QEMU-KVM 虚拟 PCI 设备优化方法

车翔　王华军

（成都理工大学　信息科学与技术学院，四川　成都　610059）

[摘　要] 随着 QEMU-KVM 虚拟机的不断普及，虚拟机开发者会将各种 PCI 设备集成到 QEMU 中，因此在 QEMU 中如何来虚拟一个期望的硬件主板环境就变得非常重要。本文主要研究了 QEMU 如何虚拟 PCI 设备，并且成功虚拟一个带 PCI-PCI 桥的 PCI 设备；另一方面本文针对 QEMU 无法给 PCI 设备分配指定 IO 地址问题提出了有效解决方案并且进行了优化。

[关键词] QEMU-KV 虚拟机；虚拟化；PCI 设备；IO 地址

1. 概述

KVM（Kernel-based Virtual Machine）是一个基于 Linux 内核的虚拟机。它采用基于 Intel-VT 技术或 AMD 技术的虚拟化硬件，并结合修改过的 QEMU 来提供设备虚拟化。KVM 的特点在于充分利用最新的支持虚拟化的硬件，与 Linux 内核结合得很好，继承了 Linux 的大部分功能，能够充分利用 Linux 内核一直在不断优化和改进的技术优势。

QEMU 作为 KVM 的模拟器，帮助 KVM 模拟各种硬件设备，而随着 QEMU 的发展，它模拟的设备越来越多。本文详细地分析了 QEMU 中 X86 架构下 PCI 总线、PCI 桥和 PCI 设备的虚拟化过程，并针对 PCI 设备分析了简单的 IO 地址初始化过程。但是由于虚拟化中 PCI 设备的 IO 基地址是动态分配，而我们可能需要将设备 IO 基地址固定在某个区间，基于这个需求本文提出了一种固定 PCI 设备 IO 基地址的方法，它能够跳过 PCI 基地址的重新设置，达到我们预期的目的。

2. PCI结构简介

每一个 PCI 设备都对应一段内存空间，里面按照地址位置放置 PCI 设备的信息，包括厂家信息、bar 信息、中断信息等等，也可以理解成一个数组。一些设备一出厂，相关的信息已经写在里面，我们这里模拟设备，所有这些信息我们都要进行动态的读和写。在这里只列出了本文相关的数据结构。

表 1　PCI 数据结构表

	0x0	0x04	0x08	0xc
0x00	vendor ID dev ID	command		
0x01	bar0 address	bar1 address	bar2 address	bar3 address
0x20	bar4 address	bar5 address		
0x30			interrupt line	

另外，我们可以在 Linux 中使用 lspci -x 看到 PCI 设备的相关内存数据信息。

3. PCI设备虚拟化的实现

3.1 PCI 总线虚拟化

QEMU 在初始化硬件的时候，最开始的函数就是 pc_init1。在这个函数里面会相继的初始化 CPU、中断控制器、ISA 总线，然后就要判断是否需要支持 PCI，如果支持则调用 i440fx_init 初始化 PCI 总线。

i440fx_init 函数主要参数就是之前初始化好的 ISA 总线以及中断控制器，返回值就是 PCI 总线，之后就可以将自己喜欢的设备统统挂载在这个上面。

另外，在 Linux 里面我们可以使用 lspci -t 来看 PCI 总线的结构图。

3.2 PCI-PCI 桥的构造

在 QEMU 中，所有的设备包括总线、桥，一般设备都对应一个设备结构，通过 register 函数将所有的设备链接起来，就像 Linux 的模块一样，在 QEMU 启动的时候会初始化所有的 QEMU 设备，而对于 PCI 设备来说，QEMU 在初始化以后还会进行一次 RESET，将所有的 PCI bar 上的地址清空，然后进行统一分配。

QEMU（x86）里面的 PCI 的默认 PCI 设备都是挂载在主总线上的，没有使用 PCI-PCI 桥，而桥的作用一般也就是连接两个总线，然后进行终端和 IO 的映射。为了方便起见，可以使用 PPC 架构里面的 DEC 桥，关键操作就是要包含 DEC 的头文件，修改 x86 下面的配置文件，将 DEC 强行的配置下去，这种 i440FX 加 DEC 的组合在虚拟设备上面确实行得通。

3.3 QEMU 下实现 PCI 设备的虚拟化

一般的 PCI 设备其实和桥很像，甚至更简单，区分桥和一般设备的关键地方就是 class 属性和 bar 地址。下面是一个标准的 PCI 设备结构：

```
static PCIDeviceInfo fpga_info={
    .qdev.name = "fpga",
    .qdev.size = sizeof(FPGAState),
    .init = pci_fpga_init,
};
static void fpga_register_devices(void){
   pci_qdev_register(&fpga_info);
}
device_init(fpga_register_devices)
```

在上面的过程中，pci_fpga_init 函数在之前的文章中描述过就不再展开叙述了，其中主要的一项就是给 bar 分配 IO 地址，调用函数如下：

```
pci_register_bar(&s->dev,0,0x800,
PCI_BASE_ADDRESS_SPACE_IO,fpga_ioport_map);
```

