计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(8):278−283 [doi: 10.15888/j.cnki.csa.006068]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

云计算主机内核虚拟化技术框架及其性能分析①

高　岩[1,2], 林　军[1,2], 云　龙[3], 李　磊[3]

[1](工业和信息化部电子第五研究所 软件质量工程研究中心, 广州 510610)
[2](工业和信息化部 基础软硬件性能与可靠性测评实验室, 广州 510610)
[3](华南理工大学 电子与信息学院, 广州 510640)

摘　要: 近些年来, 云计算已经成为了互联网领域的一个重要基础设施, 越来越多的应用被部署到云计算平台上提供在线或者离线的服务. 而虚拟化技术则是云计算的关键技术, 提供包括计算、存储和网络在内的资源, 一直是云计算技术研究领域的热点问题. 从内核虚拟化技术-KVM 出现, 成为了目前主机虚拟化的主流技术之一. 本文对 KVM 虚拟化技术的架构进行了详细地研究和分析. 详细介绍了 KVM 架构以及环境的构建过程, 对于 KVM 架构中包含的模块以及工作流程进行了详细深入的分析, 并对 KVM 的性能进行测试, 得到了有效的对比信息, 可作为相关研究和工程技术的参考, 具有较高的技术价值.

关键词: 云计算; 虚拟化技术; KVM; 性能对比

引用格式: 高岩,林军,云龙,李磊.云计算主机内核虚拟化技术框架及其性能分析.计算机系统应用,2017,26(8):278–283. http://www.c-s-a.org.cn/1003-3254/6068.html

Research and Performance Analysis of Cloud Computing Host Kernel Virtualization Technology Framework

GAO Yan[1,2], LIN Jun[1,2], YUN Long[3], LI Lei[3]

[1](Software Quality Engineering Research Center CEPREI, Guangzhou 510610, China)
[2](Key Laboratory for Performance and Reliability Testing of Foundational Software &Hardware, MIIT, Guangzhou 510610, China)
[3](School of Electronic and Information Engineering SCUT, Guangzhou 510640, China)

Abstract: Cloud computing has become an important infrastructure of the Internet, and more and more applications are deployed into cloud computing to provide the online service. Virtualization technology is the key element of cloud computing and hot research field, and it provides the computation, storage and network resource virtualization. KVM whose kernel is virtualization has become the one of the main virtualization technologies . This paper analyzes the structure of KVM, and introduces the working process of KVM. Finally, we also test the performance of KVM and get the useful results. These results, which are useful for technology improvement, can be used to improve the virtualization technology.

Key words: cloud computing; virtualization technology; KVM; performance comparison

虚拟化技术自从 20 世纪 60 年代诞生以来, 经过了几十年的发展, 现在已经广泛地运用在世界各个行业公司的服务器中, 尤其是在电信和金融领域, 甚至在个人电脑中也已能探寻到虚拟化技术的足迹. 虚拟化技术不受物理环境的约束, 将底层的硬件资源抽象给上层的系统软件使用, 并将有限的硬件资源抽象成多份供给不同的客户对象使用, 简单来说, 虚拟化技术是先集中资源在进行分配, 不仅能提高灵活性, 还能提高资源利用率. 最近几年, 伴随着 Intel 和 AMD 等处理器厂商在硬件层次上提供对虚拟化技术的支持, 以及云

① 基金项目: 广东省公益研究专项(2014A040401015)
收稿时间: 2016-11-23; 采用时间: 2017-03-23

计算技术的火热应用, 虚拟化技术得到了空前的发展, 而这不仅使得人们提高对计算机资源的使用率, 促进工作效率, 还极大地促进了各个行业的发展[1]. 然而, 国内外的一些专门研究和开发虚拟机项目的机构也才处于刚刚起步的状态, KVM 是基于虚拟化扩展(Intel VT 或 AMD-SVM)的 X86 硬件, 是 Linux 环境的开源全虚拟化解决方案[2]. 然而对 KVM 方面的研究少之又少, 但 KVM 虚拟化技术早已运用在许多大型项目中, 如 Google 在 2012 年 6 月发布了自己的“基础设施即服务”(IssS)模式的云计算平台——Google Compute Engine (GCE), GCE 的底层采用的是 KVM 虚拟化技术.

