DOI:10.19492/j.cnki.1672-0946.2012.04.019

内存管理单元 MMU 虚拟化代价研究

王华斌[1],夏清泉[2],李希然[2],赵胜义[2]

(1. 哈尔滨医科大学附属 第一临床医学院,哈尔滨 150001;2. 哈尔滨工业大学 计算机科学与技术学院,150001)

摘 要: 虚拟化技术在计算系统中的作用日益重要. 越来越多处理器内集成了 MMU 内存管理单元为内存虚拟化提供支持. 基于 SPARC 平台,对 hypervisor MMU 虚拟化机制进行深入探讨,并对虚拟化代价进行分析: MMU 虚拟化代价主要来自于 Guest OS 对 MMU 操作时的 hypercall 调用以及访存异常的处理代价. 实验表明: data_real_translation_miss,IMMU_miss_HWTW 和 mmu_ map_addr 等 hypervisor API 和异常处理成为 MMU 虚拟化代价的重要因素,MMU 支持下 hypervisor 虚拟化引入了很小的性能开销.

关键词: 虚拟化; MMU 内存管理单元; SPARC T2; hypervisor

中图分类号: TP 391 **文献标识码:** A **文章编号:** 1672 - 0946(2012)04 - 0447 - 04

MMU virtualization overhead analysis for SPARC processor

WANG Hua-bin[1], XIA Qing-quan[2], LI Xi-ran[2], ZHAO Sheng-yi[2]

(1. The First Clinical Medical College, Harbin Medical University, Harbin 150001, China;
2. School of Computer Science, Harbin Institute of Technology, Harbin 150001, China)

Abstract: With the rapid development of computer technology, virtualization is playing an important role in computer system, wherein memory virtualization is an essential aspect. Recent advance in hardware technology enables MMU(Memory Management Unit) integrated in processor, which is of significance to lower the overhead of memory virtualization. In this paper, MMU virtualization mechanism in terms of hypervisor for SPARC T2 multi - core was investigated in great details. In - depth analysis reveals that hypervisor APIs and TLB missed related exceptions become the important overhead factors. Experiment results demonstrated that data_real_translation_miss, IMMU_miss_HWTW and mmu_ map_addr become the primary sources of overhead for MMU virtualization, confirming that MMU virtualization incurs negligible overhead.

Key words: virtualization; MMU(memory management unit); SPARC T2; hypervisor

随着计算技术的高速发展,虚拟化在云计算等关键应用中扮演日益重要的角色,内存虚拟化成为虚拟化技术的重要研究内容之一. 目前越来越多处理器内集成了内存管理单元 MMU(Memory Management Unit),为虚拟化提供支持. MMU 虚拟化成为内存虚拟化至关重要的一个方面,对降低虚拟化代价提高性能至关重要. 本文对 SPARC T2 处理器 MMU 虚拟化机制进行深入研究,并对虚拟化引入的代价进行分析与探讨.

1 相关背景

IBM 公司于 1965 年开发了第一台虚拟机 System/360 Model 40 VM,开创了虚拟化技术的先河[1]. 随着工艺的进步,计算机步入多核时代,虚

收稿日期: 2012 - 02 - 17.

作者简介: 王华斌(1983 -),男,助理工程师,研究方向: 计算机系统结构.

拟化技术在云计算等关键任务中扮演着日益重要的角色,以达到计算资源的高效、合理利用,提高系统可信性(dependability)[2].

按实现方式系统级虚拟化可以分为:全虚拟化(Full)、泛虚拟化(Para)和硬件辅助虚拟化(Hardware-assisted)[3].其中,内存虚拟化是虚拟化中最为关键的任务之一,虚拟化后的访存效率直接决定了系统性能.

各类不同虚拟化技术在内存虚拟化方面都有着典型实现:全虚拟化方式其内存虚拟化通常采用影子页表(Shadow Page Table)技术,泛虚拟化通常基于MMU内存管理单元实现内存的虚拟化,硬件辅助虚拟化则根据不同架构的处理器有着不同的实现.如:Intel VT-x通过扩展页表EPT(Extended Page Table)实现内存虚拟化的支持.

本文对内存虚拟化中至关重要的MMU虚拟化机制进行深入探讨,详细分析了hypervisor对MMU虚拟化的支持,并对虚拟化代价进行评测.

