计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.20210140
59(4):737-746,2022

# 申威架构下的软件平滑嵌套页表

沙 赛[1,2] 杜翰霖[1] 罗英伟[1,2] 汪小林[1,2] 王振林[3]
[1](北京大学计算机科学技术系 北京 100871)
[2](鹏城实验室 广东深圳 518000)
[3](密歇根理工大学 美国密歇根州霍顿市 49931)
(ss_boom@pku.edu.cn)

# Software-Based Flat Nested Page Table in Sunway Architecture

Sha Sai[1,2], Du Hanlin[1], Luo Yingwei[1,2], Wang XiaoLin[1,2], and Wang Zhenlin[3]
[1](*Department of Computer Science and Technology, Peking University, Beijing* 100871)
[2](*Peng Cheng Laboratory, Shenzhen, Guangdong* 518000)
[3](*Michigan Technological University, Houghton, MI, USA* 49931)

**Abstract** The nested page table (NPT) model is an effective, hardware-assisted memory virtualization solution. However, the current Sunway processor lacks hardware support of NPT. However, the privileged programmable interface of Sunway architecture can be used to emulate the necessary hardware support with software. Hardware mode is the CPU privilege level unique to Sunway. This interface runs on the Sunway hardware mode with the highest CPU privileged level. In this paper, we propose the software-based flat nested page table (swFNPT) model for Sunway. In the programmable interface, we software-implement the hardware functions required by the nested page table model, such as nested page table walking. The new design makes up for the deficiency in hardware support through software optimization. In particular, the flat (one-level) nested page table is used to improve the efficiency of page walk. We use multiple benchmarks to test the performance of swFNPT. The experiments on a Sunway 1621 server show the promising performance of swFNPT. The average memory virtualization overhead of SPEC CPU 2006 is about 3% and the average overhead for SPEC CPU 2017 benchmarks with large working set is about 4%. The STREAM result shows that the memory bandwidth loss of swFNPT is less than 3%. Therefore, this paper provides a valuable reference for future development of hardware-assisted virtualization of Sunway server.

**Key words** address translation; memory virtualization; nested page table; Sunway architecture; virtual machine monitor

**摘 要** 嵌套页表是一种硬件辅助的内存虚拟化模型,当前国产申威处理器上未能提供该模型所需的硬件支持.然而申威架构特有的特权程序可编程接口可以通过软件构建必要的底层硬件支持.该接口运行在申威硬件模式上,具有最高CPU特权级.基于这一特性,在申威平台上实现了软件平滑嵌套页表模型swFNPT,通过软件设计优化弥补了硬件支持上的不足.特别地,使用平滑(1级)嵌套页表代替4级嵌套

**收稿日期**:2021-02-26;**修回日期**:2021-07-15
**基金项目**:国家重点研发计划项目(2018YFB1003604);国家自然科学基金项目(62032001,62032008,61672053,U1611461)
This work was supported by the National Key Research and Development Program of China (2018YFB1003604) and the National Natural Science Foundation of China (62032001, 62032008, 61672053, U1611461).
**通信作者**:罗英伟(lyw@pku.edu.cn)

页表来提升页表查询效率.使用多组测试程序测试该设计的性能.在申威 1621 服务器上的实验结果表明:swFNPT 整体性能良好.SPEC CPU 2006 的平均内存虚拟化开销约为 3%,SPEC CPU 2017 中大工作集程序的平均开销约为 4%,STREAM 内存带宽测试结果显示 swFNPT 的带宽损失低于 3%.这一工作可以为申威架构的硬件辅助虚拟化发展提供有价值的参考.

**关键词** 地址转换;内存虚拟化;嵌套页表;申威架构;虚拟机管理器

**中图法分类号** TP316

申威处理器采用自主指令集,是我国具有完全自主知识产权的处理器系列[1].初代申威指令集是在 Alpha 指令集的基础上进行扩展的,但经过不断发展与完善,现在已经成为独立的自主可控的指令集.申威处理器最为典型的应用场景是我国自主研发的神威·太湖之光超级计算机.太湖之光超级计算机搭载了 40 960 个申威 26010 众核处理器,其性能十分优越,已蝉联多次全球超算冠军[2].申威架构的应用除众核超级计算机之外,还有多核服务器,如申威 1621 服务器.本文的研究工作主要针对 1621 型服务器展开,探讨申威处理器上高效的内存虚拟化解决方案.该解决方案依托申威架构特性,并适合在同一架构下的多种型号的服务器上进行推广.虚拟化技术是云服务产业的核心支持技术之一,可以有效提高物理资源的利用率和系统的安全性[3-4].申威架构下的虚拟化解决方案并不完善,仍处于发展阶段.经过近年来的不断发展,申威架构已经形成了基本的虚拟化框架.虚拟化技术分为 3 个主要方面:CPU 虚拟化、内存虚拟化和 I/O 虚拟化[3].申威在 CPU 虚拟化和 I/O 虚拟化已经有了较为完善的解决方案.而内存虚拟化方面,当前申威虚拟机采用固定大小的预留段式内存进行直接映射,虚拟机使用固定的物理内存空间,虚拟机的数量及整个物理机内存的使用极其受限,并不是真正意义上的内存虚拟化[5].计算机系统中,CPU 访问内存首先需要进行地址转换,即虚拟地址转换成物理地址.非虚拟化环境下只需要 1 维地址转换.当前申威虚拟机的内存访问和非虚拟化环境下相似,只是一种权宜的解决方案.