由此可见, 虚拟化技术尤其是 KVM 技术的迅猛发展, 得到业界广泛地重视. KVM 作为昂贵的专用虚拟化解决方案的企业级替代方案, 许多软件供应商都已开始着手 KVM 和基于 KVM 的解决方案, 这其中也包括国内的华为、中兴通讯等知名 IT 企业[3]. 然而能有效的使用 KVM 基础是构建 KVM 系统, 此外 KVM 性能也直接影响了服务应用的质量, 是该技术被进一步推广并提高相应服务质量的重要基础支撑.

综上所述, KVM 的部署方法和性能分析以及测试可以进一步提高目前虚拟化技术的使用, 部署的方式和测试结果也将进一步为相关技术的运用提供必要的参考和依据, 因此本文重点研究和分析 KVM 的架构, 不仅仅因为被集成到 Redhat 内核中, 更主要的是 KVM 本身的实用性, 它比 Xen 简化得多, 而且不需要对当前的 Linux 做任何修改, 也无需在更新版本重新编译内核. 同时 KVM 还拥有广大的技术发展前景, 通过本文的相关内容的介绍, 为虚拟化技术的应用和发展提供相关的技术依据和基础.

本文包括以下主要内容:

(1) 介绍 KVM 的架构和工作机理, 着重研究分析 KVM 虚拟化技术中的 CPU 虚拟化, 内存虚拟化以及 I/O 虚拟化等; (2)搭建 KVM 运行环境, 编译和安装 KVM 相关模块, 创建虚拟机, 进行相关的实验操作, 并对实验结果进行分析研究.

1 虚拟化技术架构介绍与分析

从计算机领域上来说, 虚拟化技术是先整合现有的物理资源, 包括物理 CPU、物理内存、I/O 设备、磁盘等, 再根据应用需求进行资源动态分配, 并让应用对象能够在各自获得的资源组合环境中正常运行.

从实现形式上来说, 虚拟化技术是在硬件层和操作系统中间(也可以是其他层次之间)插入一个虚拟化层[4], 即虚拟机监控器(Virtual Machine Monitor, VMM), 由 VMM 构成的传统虚拟化模型如图 1 所示. 虚拟机监控器运行在真实的物理平台之上, 并将下层的资源抽象给上层使用. 虚拟化层还可以把下层的资源抽象成多份供给多个客户对象使用, 可以按照特定的需求来打造许多互相隔离的虚拟化环境, 各个执行环境之间不互相影响, 整个系统具有很强的独立性[5].

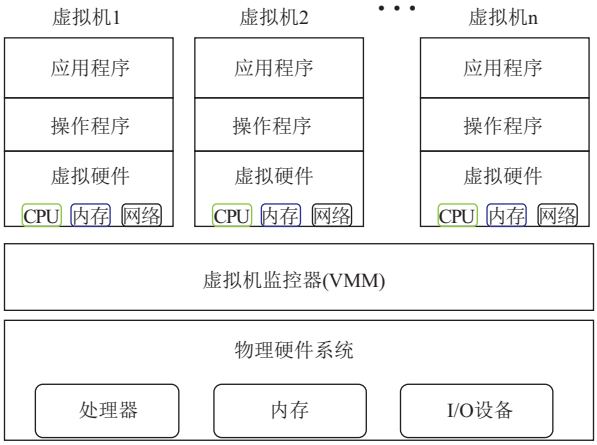

图 1 虚拟化技术架构示意图

从实现技术上来说, 虚拟化技术分为准虚拟化技术, 全虚拟化技术和硬件虚拟化. 硬件虚拟化又包括处理器虚拟化, 如 Intel 公司研发的 Intel VT-x(Intel Virtualization Technology for x86)技术以及 AMD 公司研发的 AMD SVM(AMD Secure Virtual Machine)技术; 内存虚拟化, 如 Intel EPT(Extended Page Table, 扩展页表)技术, AMD NPT(Nested Page Table, 嵌套页表)技术以及影子页表技术; I/O 虚拟化技术, 如 Intel VT-d(Intel Virtualization Technology for Direct I/O)技术和 AMD SVM 提供的 IOMMU(I/O Memory Management Unit)技术[6].