2 MMU相关机制

2.1 逻辑域

Sun公司UltraSPARC虚拟化技术被称为逻辑域(Logical Domain).Sun公司通过在物理硬件之上增加hypervisor虚拟化层来实现底层硬件的抽象,被抽象出的guest OS虚拟机被称为逻辑域.系统中的物理资源由hypervisor全权掌控并实现虚拟化,包括:CPU资源、内存资源、硬盘、网络,以及I/O资源等.逻辑域拥有各自的虚拟资源,独立运行且互不干扰.逻辑域拥有三种执行模式:NP非特权模式(Non-Privileged mode)、P特权模式(Privileged mode)和HP超特权模式(Hyper-Privileged mode)[4].

MMU是CPU内处理访存请求的硬件单元,完成内存管理与保护.传统MMU只支持VA虚拟地址到PA物理地址的转换与保护.为对虚拟化提供支持,Sun对SPARC处理器中MMU的地址转换功能进行扩充,使其支持VA虚拟地址、RA实地址和PA物理地址三种地址.

虚拟地址(Virtual Address,VA),指Guest OS提供给其应用程序使用的线性地址空间.实地址(Real Address,RA),经抽象的虚拟机Guest OS看到的伪物理地址.机器地址(Physical Address,PA),真实的机器地址,即地址总线上出现的地址信号.

逻辑域技术中三地址之间关系如下:

RA = Context_ID + VA

PA = Partition_ID + RA = Partition_ID + Context_ID + VA

Sun逻辑域技术中内存管理由hypervisor和Guest OS各司其责地共同完成.例如hypervisor维护实地址RA到物理地址PA的映射关系,而VA虚地址到RA实地址的转换由guest OS的页式内存管理机制完成,即:

Guest OS: RA = f(VA)

hypervisor: PA = g(RA).

由此,Sun逻辑域技术中hypervisor对内存的虚拟化主要是对MMU的虚拟化.

2.2 地址转换机制

SPARC T2处理器内MMU包括两个TLB(Translation Lookaside Buffer),一个是指令i-TLB,一个为数据d-TLB.TLB中包含若干TTE(Translation Table Entry),每一项TTE保存了页的映射信息.

TLB内TTE只维护VA虚拟地址到PA物理地址和RA实地址到PA物理地址的转换信息,VA虚拟地址到RA实地址的转换信息不能写入TLB中.

作为页表的高速缓存的TLB其大小是有限的,VA到PA、RA到PA的映射不能全部写入其中.因此MMU内还维护了如下两个结构STB(Software Translation Table)和TSB(Translation Storage Buffer).

STB就是传统意义上的页表,存储由Guest OS所维护的VA到RA转换.STB中保存了它所属的guest OS的所有VA到RA的映射信息,相对于TSB来讲STB存储代价更大,访问代价更高.

TSB实际上就是STB的缓存,由Guest OS维护,它可以看作硬件与STB之间的接口.

STB和TSB联合使用,当TLB缺失时,SPARC T2内MMU通过硬件访问TSB查找相应的地址信息并填充TLB.

STB、TSB和TLB及其地址转换关系[5],如图1所示.

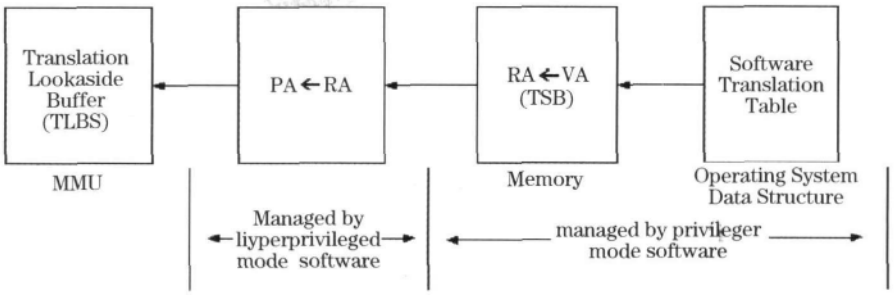

图 1　SPARC 地址转换机制

STB 和 TSB 均由 Guest OS 维护,STB 保存全部的 VA 到 RA 地址映射关系. 作为 STB 的缓存,TSB 保存最常用的 VA 到 RA 转换信息.