作为最为复杂的虚拟化技术,内存虚拟化需要完成 2 维(2D)地址转换,即客户机虚拟地址(guest virtual address, gVA)到客户机物理地址(guest physical address, gPA)再到宿主机物理地址(host physical address, hPA)的转换(gVA→gPA→hPA).现有的内存虚拟化解决方案主要包括 3 类:影子页表(shadow page table, sPT)模型、嵌套页表(nested page table, NPT)模型以及直接页表[3].前 2 种属于完全内存虚拟化,虚拟机操作系统无需做任何修改;第 3 种属于半虚拟化,需要修改虚拟机操作系统的源代码.

本文工作主要针对完全虚拟化解决方案.影子页表模型属于软件内存虚拟化,不需要硬件支持.该模型的主要思想是采用一个页表(即影子页表)来直接保存客户机虚拟地址到宿主机物理地址(gVA→hPA)的映射.在虚拟机进行地址转换过程中,内存管理单元(memory manage unit, MMU)直接加载当前进程的影子页表,加速 2 级地址转换.影子页表依赖写保护机制来同步客户机页表(guest page table, gPT)和影子页表(sPT),这种模式造成了严重的虚拟机退出(VMExit)开销.

嵌套页表模型属于硬件辅助内存虚拟化,需要有专门的硬件支持,其代表有 Intel 拓展页表(extended page table, EPT)和 AMD 嵌套页表,两者在原理和实现上基本一致[6].嵌套页表保存了虚拟机物理地址到宿主机物理地址的映射关系.嵌套页表模型的核心思想是 MMU 加载虚拟机进程页表,在页表查询(page walk)的过程中,从进程页表中获取虚拟机物理地址,然后立即访问嵌套页表,转换成宿主机物理地址,再进内存访问.嵌套页表模型实现了地址转换过程中的 2 维页表代换,避免了影子页表模型因同步产生的开销.然而 64 位机器在不开启大页机制的情况下,都具有 4 级页表结构.嵌套页表模型会导致 1 次地址转换中存在远超非虚拟化环境下的多次内存访问.若无硬件辅助支持,嵌套页表模型在 1 次 2 维页表查询过程中需要 24 次访存[7-8].因此,处理器采用必要的硬件进行辅助加速.硬件主要包括页表查询缓存(page walk cache, PWC)和嵌套旁路转换缓存(nested translation lookaside buffer, NTLB).PWC 缓存了页表项(包括页表物理页帧号和页内偏移)为索引、次级页表页物理页帧号为值的映射.PWC 可以减少页表查询产生的访存次数[9].NTLB 通过缓存 gPA 到 hPA 的映射来优化 4 级嵌套页表

的查询[6].虽然 NTLB 可以有效减少嵌套页表查询的访存次数,但是 NTLB 的大小十分有限,尤其是对于访存局部性较差的程序,系统就会产生较多的 NTLB miss,2 维 4 级页表的嵌套查询仍会造成显著的性能开销.

目前申威架构缺乏嵌套页表所需的硬件辅助,因此,难以实现传统意义上的嵌套页表模型.幸运的是,申威架构具有独特的特权程序可编程接口(hardware mode code, HMcode),提供了底层软件灵活性[10].在 HMcode 中,系统开发者可以通过软件实现丰富的类硬件支持,特别是虚拟化支持,如虚拟中断、虚拟机退出/陷入(VMExit/VMEntry)等.基于申威架构这一特性,本文实现了申威架构下的平滑嵌套页表模型(swFNPT).该模型通过纯软件的优化,弥补了申威架构硬件不足的缺点,实现了申威处理器上真正意义的内存虚拟化.

## 1　相关工作

申威处理器现有的内存虚拟化方案采用了预留内存的直接映射模型实现虚拟机的内存访问.该模型要求宿主机将物理内存进行预留,仅虚拟机可使用这部分内存.因为直接使用"基址+偏移"的方式进行 1 维地址转换,这种设计不存在额外的内存虚拟化开销.然而,这种实现既不符合内存虚拟化的理念,在资源使用上也存在着资源利用率低、灵活性差等问题.

