doi:10.3772/j.issn.1002-0470.2013.09.005

基于 MIPS 架构的异构内存虚拟化方法研究①

蔡万伟②[*,**,***] 台运方[*,**,***] 刘 奇[*,**,****] 张晓辉[*,**] 张 戈[****,*****]

([*]中国科学院计算机系统结构重点实验室 北京 100190)
([**]中国科学院计算技术研究所 北京 100190)
([***]中国科学院研究生院 北京 100049)
([****]龙芯中科技术有限公司 北京 100190)
([*****]中国科学院重庆绿色智能技术研究院 重庆 401122)

摘 要 针对传统的同构内存虚拟化方法缺乏平台扩展性,在非 X86 处理器平台上性能较差的问题,研究了影响虚拟机内存性能的几个因素,并基于 MIPS 架构处理器提出了异构内存虚拟化方法,在不增加软件复杂度的前提下,提高了内存虚拟化性能。该方法基于对同构内存虚拟化的性能瓶颈的分析,通过修改虚拟机内存管理单元(MMU)降低软件维护开销;采用宿主机与客户机共享页表的方法提升访存的异常处理速度。该方法在龙芯 3 号处理器的系统虚拟机 KVM-LOONGSON 上得到实现。测试结果表明,该方法可以显著提升各类应用程序的性能,相比同构内存虚拟化方法,性能可以提升 50% 到 700%,达到本地执行性能的 71% ~97%。

关键词 系统虚拟化,内存虚拟化,KVM,MIPS,龙芯 3 号处理器

0 引 言

大型数据中心的崛起改变了传统意义上服务器的部署方法——从物理机的直接部署转变为虚拟机的间接部署,从而大大提高了服务器的利用率和管理效率。如何提高处理器的虚拟化性能是系统虚拟化研究的热点。内存虚拟化是系统虚拟化中最核心的部分,它的效率决定着虚拟化的整体性能。提高内存虚拟化的性能可以极大地提高虚拟机的整体性能。内存虚拟化的方法可分为全虚拟化、半虚拟化和硬件辅助虚拟化等方法。二进制翻译[1,2]和旁路转换缓冲(translation lookaside buffer,TLB)模拟[3]两种方法通过软件模拟处理器的内存管理单元(memory management unit,MMU)实现地址转换。IBM 提出的影子页表(shadow page table,SPT)实现了世界上第一个虚拟机环境 IBM System/370[4,5]。2002 年,剑桥大学发布了半虚拟化系统虚拟机 Xen,在使用影子页表的同时实现了内存半虚拟化方法[6]。由于没有特殊硬件的支持,纯软件实现的内存虚拟化在软件复杂度和性能上都差强人意[7-9]。2005 年后,Intel 和 AMD 争相推出硬件辅助虚拟化技术,其中扩展页表(extend page tables,EPT)[10,11]和嵌套页表(nested page tables,NPT)[12,13]则是两种类似的硬件辅助内存虚拟化方法。近年又出现了一些优化方法,如北京大学提出的动态内存半虚拟化方法[14]和选择性内存虚拟化方法[15]。在过去几年有很多针对 MIPS 架构的虚拟化研究,最具代表性的包括 Edouard 与 Scott 设计的 Disco[16]和 OK Labs 开发的 OKL4[17]。它们提出了许多关于 MIPS 的虚拟化方法,但是前者的内存性能较差,后者主要针对嵌入式领域,对数据中心服务器缺乏实用性。为此,中科院计算所通过 TLB 模拟实现了同构内存虚拟化(本文将客户机 MMU 与宿主机 MMU 保持一致的方法称为同构内存虚拟化方法,否则称为异构内存虚拟化方法),该方法能够实现虚拟机访存的正确性,但是性能表现一般。