从 VMM 技术架构层面上来说, 又可分为 Hypervisor 模型, 宿主模型以及混合模型. 其中, Hypervisor 模型中的 VMM 处于操作系统的位置, 位于底层硬件之上, 也是一个功能完善的操作系统, 与传统操作系统相同的是, 底层的所有物理资源, 包括处理器, 内存和 I/O 设备都由 VMM 来管理和调度, 同时, VMM 还要向上提供能够运行客户机操作系统的虚拟机, 并管理虚拟环境, 如图 2 所示.

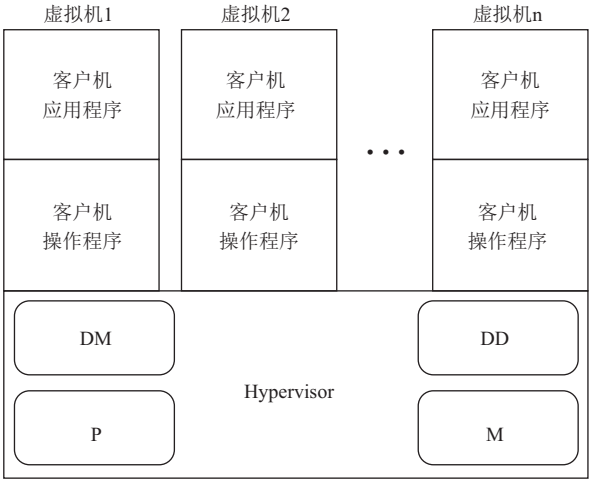

图 2 VMM 的 Hypervisor 技术架构示意图

DM(Device Model)表示设备模型, 专门负责 I/O 设备的虚拟化, DD(Device Driver)表示设备驱动, 负责 I/O 设备的管理, P(Processor)表示处理器管理代码, 负责物理处理器的管理和虚拟化, M(Memory)表示内存管理代码, 负责物理内存的管理和虚拟化, 由于 VMM 直接管所有的物理资源, 所以处理器管理代码, 内存管理代码, 设备模型以及设备驱动都是 VMM 的一部分.

在宿主模型中, 运行在物理硬件之上的仍然是传统的操作系统, 即宿主机操作系统, VMM 不再是连接硬件资源和虚拟机的间接层, 而成为了宿主机操作系统内核中的一个模块, 同时与宿主机操作系统共享底层的硬件资源. 与 Hypervisor 模型一样. 虚拟化功能的实现还是由 VMM 来负责, 它通过利用宿主机操作系统的 I/O 设备驱动和底层服务来获得资源, 以此来实现对处理器、内存和 I/O 设备的虚拟化, 而 VMM 创建出来的虚拟机通常作为宿主机操作系统的一个进程来参与调度. 宿主模型的架构如图 3 所示.

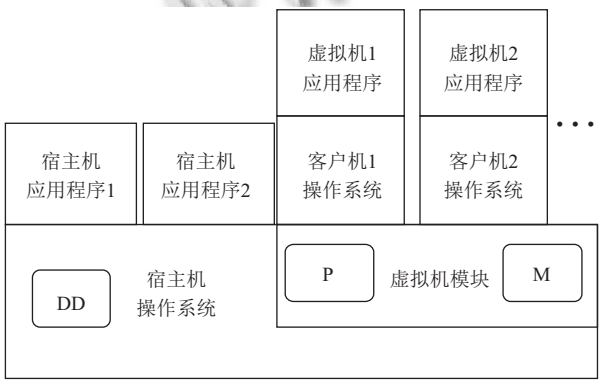

图 3 VMM 的宿主模型架构示意图

混合模型是上述两种模型的结合, Hypervisor 依旧运行在底层物理资源之上, 虚拟机的创建和管理也依然由它来负责. 但在它所创建的虚拟中, 有一个比较特殊, 叫特权操作系统, 相当于宿主模型中的宿主操作系统, Hypervisor 会把部分 I/O 设备的控制权交给它来处理, 以此来减少 Hypervisor 的负担, 所带来的好处是虚拟化的性能会得到很大的提高. 而原有的处理器虚拟化, 内存虚拟化还是由 Hypervisor 来实现, 但 I/O 设备虚拟化则是由 Hypervisor 和特权操作系统共同完成. 采用此模型的产品如 Xen 等.

