一种基于 SR-IOV 扩展的 VF 分配模型

李 南[1] 封卫兵[1,2] 张 武[1,2] 彭俊杰[1]
[1](上海大学计算机工程与科学学院 上海 200072)
[2](上海大学高性能计算机中心 上海 200072)

摘 要 I/O 的性能是许多从事虚拟化研究的学者致力于解决的难题之一。单根 I/O 虚拟化 SR-IOV(Single Root I/O Virtualization)技术规范的提出,使得 I/O 虚拟化的性能得到了较大改善。但由于客户域单独占有虚拟功能(VF),而相关设备的 VF 数目及网络设备带宽有限,影响了其可扩展性和客户域 I/O 性能。针对这种现状,提出一种扩展的 VF 分配模型。基于 Xen,通过在主域(Dom 0)SR-IOV 管理器(IOVM)中增加虚拟功能管理模块(VFM),来实现多个客户域虚拟独占拥有较大带宽的 VF。采用 Netperf 工具进行测试,实验结果表明,该模型能够较好地解决 VF 不足和 VF 增加带宽迅速下降的问题。

关键词 单根 I/O 虚拟化 虚拟化 虚拟功能 分配模型

中图分类号 TP316 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2013.11.007

A VF ALLOCATION MODEL BASED ON SR-IOV EXTENSION

Li Nan[1] Feng Weibing[1,2] Zhang Wu[1,2] Peng Junjie[1]
[1](*School of Computer Engineering and Science, Shanghai University, Shanghai 200072, China*)
[2](*High Performance Computing Center, Shanghai University, Shanghai 200072, China*)

Abstract I/O performance is one of the difficulties to be solved committed by many scholars engaging in virtualisation research. The performance of I/O virtualisation has been greatly improved with the presentation of the technology specification in single root I/O virtualisation (SR-IOV). However, due to the mechanism that guest domains access virtual function (VF) exclusively, and the numbers of VF of correlated device and the bandwidth of network device are limited, the scalability and I/O performance of guest domains are impacted. In view of this situation, an extended VFs allocation model is proposed. Based on Xen, virtual function management module (VFM) is added to the SR-IOV manager of main domain (Dom 0), in this way it is achieved that multiple guest domains could own a VF with larger bandwidth virtual exclusively. Netperf tool is used in test, and the experimental result shows that this model can well solve the problems of insufficiency in VF numbers and the bandwidth of VF declines rapidly while the number of VF is expanding.

Keywords SR-IOV Virtualisation Virtual function Allocate model

0 引 言

当前虚拟化技术得到飞速发展,虚拟机性能也得到了巨大改善。虚拟化技术主要包括三方面:即 CPU 虚拟化、内存虚拟化和 I/O 虚拟化。随着 Intel VT 和 AMD SVM 技术的推出,原先的 X86 体系在虚拟化方面的缺陷得到很好的改善,CPU 虚拟化性能也得到提升;内存虚拟化方面可以采用直接模式或者影子模式获得较好的性能[3];I/O 虚拟化主要有软件设备模拟[4]、前后端驱动(Split I/O)[3]、直接分配(Passthrough I/O)[9]等实现技术,虚拟机性能各不相同。前两者已经得到了较好的发展,相关方面虚拟机性能也能较好满足当前需求,而 I/O 虚拟化性能则成为了虚拟化技术发展的主要瓶颈[6]。从当前 I/O 虚拟化技术发展趋势来看,其主要沿着软件级虚拟化技术、处理器级虚拟化支持、芯片级(chipset)虚拟化支持到 I/O 硬件设备级支持的方向发展[5],即逐渐减少或脱离虚拟机监视器(VMM)的干预,从而提高 I/O 虚拟化性能。由于不同的 I/O 硬件厂商可能采用不同的技术和策略来实现 I/O 硬件级虚拟化,国际组织 PCI-SIG (Peripheral Component Interconnect Special Interest Group)提出了 SR-IOV 规范[1]来促进 I/O 硬件级虚拟化技术的发展。