本文通过对 MIPS 同构内存虚拟化的热点路径

① 国家"核高基"科技重大专项课题(2009ZX01028-002-003,2009ZX01029-001-003),国家自然科学基金(60921002,61003064,61050002,61070025,61100163,61133004,61173001)和计算机体系结构国家重点实验室基金课题(ICT-ARCH201002)资助项目。
② 男,1984 年生,博士生;研究方向:计算机系统结构,操作系统和虚拟化;联系人,E-mail:caiwanwei@ict.ac.cn
(收稿日期:2012-09-07)

进行分析,发现 MIPS 的操作系统对 TLB 异常有三条处理路径,其中 TLB 缺失异常处理是发生次数最多、最影响性能的关键过程。为了提升该过程的运行效率,本文提出了异构内存虚拟化方法,通过修改虚拟机的 MMU 结构,对虚拟机操作系统隐藏 TLB,降低软件维护开销;并采用宿主机与客户机共享页表的方法,显著提高了 TLB 异常处理的效率。为了验证这些方法,本文基于 MIPS64 兼容处理器龙芯 3 号的系统虚拟机 KVM-LOONGSON 实现了异构内存虚拟化方法,并进行了性能测试。实验结果表明,本文提出的异构内存虚拟化方法相对于同构方法的性能最多可提升 7 倍,而且大多数测试程序能够达到本地执行 90% 以上的性能。

1 MIPS 内存虚拟化热点路径分析

为了降低软件的开发难度,以往的研究都采用同构内存虚拟化方法,即客户机与宿主机采用相同的内存管理机制。本节介绍基于 MIPS 的同构内存虚拟化原理,并分析其性能瓶颈。

1.1 MIPS 的 MMU 工作原理

内存管理主要负责虚拟内存和物理内存的分配和释放,并通过硬件实现虚拟地址到物理地址的转换。目前流行的硬件平台主要采用两种内存管理模式:结构化页表和结构化 TLB。MIPS 的内存管理采用后一种模式,通过软硬件协同的方式实现 TLB 的维护。该模式中的硬件部分称为 MMU,主要由 TLB 和控制 TLB 读写的寄存器组成。可见,TLB 对 MIPS 的操作系统是可见的,需要软件进行维护。

MIPS 的 MMU 工作原理主要是使用特权指令维护 TLB 映射,当虚拟地址通过 TLB 对物理内存进行访问时,通常有 3 种结果:TLB 命中,虚拟地址转化为物理地址,访存成功;TLB 缺失,进入 TLB 缺失处理过程,访问相应的页表,对 TLB 进行重填;TLB 读写异常,进入 TLB 读写异常处理函数,修改页表对应项的读写权限或者分配物理页,并对 TLB 进行重填。

1.2 同构内存虚拟化方法的原理

已有的内存虚拟化采用的都是同构的方法,即客户机和宿主机使用相同的 MMU 硬件结构和内存管理机制。当前 MIPS 采用的同构内存虚拟化方法是模拟 TLB 方法,该方法通过软件为客户机模拟整套 MMU 硬件支持,包括 TLB。

该方法定义了三种 TLB:Guest TLB、Shadow TLB 和 Host TLB。其中,Guest TLB 由客户机操作系统(guest operating system,Guest OS)维护,用于保存客户机虚拟地址(guest vitual address,GVA)到客户机物理地址(guest physical address,GPA)的映射。该映射不会被 Guest OS 直接使用,而是用于生成 Shadow TLB。Guest OS 在维护 Guest TLB 时,对 TLB 的读写指令会被虚拟机管理器(virtual machine monitor,VMM)捕捉并模拟,使 Guest OS 认为成功读写了硬件 TLB。Shadow TLB 由 VMM 维护,用于保存 GVA 到宿主机物理地址(host physical address,HPA)的映射。Host TLB 由宿主机操作系统(host operating system,Host OS)维护,用于保存宿主机虚拟地址(host virtual address,HVA)到 HPA 的映射,是唯一真正的硬件 TLB,客户机最终是通过 Host TLB 装载 Shadow TLB 的内容实现 GVA 到 HPA 的转换。