2 KVM 架构分析

KVM 虚拟化技术采用宿主模型, 其架构没有 VMM 抽象层, 因此运行在物理硬件之上的仍然是传统的 Linux 操作系统, 即 KVM 的宿主操作系统. KVM 作为 Linux 内核中一个模块, 可以自由加载或卸载, 并提供了核心的虚拟化基础架构, 而 Linux 内核则成为了替代的 VMM. KVM 复用宿主操作系统 Linux 的进程调度、内存管理和 I/O 管理等功能, 由 KVM 创建的虚拟机只是 Linux 的一个进程, 与其他普通进程一样, 可以使用进程管理指令来对进程进行管理[7]. 虚拟机里运行的操作系统无需经过任何修改, 虚拟机之间互相独立, 只能独享各自拥有的虚拟资源, 如磁盘, 网卡等. 同时, 运行 KVM 的物理机器的处理器必须支持硬件虚拟化技术, 如 Intel 的 VT-x 或 AMD 的 AMD-V 技术[8]. KVM 架构如图 4所示.

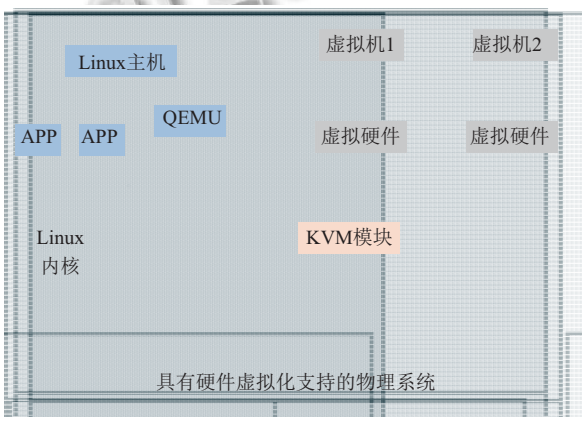

图 4 KVM 技术架构示意图

在 KVM 的架构中, Linux 内核是操作系统中最重要的组成部分, 运行在 CPU 的最高特权级别上, 它可以访问系统中的一切资源, Linux 内核的这种运行模式称为内核模式. Linux 系统中的应用程序(包括 QEMU)

运行模式为用户模式, 通常只运行在 CPU 最低的特权级别上. 操作系统中普通的 Linux 进程通常只有这两种运行模式, 即内核模式和用户模式, 而 KVM 架构增加了第三种模式: 客户模式, 图 4 中的虚拟机则是运行在此模式下, 虚拟机中的操作系统只执行非 I/O 客户机代码, 而与客户机相关的 I/O 任务则由运行在用户模式下的 QEMU 软件来执行[9].

在图 4 中, KVM 模块在操作系统运行时按需加载 Linux 内核空间运行, 成为 Linux 内核的一部分, KVM 模块通过创建字符设备文件/dev/kvm 来与用户空间中的 QEMU 通信, QEMU 则通过 KVM 提供的应用程序接口或 IOCTL 系统调用来掌握 KVM 模块进行的资源分配.

由于 KVM 采用宿主操作系统作为 VMM, 因此 Linux 内核也认为 KVM 是内核的一个驱动程序, 即/dev/kvm, 它可以启用客户模式. 虚拟机可以使自己的地址空间独立于内核或运行着的任何其他虚拟机的地址空间, 每个打开/dev/kv 的进程看到的是不同的映射, 实现了虚拟机隔离.

KVM 模块是 KVM 虚拟机中最重要的部分, 它的主要功能是先将底层硬件如 CPU 等初始化, 再将 CPU 的虚拟化模式打开, 然后将虚拟客户机置于虚拟机模式下运行, 并在虚拟客户机运行的时候提供一定的支持. KVM 模块主要分为三个部分, 即 kvm.ko、kvm-intel.ko 和 kvm-amd.ko, 其中, kvm.ko 模块是 KVM 的核心模块, 后面两个只是 KVM 的平台架构独立模块. 这三个模块的初始化流程如图 5 所示, 系统加电之后先经过 module_init 宏进行定义之后, 再根据硬件平台来选择对应的平台模块, 最后转到 kvm.ko 模块进行进一步的初始化.