本文简要介绍了 SR-IOV 规范及相关技术,同时针对支持 SR-IOV 规范的设备 VF 数量不足以及 VF 数量增大带宽迅速下降的问题,提出了一种扩展的 VF 分配模型,并进行实验分析。

1 相关研究

1.1 Passthrough I/O

Passthrough I/O[7],即穿透 VMM,直接将 I/O 设备分配给虚拟机,而不是像 Direct I/O 模型(VMware Workstation 采用)和

收稿日期:2012-09-20。国家自然科学基金项目(61103054);上海市重点学科建设基金项目(J50103)。**李南**,硕士生,主研领域:云计算及虚拟化。**封卫兵**,副教授。**张武**,教授。**彭俊杰**,副教授。

Split I/O 模型(半虚拟化 Xen 采用)那样,客户域使用的 I/O 是逻辑上虚拟出来的。Passthrough I/O 技术的实现需要 IOMMU 等硬件技术的支持[11],它能将客户域物理内存中的内存映射 IO(MMIO)映射到主机物理内存中。其模型如图 1 所示。

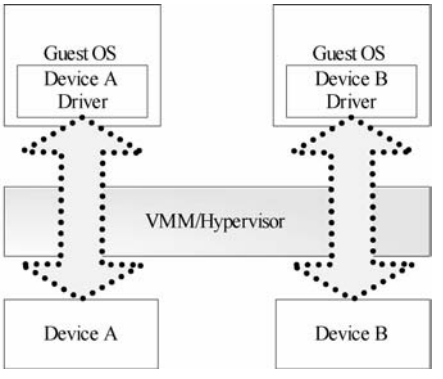

图 1　Passthrough I/O 模型

1.2　SR-IOV 规范

虚拟机通过 Passthrough I/O 模式访问 I/O 设备,能够获得非常好的 I/O 性能[5]。但是这种方式下 I/O 设备的共享率比较低,每个设备只能被一个虚拟机所独立占有,将导致较高的硬件成本,同时脱离了虚拟化的实质。为此,SR-IOV 技术在继承了 Passthrough I/O 技术的同时,有效地实现了 I/O 设备的共享性。将 Passthrough I/O 技术所需要的多个物理设备转变为一个支持 SR-IOV 技术的设备,这样的设备具有一个或几个物理功能(PF)和多个虚拟功能(VF),每个虚拟功能可以作为一个独立的 I/O 设备使用。

支持 SR-IOV 技术的设备是基于 PCIe 规范的[2],其包含三个主要部分,PCIe PF(Physical function),VF(Virtual function)和 PCIe 交换器(PCIe Switch)[8]。PF 具有标准的 PCIe 功能,在 PCIe 总线上是主要的实体,具有唯一的申请标识 RID;VF 是具有基本 PCIe 功能的轻量级实体,能够实现大量数据的传输,由 PF 负责建立与管理,一经创建,也将具有唯一的申请标识 RID;PCIe 交换机则主要负责数据能从指定的 VF/PF 转入或转出。

基于 SR-IOV 技术实现 I/O 虚拟化,Xen 的实现模型如图 2 所示:PF 驱动和 SR-IOV 管理器(IOVM)位于主域中,VF 驱动可为客户域中的一般设备驱动,通常一个客户域对应一个 VF。PF 驱动用于创建和管理 VF,IOVM 主要管理每个 VF 的 MMIO 配置空间。

VF 的创建和配置过程如下:

1) 主域初始化 SR-IOV 设备,开始 PCIM 只能识别 PF,并为之初始化 PCIe 配置空间;

2) 通过主域设置 VF 数目,PF 创建相应数目的 VF,系统为每个 VF 分配 MMIO 配置空间,此时所有的 VF 相当于独立的 PCIe 设备,能实现具体数据的转入转出,并为主域所占有;