根据 1.1 节,Guest OS 通过 TLB 访存会出现 3 种结果,VMM 对这 3 种结果的处理过程分别进行模拟。VMM 运行于 Host OS 的内核态,通过对宿主机物理内存的管理,实现了 GVA 到 HPA 的转换:当 Guest OS 发生 Shadow TLB 缺失时,陷入到 VMM,进行 Shadow TLB 缺失处理过程,该过程首先遍历 Guest TLB 查找对应项,如果命中,则将对应项中的 GPA 转化为 HPA,然后填入 Shadow TLB,并返回 Guest OS 继续运行;如果不命中,则向 Guest OS 注入 TLB 缺失异常,由 Guest OS 重填 Guest TLB,而 VMM 通过捕捉 Guest OS 的 TLB 指令实现对 Guest TLB 读写的模拟,当 Guest TLB 重填成功后,返回 Guest OS 发生 TLB 缺失的现场重新执行,最终实现 Guest OS 访存的 TLB 命中。

1.3 同构内存虚拟化瓶颈分析

同构内存虚拟化方法成功将 GVA 转化为 HPA,实现了 Guest OS 的访存功能。但是通过软件模拟 TLB 会带来很大的性能开销。如表 1 所示,虚拟机运行时 TLB 缺失发生的次数占所有虚拟机退出次数的 87.62%,是决定虚拟机性能的热点路径。同构内存虚拟化方法为了软件模拟 TLB,其繁琐复杂的 TLB 缺失处理过程极大地降低了虚拟机的整体性能。

2 基于 MIPS 的异构内存虚拟化方法

根据上一节对 MIPS 同构内存虚拟化方法的分析可见:系统虚拟机为了降低软件的开发难度需要

表 1 虚拟机异常退出次数

异常类型	异常数量(次)
中断	1355
TLB 修改	1
TLB 读	48
TLB 写	12
TLB 缺失	200228
系统调用	1018
保留指令	44
特权指令	25823
溢出	1
总计	228530

保持客户机与宿主机同构,采用结构化 TLB,因此虚拟出一套与宿主机相同的 MMU 结构,并进行维护,从而导致性能低下。本文提出的异构内存虚拟化方法,通过对 Guest OS 隐藏 MMU 硬件结构,实现结构化页表,消除对 Guest TLB 的维护开销,加速了 TLB 缺失的处理过程,显著提高了 Guest OS 的访存能力。

2.1 异构内存虚拟化地址转换原理

Guest OS 的 TLB 缺失处理过程是内存虚拟化的热点路径,决定着虚拟机的整体性能。本节提出的异构内存虚拟化方法,对 Guest OS 和 VMM 的内存管理模块进行了重新设计,实现了高速的地址转换。

如图 1 所示,异构内存虚拟化方法只需要维护一个 TLB,即物理 TLB。当 Guest OS 发生 TLB 缺失异常时,将页表中保存的 GVA→HPA 映射填入物理 TLB,就能实现 GVA 到 HPA 的转换。可见,该方法

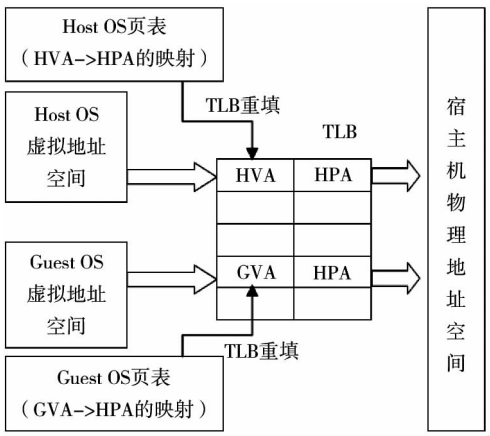

图 1 异构内存虚拟化的地址转换

通过客户机和宿主机共享物理 TLB 实现地址转换功能。

其中,MIPS 的 GVA 分为 3 类:内核固定地址、内核可映射地址和用户进程地址。为了实现上述过程,异构内存虚拟化方法需要对 3 类 GVA 访存导致的两种虚拟机退出分别进行处理,这两种处理过程分别实现了 Guest TLB 的快速重填以及 3 类 GVA 地址页表的维护。