结合用户模式下的 QEMU 软件, KVM 的工作流程如下: 在用户模式下, 虚拟机管理软件 QEMU 先开启一个虚拟机, 再通过 IOCTL 系统调用进入内核模式, 并向内核申请资源, 如是虚拟机创建好虚拟 CPU 和虚拟内存等, 创建好虚拟环境后, 向客户模式下的虚拟机发送指令启动客户机操作系统, 接着客户机操作系统向内核反馈相关的指令任务请求处理, 如果是 I/O 请求, 则通过/dev/kvm 设备交给 QEMU 软件处理, 如果是非 I/O 请求, 由宿主机操作系统内核来处理并将处理结果反馈回客户机操作系统. KVM 的执行过程如图 6 所示.

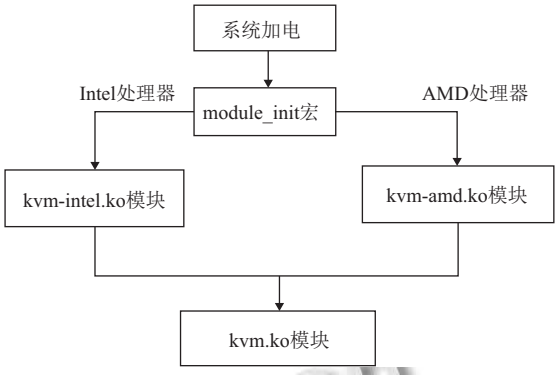

图 5 KVM 的初始化流程示意图

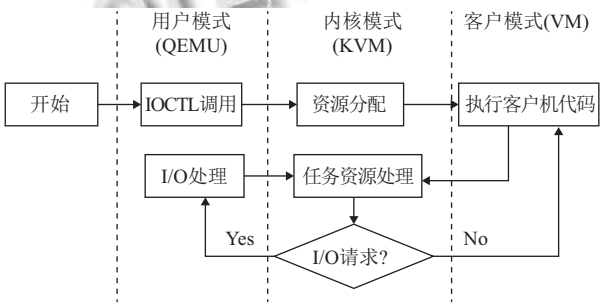

图 6 KVM 执行流程示意图

3 KVM 测试和运行环境的构建

在测试 KVM 性能前, 本文首先介绍 KVM 的构建方法, 为相关技术的使用和论文所属方法的重现提供必要的技术基础, 同时 KVM 构建的方法也是该技术运用的重要基础, 因此本文也做了详细的介绍.

本文选用了 Ubuntu 操作系统为基础, KVM 环境的构建主要步骤如下:

1) 配置物理硬件系统及安装宿主机操作系统

由于 KVM 只支持具有硬件虚拟化功能的系统, 所以在安装 KVM 虚拟机之前, 首先确认硬件系统支持硬件虚拟化, 再在 BIOS 上进行一定的设置, 让处理器打开虚拟化功能. 对于宿主操作系统, 目前常见的 Linux 操作系统, 包括 Redhat、Ubuntu 和 CentOS 等都支持 KVM 的安装. 本文采用 Ubuntu14.04 操作系统作为宿主操作系统. 在操作系统呢可通过如下指令确认硬件虚拟化功能的开启.

```
# grep -E 'vmx|svm' /proc/cpuinfo
```

其中 svm 为 AMD 处理器的虚拟化功能开启.

2) 编译安装 KVM 模块和 QEMU 软件

Linux2.6.20 之后的版本都在内核中加入了 KVM 模块, 因此安装并不需要在此编译 KVM 模块,

宿主系统只需要安装 qemu 和其他一些软件即可. 在 Ubuntu 系统中的安装指令如下:

#apt-get install qemu-kvm libvirt-bin virt-manager bridge-utils

其中, qemu-kvm 即是上述的 qemu 模块; libvirt-bin 是 Linux 下虚拟化工具的 C 函数库, 为虚拟化工具提供编程接口; virt-manager 是虚拟机的管理界面, bridge-utils 提供网络连接. 执行完以后, 接着输入以下命令:

kvm-ok

如果输出为以下信息:

INFO: /dev/kvm exists

KVM acceleration can be used

说明 KVM 已经可以正常使用.