3) 为客户域分配 VF。两种方式:静态分配 VF,即在客户域启动之前,将相应的 VF 从主域(Dom0)中隐藏(剥夺),通过配置将其分配给相应客户域。在客户域启动后,则其就拥有了该 VF;动态分配 VF,即在客户域启动之后再动态地为其分配 VF,此时 VF 是以 hot-plug 的方式分配给客户机的。

4) VF 分配给具体客户域后,客户域可以直接访问相应设备寄存器进行 I/O 操作。

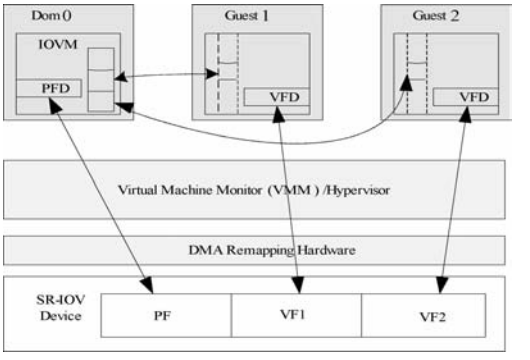

图 2　SR-IOV 模型

2　一种扩展的 VF 分配模型

2.1　模型的提出

SR-IOV 虚拟化技术具有较多的优势,同时也有一些不足。首先,支持 SR-IOV 技术的设备的 VF 数量一定,随着客户域的增多,将无法为每个客户域 Passthrough 方式分配 VF;其次,SR-IOV 虚拟化技术相当于把一个较大带宽(网络设备)设备划分为多个小带宽的物理设备来使用,当 VF 数量增多时,VF 平均带宽相应减小,可能会影响客户域 I/O 数据传输效率。因此,需要考虑 VF 数量不足的问题。

解决该问题最直观的方法则是对于后来无法分配到 VF 的客户域使用 Xen 传统的前后端驱动的方式来共享 PF,但是这样的话可能会一定程度上增加 PF 的负担,同时客户域需要添加相应的前端驱动,并增加了 VMM 的干预,导致客户域的 I/O 性能受到较大的影响。

我们提出了一种扩展的 VF 分配模型,基本思想为:对于性能要求较高的客户域,先利用 Passthrough 方式分配 VF,而其他性能要求较低的客户域,则采用虚拟 Passthrough(VPIO[9])方式使用 VF,即多个客户域共享一个 VF。其实现模型如图 3 所示,客户域 1 和客户域 2 独立占有 VF1 和 VF2,客户域 3 和客户域 4 对 I/O 性能要求较低,借助新加的 VFM 模块实现虚拟 Passthrough VF3。

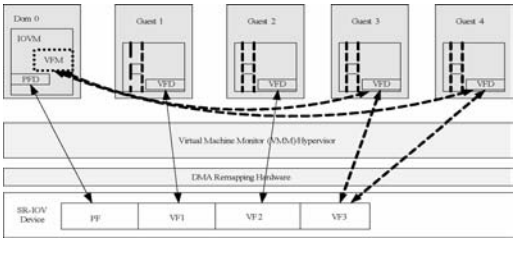

图 3　扩展的 SR-IOV 模型

2.2　算法描述

为客户域分配 VF,采用的是动态(hot-plug)方式。此外,默认为 I/O 性能要求更高的客户域先分配 VF。具体过程即算法如下所示。当可用的 VF 数目大于 0 时,则直接采用 Passthrough 方式为该客户域分配 VF,并更新 VFM 中 VF 信息;当没有可用的 VF 时,此时需要借助 VFM 模块实现多个客户域共享一 VF:首先计算各个 VF 在一定时间内的网络吞吐率,然后对其进行由低到高的链式排序,最后将网络吞吐率即利用率最低的 VF 分配给该客户域,即该客户域和其他客户域共享该 VF,并更新 VF 信息。

1）客户域启动；
2）if(vf_n > 0)
3）选取一可用 VF 直接分配给该客户域；
4）更新 VF 信息；
5）else
6）计算过去一定时间内每个 VF 的网络吞吐率；
7）按照网络吞吐率从低到高对 VF 进行排序；
8）将网络吞吐率最低的 VF 分配给该客户域；
9）更新 VF 信息；
10）分配完成。