2.2 Guest TLB 快速重填

如图 2 所示,当 Guest OS 发生 TLB 缺失异常,VMM 根据 GVA 的类别对内核固定地址页表、内核可映射地址页表或者当前用户进程页表进行索引,查找出对应的页表项,填入物理 TLB,用于下一次 GVA 访存映射。由于这个过程是 VMM 负责,并且 3 种页表通过内存共享可以被 VMM 直接访问,因此查询页表和重填 TLB 表项的过程可以精简到 50 条指令,使作为热点路径的 TLB 缺失异常处理过程能够非常高效地完成。

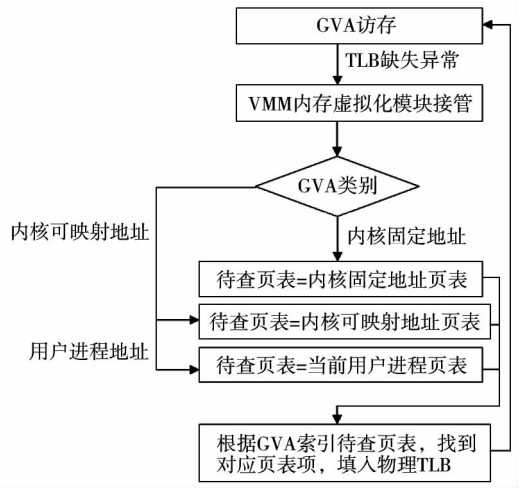

图 2 异构内存虚拟化 TLB 缺失处理流程

2.3 三类 GVA 地址页表的维护

如图 3,当 Guest OS 发生 TLB 读写异常时,VMM 对发生异常的 GVA 进行分类处理,最终将 GVA→HPA 映射填入对应的页表:如果 GVA 是内核固定地址,将直接进行 GVA→GPA,然后通过内存共享方法实现 GPA→HVA→HPA 的转换;如果 GVA 是内核可映射地址或者当前用户进程地址,VMM 将向 Guest OS 注入 TLB 读写异常,即缺页异常,由 Guest OS 对相应页表进行维护,填入对应的 GVA→HPA 映射。

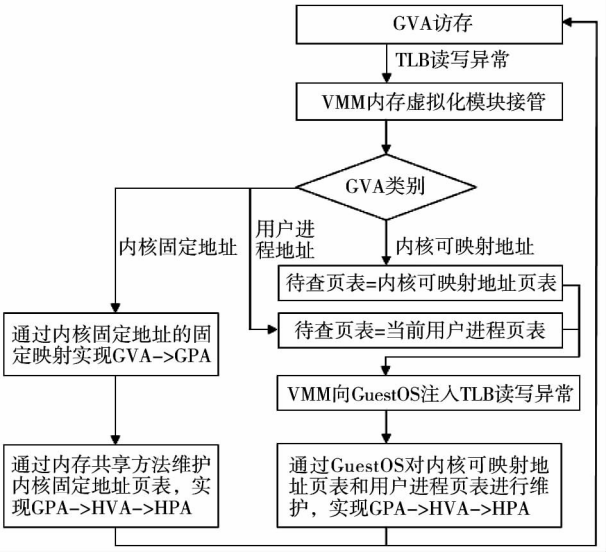

图 3　异构内存虚拟化 TLB 读写异常处理流程

2.4　页表共享

根据上两节所述,Guest OS 中的页表由 Guest OS 维护,并提供给 VMM 在 TLB 缺失异常处理过程中访问,因此需要实现 Guest OS 和 VMM 之间的页表共享。