3)KVM 虚拟机的创建:

可以通过指令来创建一个 KVM 虚拟机, 指令如下:

qemu-img create -r 2048 -f kvm.img -vcpus=2 --network bridge=br0 -c ubuntu14.04.3.iso

可按照一般的方法来进行虚拟机操作系统的安装.

4 KVM 的性能测试与对比

4.1 CPU 测试

本次测试通过分别在 KVM 虚拟机和宿主机上运行一段寻找指定范围内的质数的 Python 程序, 并比较在虚拟机和宿主机上花费的时间, 本次测试使用 Linux 的 time 工具来获取执行时间. Python 程序如下:

```
mport string
def find_prime(num):
  i=2
  while(i<num):
    if (0== num%i):
      return False
      break
    else:
      i=i+1
  return True
if__name__==’__main__’:
  input_num= [指定一个范围]
  j=0
  for j in range(1, input_num+1):
    if(find_prime(j)):
      print j
```

经过在宿主机和虚拟机指定不同范围的质数, 并运行三次后, 取得的平均结果如表 1 所示.

表 1 CPU 测试的数据结果

	寻找质数所花时间(s)		
	1-十万	1-1百万	1-1千万
宿主机	1.051	5.755	30.326
虚拟机	1.123	6.132	40.217

由以上结果可以看出, 在 1-1 百万之间 KVM 虚拟机和宿主机在寻找质数上所花费的时间差别不大, 可以认为是接近宿主机的性能的, 进一步说明了在计算负载低于一定范围时候, KVM 能有效的利用 CPU 的计算能力, 单次计算所需的时间较小, 操作系统调度颗粒度对计算时间影响较低, 此时中间层的性能损耗较低; 当计算负载交大时候, 单次计算所需的时间较长, 此时操作系统调度和中间件的资源切换会进一步影响计算的资源, 因此在计算负载较大时, 操作系统和中间件的切换会造成计算资源损耗的增加.

4.2 内存测试

本次内存测试使用内存测试工具 memtester 抓取虚拟机和宿主机不同大小的内存, 并 linux 自带工具 time 来获取时间, 从而进行对比, 跟 CPU 测试一样, 在宿主机和虚拟机分别进行三次抓取不同大小的内存, 在用三次时间的平均值来进行对比, 结果如表 2 所示.

表 2 内存测试的数据结果

	内存性能测试数据(s)		
	10 M	100 M	1000 M
宿主机	3.147	44.916	447.286
虚拟机	3.661	47.448	503.279

从表 2 中的数据可以看出, KVM 虚拟机的内存性能在程序占用内存较小时与宿主机相差无几, 主要是由于数据块较小, 操作系统在做内存映射时候, 能够快速找到数据的分配地址空间, 且数据传输的 IO 时间消耗也较低; 但在占用大容量内存时, KVM 虚拟机相对宿主机来说还是有一定的性能损耗, 主要是由于数据块较大, 操作系统需要多次计算数据存储的地址空间, 为数据的分配和映射做计算, 且数据的传输 IO 消耗较大. 因 KVM 在数据存储上, 性能表现和 CPU 基本一致, 具体而言由于 KVM 自身也是一种软件进程, 受操

作系统和中间件的影响, 当负载或者数据较小时, 所需的处理时间较低受系统调度的影响较小; 当负载或者数据增大时, 处理过程会受到系统调度和资源管理切换的影响, 进一步造成处理消耗.

4.3 I/O 测试

I/O 测试使用 Linux 自带的 dd 工具和 time 工具来进行检测, 分为读和写两部分, 而且每部分又分为 1 M, 10 M, 100 M 三部分. 每个小部分同上面一样进行三次, 最后取得平均值, 汇总如表 3 所示.

