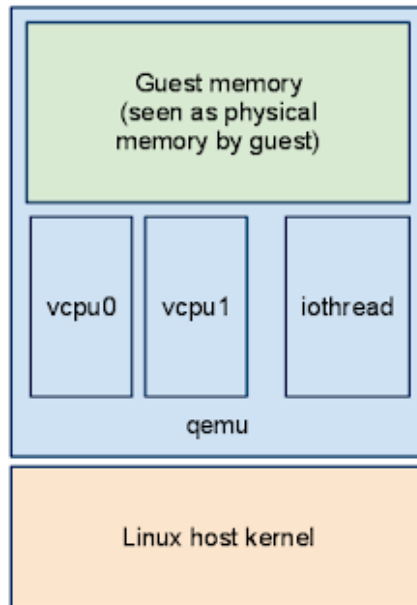
从宿主机视角来看vm的执行

宿主机内核会像调度其他进程一样来调度qemu进程。多个vm在完全不知道其他vm的存在的情况下同时运行。Firefox或Apache等应用程序也会与qemu竞争宿主机资源, 但可以使用资源控制方法来隔离qemu并使其优先级更高。

qemu在用户空间进程中模拟了一个完整的虚拟机, 它无从得知vm中运行有哪些进程。也就是说, qemu为vm提供了RAM、能够执行vm的代码并且能够模拟硬件设备, 从而, 在vm中可以运行任何类型的操作系统。宿主机无法“窥视”任意vm。

vm会在每个虚拟cpu上拥有一个vcpu线程。一个独立的iothread线程会通过运行select事件循环来处理类似于网络数据包或硬盘等IO请求。具体细节已经在上一篇[【翻译】QEMU内部机制: 宏观架构和线程模型](#)中讨论过。

下图是宿主机视角的qemu进程结构：



其他信息

希望本文能够让你熟悉qemu和kvm架构。

以下是两则介绍kvm架构的ppt，可以了解到更多细节：

[Jan Kiszka's Linux Kongress 2010: Architecture of the Kernel-based Virtual Machine \(KVM\)](#)

[我自己写的KVM Architecture Overview](#)

===精选评论===

->/dev/kvm设备是由谁打开的？vcpu线程还是iothread线程？

它是在qemu启动时由主线程调用的。请注意，每个vcpu都有其自己的文件描述符，而/dev/kvm是vm的全局文件描述符，它并非专属于某个vcpu。

->vcpu线程如何给出cpu的错觉？它是否需要在上下文切换时维护cpu上下文？或者说它是否必须硬件虚拟化技术的支持？

是的，需要硬件支持。kvm模块的ioctl接口支持操作vcpu寄存器的状态。与物理CPU在机器开启时具有初始寄存器状态一样，QEMU在复位时也会对vcpu进行初始化。

KVM需要硬件的支持，intel cpu上称为VMX，AMD cpu上称为SVM。二者并不兼容，所以kvm为二者分别提供了代码上的支持。

->hypervisor和vm如何实现直接交互？

这要看你是需要什么样的交互，virtio-serial可以用于任何vm/host的交互。

qemu guest agent程序建立于virtio-serial之上，他支持json rpc api。

它支持宿主机在vm中调用一系列命令（比如查询IP、备份应用等）

->能否解释vm中的一个应用的IO操作流程是怎样的么？

根据KVM还是TCG、MMIO还是PIO，ioeventfd是否启用，会有多种不同的代码路径。

基本流程是vm的内存或IO读写操作会“陷入”kvm内核模块，kvm从KVM_RUN的ioctl调用返回后，将返回值交给QEMU处理。

QEMU找到负责该块内存地址的模拟设备，并调用其处理函数。当该设备处理完成之后，QEMU会使用KVM_RUN的ioctl调用重入进vm。

如果没有使用Passthrough，那么vm看到的是一个被模拟出来的网卡设备，此时，kvm和qemu不会直接读写物理网卡。他们会将数据包交给linux的网络栈(比如tap设备)进行处理。virtio可以模拟网络、存储等虚拟设备，它会用一些优化方式，但是基本原理还是模拟一个“真实”设备。

->kvm内核模块是如何捕捉到virtqueue的kick动作的？

ioeventfd被注册为KVM的IO总线设备，kvm使用ioeventfd_write()通知ioeventfd。

trace流程为：

vmx_handle_exit with

EXIT_REASON_IO_INSTRUCTION

--> handle_io

--> emulate_instruction

--> x86_emulate_instruction

--> x86_emulate_insn

--> writeback

--> segmented_write

--> emulator_write_emulated

--> emulator_read_write_onepage

--> vcpu_mmio_write

--> ioeventfd_write

该流程显示了当vm kicks宿主机时，信号是如何通知ioeventfd。

原文如下：

1. **QEMU Internals: Big picture overview**

Last week I started the **QEMU Internals** series to share knowledge of how QEMU works. I dove straight in to the [threading model](#) without a high-level overview. I want to go back and provide the big picture so that the details of the threading model can be understood more easily.