本文中的页表共享是通过内存共享实现的。由于 GVA→GPA 映射和 HVA→HPA 的映射分别由 Guest OS 和 Host OS 维护,所以 VMM 只需要维护 GPA→HVA 的映射关系,就可以实现 Host OS 与 Guest OS 之间的内存共享。由于客户机是 Host OS 的用户进程,所以当客户机启动时,会分配一定数量的 HVA 地址作为客户机的内存,同时建立 HVA 与 GPA 的映射表。通过该映射表,Guest OS 可以随时根据 GVA 生成对应的 HVA。在具体实现时,为了便于下次快速查询地址映射,Guest OS 还可以将生成的 GVA→HVA 和 HVA→GVA 映射存入哈希表 G2H 和 H2G。通过该方法,VMM 中的 HVA 可以与 Guest OS 中对应的 GVA 访问同一内存单元,实现内存共享。

异构内存虚拟化在内存共享的基础上实现了页表共享。现代操作系统一般采用 2 级分页机制,页表包括页目录项和页表项两级。Guest OS 对页表的访问主要有 4 种:页目录项读、写以及页表项读、写。由于 Guest OS 的页表需要与 VMM 共享,使 VMM 能够高效地实现 TLB 缺失异常处理,因此页目录项和页表项需要分别填充 HVA 和 HPA。但是 Guest OS 在维护页表时只能识别 GVA 和 GPA 的地址空间,因此在读页目录项和页表项时需要做 HVA→GVA 和 HPA→GPA 的转换;相反,在写页目录项和页表项时需要做 GVA→HVA 和 GPA→HPA 的转换。

综上,通过异构内存虚拟化方法,VMM 对 Guest OS 成功隐藏了 TLB,实现了高效的 TLB 缺失异常处理。

3　实验平台和性能分析

3.1　实验平台

根据上述理论,本文基于龙芯 3A 多核处理器的虚拟化系统 KVM-LOONGSON 实现了异构内存虚拟化方法,并对其进行了功能测试和性能分析。其中,龙芯 3A 多核超标量处理器是 MIPS64 兼容处理器,主频设为 800MHz,内存频率设为 300MHz,搭载 AMD SB700 和 RS780E 南北桥组成测试开发平台;KVM-LOONGSON 是中科院计算所微处理器中心基于龙芯 3A 处理器开发的系统虚拟机,采用的是当前流行的 KVM 虚拟机框架,分别实现了处理器虚拟化模块、内存虚拟化模块和 IO 设备虚拟化模块。

3.2　实验结果

本文采用测试程序集为 STREAM 和 SPEC CINT2000,分别针对虚拟化系统的访存性能和整体性能进行评测。其中,STREAM 是业界广为流行的综合性内存带宽实际性能测量工具之一,是用来全面测试高性能计算系统的测试套件的一部分;SPEC CINT2000 是 SPEC(标准性能评测组织)开发的专门用于评价处理器性能的一套基准程序,在性能评价领域具有很高的权威性。

表 2 是 STREAM 内存测试结果:针对 4 种不同的访存行为,异构内存虚拟化相对于同构虚拟化内存带宽提升了 42.6% ~51.6%,可以达到物理机内存带宽的 95.9% ~97.3%。可见,对于访存密集的程序,异构虚拟化可以达到与本地执行近乎相同的性能。

表 2　三种系统的 STEAM 访存带宽比较

访存	同构虚拟化 (MB/s)	异构虚拟化 (MB/s)	物理机 (MB/s)
Copy	186.09	272.69	280.35
Scale	189.05	269.63	278.99
Add	250.91	380.51	391.13
Triad	235.42	348.05	362.88

SPEC CINT2000 性能测试结果见图 4。由图可见,异构内存虚拟化使虚拟机的整体性能得到显著提升,大部分测试程序达到了原来性能的 3 ~ 7 倍。其中,访存密集型测试程序 181. mcf、253. perlbmk 和 255. vortex 相对原来的性能分别提高了 76. 9 倍、10. 2 倍和 71. 3 倍;而计算密集型和 I/O 密集型测试程序 164. gzip、186. crafty、252. eon 和 300. twolf 相对于原来的性能只提高了 15. 6% 、5. 9% 、26. 7% 和 9. 1% 。同时,异构内存虚拟化可以达到本地执行性能的 71. 4% ~ 97. 4% 。