3 模型实现

IOVM 中增加一个 VF 管理模块(VFM),用来记录每个 VF 的基本信息,包括是否处于空闲,I/O 处理量,网络吞吐率,最近重置时间,对应的客户域等。其数据结构的主要字段及 VF 网络吞吐率计算方法如下：

```
/* VF 设备信息 */
struct vf_info{
    //当前使用 VF 的客户域
unsigned int vf_used_no;
long throughput;                          // VF 网络吞吐率
time_t last_time;                         // 最近重置时间
long io_size;                             // 处理 IO 的字节数
int dom_count;                            // 对应的客户域数
int vf_num;                               // 对应 SR-IOV 的 VF
……

/*计算 VF 的网络吞吐率*/
static void vf_throughtput (struct vf_info *vfinfo_pt){
long time;
time = now - last_time;
//计算 time 时间内 VF 吞吐率
vfinfo_pt ->throughput = (vfinfo_pt ->io_size)/time;
}
```

图 4 为客户域访问 VF 流程。

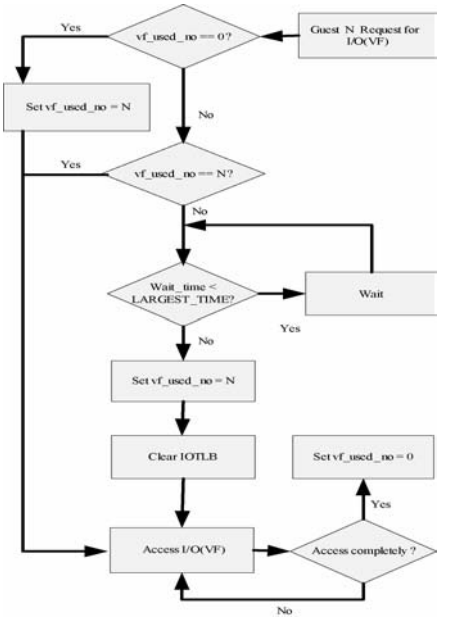

图 4 客户域访问 VF 流程

VFM 模块还需要定期对所有 VF 信息进行重置,主要是最近重置时间和处理 IO 的字节数(重置为 0)两个字段,以便获得更加时效的 VF 网络吞吐率,通过采用这种 VF 分配模型,单独占有 VF 的客户域可以正常通过 Passthrough 方式访问 VF,共享 VF 的客户域使用 VF 流程如图 4。

4 实验结果与分析

实验环境为:采用 Xen 4.1.0 版本,宿主机为 Ubuntu 10.04,客户域为 CentOS 5.5, Linux 内核均升级为 linux 2.6.38,支持 SR-IOV 虚拟化技术。CPU 为支持虚拟化的 Intel(R) Core(TM) 2 Quad 2.66GHz 4 核, 内存为 4GB, 网络设备为支持 SR-IOV 的 Intel 82576 Gigabit 网卡。客户域配置为 VCPU 双核,虚拟内存 2GB。借助网络性能测试软件 Netperf 进行端到端测试,测试时间为 10 分钟。

正常使用 SR-IOV 技术实验结果如表 1、表 2 所示。

表 1 正常 SR-IOV 3 客户域的吞吐率

	VF 带宽(Mb/s)	吞吐率(Mb/s)
Dom0		940.69
Guest 1	400(VF1)	374.25
Guest 2	400(VF2)	373.82
Guest 3	200(VF3)	186.34

表 2 正常 SR-IOV 4 客户域的吞吐率

	VF 带宽(Mb/s)	吞吐率(Mb/s)
Dom0		938.46
Guest 1	400(VF1)	369.89
Guest 2	400(VF2)	372.54
Guest 3	100(VF3)	92.66.
Guest 4	100(VF4)	91.71