RA 到 PA 的映射由 HP 超特权模式下的 Hypervisor 维护,映射结果直接写入 TLB. 但 VA 到 PA 映射则需通过 Hypervisor 将 VA 间接转换到 PA 后再写入 TLB. 其中 VA 到 RA 的映射不会存放到 TLB 中.

实际上,TLB 是以 VA、Partition ID 和 Context ID,或 RA 和 Partition ID 为输入,当 TLB 命中,返回 PA 和相应页属性. 当 TLB 缺失时,则导致 TSB 访问; 如果 TSB 发生访问缺失,则触发 STB 访问. 处理器内 MMU 对 TSB 访问提供硬件支持.

3　MMU hypervisor 虚拟化机制

3.1　MMU Hypervisor API

由于 MMU 要被多个 guest OS 所共用,因此不允许客户机操作系统直接操作 MMU,只有运行在 HP 超特权模式下的 Hypervisor 软件才有对 MMU 直接访问的权限. Hypervisor 将对 MMU 的操作抽象成 hypervisor API 供客户机 OS 调用完成相应 MMU 的操作. MMU 相关的重要 Hypervisor API 如表 1 所示[5].

表 1　MMU 相关重要 Hypevisor API

API 名称	描述
mmu_enable	打开 mmu
mmu_disable	关闭 mmu
mmu_fault_area_conf	配置 mmu 错误信息区域
mmu_demap_page	将 TLB 内的某一页的映射信息无效
mmu_demap_all	置 TLB 内的所有 TTE 无效
mmu_map_addr	将某地址映射信息填入 TLB
mmu_unmap_perm_addr	撤销 Hypervisor 内的固定映射信息

3.2　TLB 缺失处理

MMU 辅助下的 TLB 缺失处理成为 hypervisor 虚拟化机制的重要内容,也是引入虚拟化代价的重要瓶颈. 根据 MMU 设置的不同,对于 TLB 缺失的处理有以下两种方式:

1) 在 MMU 打开 Hardware Tablewalk(HWTW) 支持的情况下,由 HWTW 硬件访问 TSB,在 TSB 中查找相应的映射信息进而填充 TLB;

2) TLB 缺失的处理也可以完全由软件实现,通过配置 MMU,可以使 HWTW 无效;

HWTW 根据 TSB 配置寄存器的信息完成 TSB 的正确访问,具体过程如下:

1) 根据 TSB 配置寄存器和虚拟地址计算 TSB 内可能 TTE 的物理地址.

2) HWTW 获取 TTE 将 TTE 内的 VPN、Context ID 与虚拟地址的 VPN、Context ID 进行比对.

3) 如果 Context ID、虚拟页号匹配,则从 TTE 内获取该 VPN 对应的 RPN; 否则在下一个 TSB 配置寄存器指示的 TSB 内进行查找.

4) 在 TSB 配置寄存器内的 ra_not_pa 被置为 1 的情况下,HWTW 将获取的 RPN 翻译成物理页号 PPN. RPN 到 PPN 的翻译需要 Real Range 寄存器和 Physical Offset 寄存器的支持,它们完全由 Hypervisor 维护. 如果 RPN 在 Real Range 寄存器的支持下得到翻译,HWTW 返回 PPN 的 TTE,继而将 TTE 填入 TLB; 如 RPN 未能转换,则产生 Real_address_mmu_miss 异常.

5) HWTW 将在每个有效的 TSB 进行查找直到找到一个匹配的 TTE 或将所有的 TSB 查找完为止.

6) 如果没有 TTE 得到匹配,硬件触发 data_access_MMU_miss 或 instruction_access_MMU _miss 异常.

此外,UltraSPARC T2 处理器还可以通过配置 MMU 使 HWTW 失效,此时 TLB 缺失的处理就完全由软件实现,典型 TLB 缺失软件处理过程如下:

1) 在 HWTW 无效的请情况下 TLB 缺失根据

是指令还是数据访存会触发名为 fast_instruction_access_MMU_miss 或 fast_data_ access_miss 的异常.

2) 异常首先由 Hypervisor 捕获,如果访存的地址是虚拟地址,Hypervisor 将该异常移交给上层的客户机操作系统处理.

3) 客户机操作系统使用软件的方式查找 TSB,如果相应 TTE 不存在,继续调用相应的处理程序在 Software translation table 内继续查找.