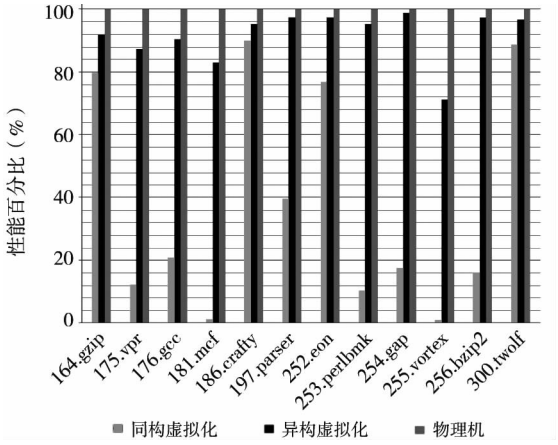

图 4 同构内存虚拟化、异构内存虚拟化以及物理机的 SPEC CINT2000 性能比较

3.3 性能分析

根据测试结果可以看到,异构内存虚拟化大大提高了系统虚拟机的访存速度,在一些应用上能够接近本地执行的性能。

根据测试结果可以将测试程序分为 4 类:第一类是异构内存虚拟化相对同构内存虚拟化性能提升显著的,如 stream、181. mcf、253. perlbmk 和 255. vortex,这类程序的特点是访存行为复杂或者系统调用较多,导致用户态和内核态的 TLB 缺失总次数较多,异构内存虚拟化方法可以大大降低 TLB 缺失后的开销,从而显著提升性能;第二类是异构内存虚拟化相对同构内存虚拟化性能提升不明显的,如 164. gzip、186. crafty、252. eon 和 300. twolf,这类程序主要进行对内存需求较小的科学计算,同构虚拟化已经达到了较好的性能;第三类是异构内存虚拟化的性能接近本地执行的,如 stream、197. parser、252. eon、254. gap 和 256. bzip2,这些程序进行计算的数据主要存放在内存中,而测试环境中虚拟机独占整个 TLB,因此可以接近本地执行的性能;最后一类程序是异构内存虚拟化的性能与本地执行相差较大,如 175. vpr、181. mcf 和 255. vortex,这类程序运行时需要的内存较多,导致最终使用了宿主机或者客户机的 swap 分区,另外此类程序还有 IO 访问密集的特点,由此可见内存虚拟化在 IO 缓存机制方面仍然需要做深入研究。

4 结 论

本文研究了 MIPS 架构处理器的内存管理特性,详细分析了同构内存虚拟化的热点路径和性能瓶颈,并提出异构内存虚拟化方法,通过改变虚拟机的 MMU 结构,最终在 MIPS 架构处理器的虚拟化系统 KVM-LOONGSON 上实现了高效的内存虚拟化机制。基于龙芯 3A 平台,本文验证了 MIPS 异构内存虚拟化的正确性和有效性。实验结果表明,基于本文方法实现的 MIPS 异构虚拟化系统能够得到较好的访存性能和整体性能,相对同构内存虚拟化方法可以提升 0. 5 倍到 7 倍的性能,达到本地执行性能的 71% ~ 97% 。

另外,架构分析和实验数据表明 MIPS 异构虚拟化系统在运行内存缓存行为复杂的程序时存在性能瓶颈,虚拟内存管理机制和硬件辅助 TLB 模拟均有值得研究的地方,通过这些研究将更好地开发出 MIPS 架构处理器的内存虚拟化特性。

参考文献

[1] Hu W W, Liu Q, Wang J, et al. Efficient binary translation system with low hardware cost in Computer Design. In: Proceedings of the 27th International Conference on Compute Design, California, USA, 2009. 305-312

[2] Bellard F. Qemu, a fast and portable dynamic translator. In: Proceedings of the USENIX 2005 Annual Technical Conference, California, USA, 2005. 41-46