表 3 I/O 测试的数据结果

I/O性能测试数据(s)						
	读			写		
	1 M	10 M	100 M	1 M	10 M	100 M
宿主机	0.004	0.011	0.055	0.003	0.027	0.857
虚拟机	0.006	0.013	0.062	0.011	0.035	1.176

从表 3 中的数据可以看出, 在读方面, KVM 虚拟机在读小文件时性能与宿主机相当, 即使读 100 M 的数据, 也跟宿主机相差无几. 在写方面, KVM 虚拟机与宿主机相差较大, 文件越大, KVM 虚拟机写的时间越长, 这也是 KVM 虚拟机在 I/O 方面较弱的地方, 如果在硬件层面上使用 I/O 虚拟化技术, 情况会有所好转.

4.2 不同虚拟技术对比

在性能比较方面, 主要是和 KVM 有较大竞争关系的 Xen 来做比较, 通过不同的测试工具分别在 KVM 虚拟机和 Xen 虚拟机上进行测试, 再将测试结果进行对比从而得知两者在性能上的差异性, 测试结果如表 4 所示.

表 4 虚拟化与物理机的性能对比结果

	宿主机	KVM	Xen
C-Ray	73.25	80.52	107.65
POV-Ray	900	905	1328
Blowfish	714	710	492
DES	3475000	3383000	2397333
MD5	11124	11063	7628
OpenSSL	67.75	67.15	46.73
7-Zip	13331	11757	9771
PostMark	4166	3393	2409

表 4 中, C-Ray 为多线程测试, 数值越小越好. POV-Ray 为单线程图像渲染测试, 数值越小越好. Blowfish、DES、MD5、OpenSSL 皆为加密解密测试, 主要是测试 CPU 的运算性能, 数值越高越好. 7-Zip 为压缩解压测试, 主要测试磁盘的读写性能, 数值越高越好. PostMark为文件读写测试, 也是测试磁盘的读写性能, 数值越高越好.

从以上数据中可以看出, KVM 虚拟机各方面的性能均要优于 Xen 虚拟机, 主要是由于 KVM 采用驱动方式直接与内核进行数据交互和资源管理切换, 能进一步的提高资源的使用率, 减小系统切换和调度的损耗, 该特点使得越来越多的厂商开始关注和使用 KVM 作为虚拟化的技术.

5 结语

本文详细介绍了虚拟化技术的发展, 并分析了虚拟化技术的框架和技术发展, 重点对 KVM 的架构和执行过程进行分析和介绍. 为了进一步得有价值的信息, 我们做了大量的测试实验来测试 KVM 的性能, 并直观的给出了结果和分析. 通过本文的论述, 相关的研究工作者可以获取有价值的信息用于相关的研究, 同时也为虚拟化技术的研究发展提供了必要的实验信息和数据.

参考文献

1 任永杰, 单海涛. KVM 虚拟化技术: 实战与原理解析. 北京: 机械工业出版社, 2013.

2 崔泽永, 赵会群. 基于 KVM 的虚拟化研究及应用. 计算机技术与发展, 2011, 21(6): 108–111, 115.

3 张玉. KVM 崛起企业虚拟化的新选择. http:// server.51cto.com/sManage-369377.html. [2012-12-03].

4 李博, 李建欣, 胡春明, 等. 基于 VMM 层系统调用分析的软件完整性验证. 计算机研究与发展, 2011, 48(8): 1438–1446.

5 黄煜, 罗省贤. KVM 虚拟化技术中处理器隔离的实现. 计算机系统应用, 2012, 21(1): 179–182.

6 英特尔开源软件技术中心, 复旦大学并行处理研究所. 系统虚拟化—原理与实现. 北京: 清华大学出版社, 2009.

7 唐源, 李建平, 白雪, 等. 虚拟机监视器结构与实现技术. 计算机应用研究, 2009, 26(5): 1632–1635, 1649.

8 Uhlig R, Neiger G, Rodgers D, *et al*. Intel virtualization technology. Computer, 2005, 38(5): 48–56. [doi: 10.1109/MC.2005.163]

9 李永达. 虚拟机应用系统的设计与实现[硕士学位论文]. 西安: 西安电子科技大学, 2010.