4) 获得相应的 TTE,将 TTE 提交给 Hypervisor,Hypervisor 将 TTE 内的实地址转换为物理地址之后,填充 TLB.

可见,TLB 缺失的处理根据 MMU 配置等情况的不同,有着不同的处理过程. TLB 缺失处理过程中的每一步都可能因某些条件不满足而产生异常,包括: Instruction_real_translation_miss、data_real_transl - ation_miss、Fast_instruction_access_MMU_miss、Fast_data_access_MMU_miss、IMMU_miss_HWTW、DMMU_miss_HWTW、instruction_invalid_TSB_entry,以及 data_invalid_TSB_entry 等. UltraSPARC T2 在 TLB 缺失的处理过程中在不同情况,这些成为引入虚拟化代价的重要因素.

当 Guest OS 需要向 TLB 填充地址映射时,首先通过调用 mmu_map_addr API 将 VA 和 TTE tag 映射信息发送给 Hypervisor. Hypervisor 则根据给 TTE 获取 VA 对应的 RA,并将 RA 转换为 PA 生成新的 TTE. 最后,通过写 MMU 相应的寄存器将其写入 TLB 中.

当 TLB 发生缺失时,该异常首先被 Hypervisor 捕获. 由于 Hypervisor 只维护 RA 到 PA 的映射,因此需要 Guest OS 参与提供 VA 到 RA 映射. Hypervisor 将 TLB 缺失请求发送给 Guest OS,它则调用 mmu_fault_area_conf API 通知 Hypervisor 存放该信息的内存位置.

也就是说,SPARC 处理器 MMU 虚拟化代价主要来自于 Guest OS 对 MMU 操作时的 hypercall 调用以及访存异常的处理代价.

3 代价评测

本实验环境基于 SUN 公司全系统(full system) SAM 模拟器来模拟 SPARC T2 多核处理器; SAM 之上加载 Hypervisor 虚拟化层,在 hypervisor 之上运行 Solaris 操作系统和应用. 本文通过对 hypervisor 和 SAM 大量修改,来捕获系统执行时触发的 hypercall,并对相关代价进行记录. 测试基准选择 qsort 和 basicmath,实验结果如图 2 所示.

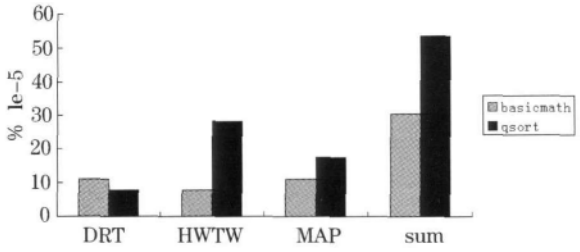

图 2 虚拟化代价评测

实验表明,对于这两个测试基准,DRT (data_real_translation_miss), HWTW (IMMU_miss_HWTW) 和 MAP(mmu_map_addr) 成为引入虚拟化代价的重要因素,三者引入的总代价在 10 -4 数量级. 这说明 SPARC T2 MMU 虚拟化引入可忽略的代价,获得满意效果.

4 结语

本文对 SPARC T2 处理器 MMU 内存管理单元虚拟化机制进行深入研究,对虚拟化引入的代价因素进行分析. 分析表明: MMU 虚拟化代价主要来自于 Guest OS 对 MMU 操作时 hypercall 调用以及访存异常的处理代价. 其中 data_real_translation_miss,IMMU_miss_HWTW 和 mmu_ map_addr 等 hypervisor API 成为引入虚拟化代价的重要因素. 实验表明 MMU 支持下虚拟化机制引入了可忽略的开销,获得满意效果.

参考文献:

[1] CREASY R J. The Origin of theVM/370 Time - sharing System [J]. IBM Journal of Research and Development, 1981, 25(5): 480 - 490.

[2] 刘晓亮. 基于虚拟化技术的云计算平台架构 [J]. 信息科技, 2011(23): 188.

[3] 复旦大学并行处理研究所, 英特尔开源软件中心. 系统虚拟化原理与实现 [M]. 北京: 清华大学出版社, 2009: 185 - 250.

[4] DAVID L W. OpenSPARC Internals [M]. USA: SUN Microsystems Inc., 2008: 23 - 24.

[5] Ultra SPARC. Virtual Machine Specification (The sun4v architecture and Hypervisor API specification), Revision 1.0 [M]. USA: SUN Microsystems Inc., 2006.