其中第一个参数是设备；第二个参数是 bar 的编号，每个

作者简介：车翔，男，江西新余人，硕士。主要研究方向：虚拟机开发。

PCI 设备有 6 个 bar，对应 0-5，这个我们也可以在上面的 PCI 基本结构中看到这 6 个 bar，也就是后文中提到的 6 个 region；第三个参数是分配的 IO 地址空间范围；第四个参数是表示 IO 类型是 PIO 而不是 MMIO；最后一个参数是 IO 读写映射函数。

从上面可得知这里并没有给设备分配 IO 空间的基地址，只有一个空间长度而已，这也进一步说明 PCI 设备在 QEMU 中一般是随机动态分配空间的，通过不断地 Update Mapping 来更新 IO 空间的映射。

当 PCI 设备结构都构造好以后，就可以通过 pci_create_simple_multifunction(sub_bus, -1, true,"fpga")); 来挂载我们设备了，这里的 sub_bus 就是我们之前通过创建桥得到的子总线。

经过上面的初始化我们就得到了系统的主 PCI 总线 b 了，接着就可以挂载我们的设备。

4. 固定PCI设备的IO基地址

要给 PCI 设备分配固定的 IO 基地址，需要先了解 PCI 设备是如何刷新和分配 IO 基地址的。?PCI 在需要的时候，如第一次启动时，IO 重叠等就需要重置 PCI 设备，并且清空 PCI bar 上面的地址信息。主要调用函数 pci_device_reset。而设备的地址分配又是受桥的地址分配约束的，一旦桥的地址分配了，设备的地址只能分配在桥的范围内，否则就会被置为无效，然后重新分配，直到分配在桥的范围内为止。所以只要固定了桥的地址，自然就固定了设备的地址。

因而只需要初始化桥的地址，并且在 reset 的时候跳过桥的基地址重置，就能实现设备和桥地址的固定。添加的函数和代码如下：

添加桥的初始地址，因为桥的地址固定写在 bar3 上，通过写 20 可以将基地址固定在 0x2000 上，同时还需要写命令位，置 1。

```
static int dec_21154_initfn(PCIDevice *dev){
    ... ...    ... ...
    pci_set_word(dev->config + PCI_BASE_ADDRESS_3, 0x2020);
    pci_set_word(dev->config + PCI_COMMAND,0x1);
    void pci_device_reset(PCIDevice *dev)
    return 1;
}
```

在重置桥里面过滤我们的桥，通过 dev 的名字可以识别我们自己定义的设备，如果是我们的设备就不需要重置，直接更新 IO 映射。

```
void pci_device_reset(PCIDevice *dev){
  if(strcmp(dev->name,"dec_name")==0){
    pci_update_mappings(dev);
    return
  }
  ... ...    ... ...
}
```

通过上面的步骤就能实现一般的 IO 基地址固定，我们可以在 Linux 中使用 cat /proc/ioports 命令来查看当前 PCI 设备的 IO 映射地址关系。

5. 总结

本文介绍了基本的 PCI 设备，并依次讲述了 QEMU 中如何实现总线、桥和设备的模拟过程。在 QEMU 中有一根根总线，然后通过桥连接到 PCI 总线，或者 ISA 总线。本文通过 PCI-PCI 桥从根总线派生出 PCI 总线，然后挂载虚拟的 PCI 设备，在初始化设备时设置 PCI 设备的 IO 地址，并刷新看是否在指定的桥 IO 地址范围内。本文针对这个特点跳过 PCI 设备的 IO 地址刷新，强行固定分配 PCI 设备的 IO 地址以达到我们的需求，这对一些固定的操作系统有更强的兼容性。另外也在一定的程度上帮助我们更深入地理解了 PCI 设备，理解了硬件与操作系统的 IO 交互。

参考文献：

[1] 英特尔开源软件技术中心,复旦大学并行处理研究所著. 系统虚拟化[M]. 北京：清华大学出版社.2009.

[2] Intel Corporation The Intel® 64 and IA-32 Architectures Software Developer's Manual.2009.

[3] BOVET&CESATI 著.深入理解 Linux 内核[M]. 北京：中国电力出版社,2010.

[4] KVM 开源项目 [OL]. http://www.linux-kvm.org/page/Main_Page, 2010.

[5]QEMU 开源项目[OL]. http://wiki.qemu.org/Main_Page,2010.

[6] Naba Barkakati, Red Hat LINUX 核心技术精解[M] 北京：中国水利水电出版社.1999.

[7] Mel Gorman., 深入理解 Linux 虚拟内存管理[M]. 北京：航空航天大学出版社.2006.

Optimization of the PCI Device Virtualization of QEMU-KVM

Che Xiang Wang Huajun

(Chengdu University of Technology , Chengdu 610059, Sichuan)

【 **Abstract** 】 In pace with popularization of QEMU-KVM virtual machine, the developer will integrate some new PCI Devices to QEMU. So it will be more and more important to virtualize an expected hardware environment. This paper focuses on researching how to virtualize a PCI device with PCI-PCI Bridge in QEMU successfully. In the other hand, this paper provides a effective solution to allocate a fix IO address to PCI device and gives some advice for improving.

【 **Keywords** 】 QEMU-KVM; virtualization; PCI device; IO address