1. **The story of a guest**

A guest is created by running the qemu program, also known as qemu-kvm or just kvm. On a host that is running 3 virtual machines there are 3 qemu processes:

When a guest shuts down the qemu process exits. Reboot can be performed without restarting the qemu process for convenience although it would be fine to shut down and then start qemu again.

1. **Guest RAM**

Guest RAM is simply allocated when qemu starts up. It is possible to pass in file-backed memory with `-mem-path` such that `hugetlbfs` can be used. Either way, the RAM is mapped in to the qemu process' address space and acts as the "physical" memory as seen by the guest:

QEMU supports both big-endian and little-endian target architectures so guest memory needs to be accessed with care from QEMU code. Endian conversion is performed by helper functions instead of accessing guest RAM directly. This makes it possible to run a target with a different endianness from the host.

1. **KVM virtualization**

KVM is a virtualization feature in the Linux kernel that lets a program like qemu safely execute guest code directly on the host CPU. This is only possible when the target architecture is supported by the host CPU. Today KVM is available on x86, ARMv8, ppc, s390, and MIPS CPUs.

In order to execute guest code using KVM, the qemu process opens `/dev/kvm` and issues the `KVM_RUN` ioctl. The KVM kernel module uses hardware virtualization extensions found on modern Intel and AMD CPUs to directly execute guest code. When the guest accesses a hardware device register, halts the guest CPU, or performs other special operations, KVM exits back to qemu. At that point qemu can emulate the desired outcome of the operation or simply wait for the next guest interrupt in the case of a halted guest CPU.

The basic flow of a guest CPU is as follows:

```
open("/dev/kvm")
ioctl(KVM_CREATE_VM)
```

```
ioctl(KVM_CREATE_VCPU)
for (;;) {
    ioctl(KVM_RUN)
    switch (exit_reason) {
        case KVM_EXIT_IO: /* ... */
        case KVM_EXIT_HLT: /* ... */
    }
}
```

1. The host's view of a running guest

The host kernel schedules qemu like a regular process. Multiple guests run alongside without knowledge of each other. Applications like Firefox or Apache also compete for the same host resources as qemu although resource controls can be used to isolate and prioritize qemu.

Since qemu system emulation provides a full virtual machine inside the qemu userspace process, the details of what processes are running inside the guest are not directly visible from the host. One way of understanding this is that qemu provides a slab of guest RAM, the ability to execute guest code, and emulated hardware devices; therefore any operating system (or no operating system at all) can run inside the guest. There is no ability for the host to peek inside an arbitrary guest.

Guests have a so-called **vcpu** thread per virtual CPU. A dedicated **iothread** runs a `select(2)` event loop to process I/O such as network packets and disk I/O completion. For more details and possible alternate configuration, see the [threading model](#) post.

The following diagram illustrates the qemu process as seen from the host:

1. Further information

Hopefully this gives you an overview of QEMU and KVM architecture. Feel free to leave questions in the comments and check out other **QEMU Internals** posts for details on these aspects of QEMU.

Here are two presentations on KVM architecture that cover similar areas if you are interested in reading more:

1. Jan Kiszka's Linux Kongress 2010 presentation on the [Architecture of the Kernel-based Virtual Machine \(KVM\)](#). Very good material.
2. My own attempt at presenting a [KVM Architecture Overview](#) from 2010.

分类: [虚拟化](#)

好文要顶

关注我

收藏该文



钱小小

关注 - 1

粉丝 - 0

[+加关注](#)

0

0

« 上一篇: [【翻译】QEMU内部机制: 宏观架构和线程模型](#)

» 下一篇: [【转】理解qemu对设备的模拟机制](#)

posted on 2019-06-20 13:09 钱小小 阅读(249) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#) [网站](#) [首页](#)。

【推荐】超50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库

【推荐】腾讯云产品限时秒杀，爆款1核2G云服务器99元/年！

相关博文：

- [QEMU,KVM及QEMU-KVM介绍](#)
- [Qemu 简述](#)
- [AjaxPro 内部机制探讨](#)
- [浅谈SQLServer内部运行机制](#)
- [理解 QEMU/KVM 和 Ceph \(1\)：QEMU-KVM 和 Ceph RB...](#)
- » [更多推荐...](#)

最新 IT 新闻：

- [特斯拉经营范围新增电信业务等 并正式迁入上海自贸区](#)
- [任正非最新讲话：艰苦奋斗的目的是过幸福生活](#)
- [聚美优品宣布完成私有化，正式从纽交所退市](#)
- [智能高空作业机器人公司史河科技完成3500万元Pre A+轮融资](#)
- [亚马逊下调广告营销联盟佣金费率 或重创出版商营收](#)
- » [更多新闻...](#)

Powered by:
[博客园](#)

Copyright © 2020 钱小小
Powered by .NET Core on Kubernetes