采用扩展模型 Guest3 和 Guest4 共享 VF3,实验结果如表 3、表 4 所示。

表 3 扩展的 SR-IOV 4 客户域的吞吐率

	VF 带宽(Mb/s)	吞吐率(Mb/s)
Dom0		922.34
Guest 1	400(VF1)	371.57
Guest 2	400(VF2)	373.80
Guest 3	200(VF3)	81.58
Guest 4	200(VF3)	79.62

表 4 扩展的 SR-IOV 分时段 4 客户域的吞吐率

	VF 带宽(Mb/s)	吞吐率(Mb/s)	
		前 5 分钟	后 5 分钟
Dom0		930.48	
Guest 1	400(VF1)	373.51	
Guest 2	400(VF2)	369.91	
Guest 3	200(VF3)	182.44	0
Guest 4	200(VF3)	0	183.29

本实验主要考虑两客户域共享一个 VF 的情形。从表 1、表 2 数据可知,采用 SR-IOV 技术客户域可以获得非常好的 I/O 性能,而随着 VF 数量的增多,VF 平均带宽相应减小。从表 2、表

3、表 4 数据可知,采用扩展的 VF 分配模型,当共享同一 VF 的客户域同时进行 I/O 访问时,性能下降并不明显;考虑到共享同一 VF 的多个客户域对 I/O 实时性访问要求较低,因而当客户域需要访问 I/O 设备进行数据传输时,可以获得较快的传输效率,减少等待时间。

5　结　语

I/O 虚拟化的关键在于解决 I/O 设备与虚拟机数据交换的问题,而这部分主要相关的是 DMA 直接内存存取,以及 IRQ 中断请求。基于 SR-IOV 的 I/O 虚拟化,采用了 IOMMU 技术,实现了 DMA 重映射,以及采用重新定义的 MSI/MSI-x 消息中断格式,使得客户域可以直接使用 I/O 设备,很好地解决了上述两方面的问题,因此基于 SR-IOV 的 I/O 虚拟化将成为今后该方面虚拟化技术发展的主流[10]。针对 SR-IOV 设备的 VF 数目不足或 VF 数目增多带来的带宽下降等问题,对于不同客户域对 I/O 设备要求的差异,本文所提出的 VF 分配模型可以较好地解决。

参　考　文　献

[1] Single Root I/O Virtualization and Sharing 1.1 specification[S/OL]. 2012 - 09. http://www.pcisig.com/ specifications/iov/single_root/.

[2] PCI Express Base 3.0 specification[S/OL]. 2012 - 09. http://www.pcisig.com/specifications/pciexpress/ base3/.

[3] Paul B, Boris D, Keir F, et al. Xen and the art of virtualization[J]. ACM, 2003(10): 164 - 177.

[4] Sugerman J, Venkitachalam G, Lim B H. Virtualizing I/O devices on VMware workstation's hosted virtual machine monitor[C]//Proceedings of the USENIX Annual Technical Conference, 2001.

[5] Dong Y, Dai J, et al. Towards high-quality I/O virtualization[C]//Proceeding of the Israeli Experimental Systems Conference (SYSTOR), 2009.

[6] Jeffrey S. I/O virtualization bottlenecks in Cloud Computing today[C]// Proceeding of the Second Workshop on I/O Virtualization (WIOV), 2010.

[7] Ram K K, Santos J R, Turner Y, et al. Achieving 10Gb/s using safe and transparent network interface virtualization[C]//Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on virtual execution environments, 2009.

[8] Jun S, Yoichi H, Junichi H. Multi-Root share of Single-Root I/O Virtualization (SR-IOV) compliant PCI Express device[C]//2010 18th IEEE Symposium on High Performance Interconnects, 2010.

[9] Xia L, Lange J, Dinda P, et al. Investigating virtual Passthrough I/O on commodity devices[J]. ACM SIGOPS Operating Systems Review, 2010: 83 - 94.

[10] Dong Y Z, Yang X W, Li J H, et al. High performance network virtualization with SR-IOV[J]. Parallel Distribute Computing, 2012.