[3] Kivity A, Kamay Y, Laor D, et al. KVM: the linux virtual machine monitor. In: Proceedings of the Linux Symposium 2007, Ottawa, Canada, 2007. 255-230

[4] Creasy R J. The origin of the vm/370 time-sharing system. *IBM Journal of Research and Development*, 1981, 25: 483-490

[5] Alkassar E, Cohen E, Hillebrand M, et al. Verifying Shadow Page Table Algorithms. In: Proceedings of the 2010 Conference on Formal Methods in Computer-Aided Design, Texas, USA, 2010. 267-270

[6] Barham P, Dragovic B, Fraser K, et al. Xen and the art of virtualization. *SIGOPS Operating System Review*, 2003, 37:

164-177

[7] Agesen O, Garthwaite A, Sheldon J, et al. The evolution of an x86 virtual machine monitor. *SIGOPS Operating System Review*, 2010, 44: 3-18

[8] Adams K, Agesen O. A comparison of software and hardware techniques for x86 virtualization. *ACM SIGARCH Computer Architecture News*, 2006, 34: 2-13

[9] Doorn L. Hardware virtualization trends. In: Proceedings of the 2nd international Conference on Virtualization Execution, New York, USA, 2006. 45-45

[10] Uhlig R, Neiger G, Rodgers D, et al. Intel virtualization technology. *Computer*, 2005, 38(5): 48-56

[11] Vmware. Performance evaluation of intel ept hardware assist. http://www.vmware.com/pdf/Perf_ESX_Intel-EPT-eval.pdf: Vmware, 2009

[12] Vmware. Performance evaluation of amd rvi hardware assist. http://www.vmware.com/pdf/RVI_performance.pdf: Vmware, 2009

[13] Advanced Micro Devices. Amd-v nested paging revision: 1.0. http://developer.amd.com/assets/NPT-WP-1%201-final-TM.pdf: Advanced Micro Devices, 2008

[14] 汪小林,孙逸峰,罗英伟等. 面向操作系统透明的动态内存半虚拟化技术. 中国科学:信息科学, 2010, 53(5): 692-705

[15] 罗英伟,汪小林,肖臻等. 单计算系统资源虚拟化方法. 中国计算机学会通讯, 2011, 7(9): 13-26

[16] B. Edouard, D. Scott. Disco: running commodity operating systems on scalable multiprocessors. *ACM Transactions on Computer Systems*, 1997, 15(4): 412-447

[17] Wikipedia. http://en.wikipedia.org/wiki/Virtualization: Wikipedia, 2012

Heterogeneous Memory Virtualization on the MIPS Architecture

Cai Wanwei[*] [**] [***], Tai Yunfang[*] [**] [***], Liu Qi[*] [**] [****], Zhang Xiaohui[*] [**], Zhang Ge[****] [*****]

([*] Key Laboratory of Computer System and Architecture, Chinese Academy of Sciences, Beijing 100190)

([**] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([***] Graduate University of Chinese Academy of Sciences, Beijing 100049)

([****] Loongson Technology Corporation Limited, Beijing 100190)

([*****] ChongQing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 401122)

Abstract

In consideration of the problem that the traditional isomorphic memory virtualization (IMV) method lacks platform scalability and performs poorly on the platforms of non X86 processors, the factors influencing the performance of virtual memories were investigated, and a heterogeneous memory virtualization (HMV) method based on the MIPS architecture, was presented to improve the performance of memories' virtualization while without increasing the software complerity. The HMV method can reduce the software maintenance overhead by modifying the structure of the virtual machine's memory management unit, and enhance the speed of memory access exception handling by sharing page tables between host and guest OS. This model was implemented on the Loongson3 processor virtual machine KVM-LOONGSON. The test results show that the method can greatly enhance the performance of the various types of applications. Comparing to the IMV method, the performance of the HMV method can increase from 50% to 700%, and approache 71% to 97% of the performance of the native implementation.

Keywords: system virtualization, memory virtualization, KVM, MIPS, Loongson3