[11] Dong Y Z, Zheng X D, Zhang X T, et al. Improving virtualization performance and scalability with advanced hardware accelerations[J]. IEEE, 2010.

(上接第 3 页)

由图 7 可知,系统工作时,DMA 将存放在 RAM 中的一次图像数据送入 TC,等到 TC 中数据处理完毕时,会自动发出中断信号,C * CORE340 处理器收到中断信号后,经 DMA 将处理结果送到 Frame Buffer 中显示。然后,DMA 将新的图像数据送至 TC 依次循环,直至图像数据处理完毕。该平台的计数器 TIMER 模块可记录执行降晰处理程序所耗用的周期数(包括导入一次图像数据到 TC 和将处理结果写入 Frame Buffer 的时间)。

应用程序的 C 语言代码通过 C * CORE340 的交叉编译器得到二进制代码,利用虚拟 ELF 装载器加载到存储器中。启动编译后生成的仿真程序,虚拟的终端 TTY 显示程序运行结果。

3.2　处理结果

本文设计的 32 位自定义浮点格式的运算结果与使用 IEEE-754 双精度浮点格式的 PC 运算结果相比较,信噪比(SNR)和绝对误差(AE)在表 2 中列出。

表 2　图像信噪比和绝对误差

信噪比	绝对误差
89.471026	≤9.471

由表 2 可知,本文设计的数据格式能表示天文图像的数据范围,满足其精度要求。

为了验证本设计的性能,本文分别测试了本设计和 PC 所使用的计算时间,如表 3 所示。

表 3　计算时间

方案	图像大小	频率	时间(s)
PC	2K×2K	2.66 GHz	25
本设计	2K×2K	125 MHz	1.7

由表 3 可知,假设本设计可工作在 125 MHz 的时钟频率下,因此完成一帧的降晰处理需要 1.7 s,而 2.66 GHz 下的 PC 将需要 25 s,因此该设计的速度比 PC 提高了 14.7 倍。

4　结　语

本文针对天文图像处理中的空间变换核降晰算法,提出了一种采用传输触发架构的专用处理器设计,该设计在基于微处理器 C * CORE340 的电子系统级平台上验证,完成了数据密集型运算。本文设计的处理器架构不仅减少了芯片面积,而且速度比基于 PC 的纯软件实现提高了 14.7 倍,有效加快了运算速度,克服了普通计算机难以实现对天文图像观测数据实时处理的困难,提高了系统性能,并及时将有用信息传送到控制站,可有效应用在天文台嵌入式图像处理系统中以快速地完成数据密集型运算。

参　考　文　献

[1] Alard C. Image subtraction using a space-varying kernel[J]. Journal of Astronomy & Astrophysics Supplement, 2000, 144(1): 363 - 370.

[2] Alard C. ISIS: Image Subtraction Package[EB/OL]. (2007-05-11). http://www2.iap.fr/users/alard/package.html.

[3] 佟凤辉,樊晓桠,王党辉. 基于 SIMD 技术的图像卷积处理器体系结构研究[J]. 微电子学与计算机, 2003, 20(3): 13 - 16.

[4] Mohammad K, Agaian S. Efficient FPGA implementation of convolution [C]. IEEE International Conference on Systems, Man and Cybernetics, 2009: 3478 - 3483.

[5] Wong S, Jasiunas M, Kearney D. Fast 2d convolution using reconfigurable computing [C]. The Eighth International Symposium on Signal Processing and its Applications, Sydney, 2005: 791 - 794.

[6] Corporaal H. Microprocessor Architecture: From VLIW to TTA[M]. John Wiley & Sons, Chichester, 1997: 101 - 125.

[7] Israel H, Hessman F V, Schuh S. Optimising Optimal Image Subtraction [J]. Astronomische Nachrichten, 2007, 328(1): 16 - 24.

[8] SoCLib website [EB/OL]. https://www.soclib.fr/trac/dev/wiki.