Gandhi 等人[7]也提出通过直接映射模型加速虚拟化环境下的地址转换,从而减少页表访问带来的开销.他们先是在非虚拟化环境下,针对大工作集程序实现了一种段式内存映射,然后将这种方案拓展到内存虚拟化当中.他们为 1 个虚拟机申请充足的连续物理内存,该虚拟机的地址转换就变成了简单的段基址加偏移,这样直接避免了 4 级页表的访问,提高了转换效率.但是,这种直接映射要求虚拟机的物理内存是连续的,这可能导致一些内存资源的损失,降低了内存利用率;另一方面,这种设计直接丧失内存虚拟化带来的灵活性,不能够根据应用的需要调节虚拟机的内存,也难以细粒度地进行内存管理.这种设计也难以和现有的一些虚拟机功能有效兼容,比如虚拟机热迁移.

Ahn 等人[11]提出将 4 级嵌套页表改为 1 级嵌套页表,即平滑嵌套页表(flat nested page table).该设计将 2 维嵌套页表查询最坏情况下需要 24 次访存缩减到最多 9 次访存.该设计保留硬件辅助虚拟化支持,是通过全功能模拟器进行评估,由于进行了必要的硬件改造,无法在真机上实现.HMcode 恰好为申威处理器实现这一设计提供了条件.本文吸收了该工作的思想,在申威架构下实现了平滑嵌套页表来缓解页表查询压力.

采用大页机制可以有效缓解嵌套页表查询压力.因为使用大页可以将 4 级页表缩短为 3 级甚至 2 级页表,加快了页表查询的速度.同时每个映射都具有更大的映射空间,TLB miss 率也会显著下降.然而,大页机制存在着碎片化严重、资源浪费、灵活性差、不支持虚拟机热迁移等诸多问题.针对这些问题,研究者们提出了一系列的解决方案来缓解大页机制带来的弊端.Guo 等人[12]在 VMware 上提出了主动积极拆分大页的技术方案来缓解大页导致的内存资源浪费问题;针对大页机制不适应于 NUMA 架构的问题,Gaud 等人[13]提出了改良版的 Carrefour 算法来保障大页机制在 NUMA 架构上的性能;Pham 等人[14]则提出了一种推测式大页机制来进行大页的轻量级管理.

研究者们在内存虚拟化领域进行诸多尝试与探索,取得了一系列丰富成果.然而一方面,一些方法需要依赖一定的硬件改造;另一方面,由于申威架构的特殊性,针对申威架构的内存虚拟化工作还存在很大的空白.本文借鉴吸收国内外先进的研究经验,提出了首个申威架构下的纯软件的嵌套页表内存虚拟化解决方案.

## 2　申威虚拟化框架

本节主要介绍申威架构虚拟化框架的相关实现,主要包括申威架构 CPU 特权级模型、虚拟化模型支持以及申威 HMcode.

### 2.1　申威 CPU 模式

如图 1 所示,申威 CPU 具有 4 个特权模式,由高到低分别是硬件模式(L-0)、虚拟模式、核心模式(L-1)以及用户模式(L-2).特有的 HMcode 运行在最高权限的硬件模式.该接口主要用于代替部分硬件功能,这些功能用硬件实现过于复杂,而又无法用常规程序实现,例如页表代换、进程上下文切换、TLB 刷新等;也用于实现一些原子操作和一些兼容不同平台的代码.虚拟模式暂未启用;核心模式类似于 X86 架构的内核模式,操作系统运行在该模式下;用户模式具有最低的特权级,用户程序在该模式

下执行.其他非硬件模式都可以通过受限的系统调用(syscall)调用硬件模式下的特权指令实现一些底层支持.

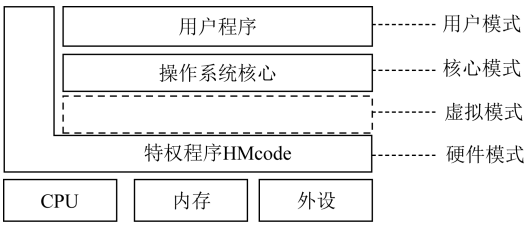



Fig. 1 The privilege modes of Sunway CPU

图 1 申威 CPU 特权模式

### 2.2 申威虚拟化模式

如图 2 所示,申威 CPU 具有和 Intel X86 VT-X 根/非根模式(root/non-root mode)类似的虚拟化模式.宿主机和客户机分别在根/非根模式下运行,但两者都具有核心模式和用户模式.宿主机操作系统及虚拟机管理器(virtual machine monitor, VMM,也称 hypervisor)运行在根模式下的内核模式特权级.VMEntry 实现根模式到非根模式的切换,VMExit 实现非根模式到根模式的切换.例如,虚拟机触发时钟中断处理时,虚拟机触发 1 次 VMExit.首先,系统保存客户机上下文信息,其中包括主要的 CPU 寄存器状态、栈指针、欲执行的指令地址(PC)等.然后系统恢复宿主机上下文信息,陷入 VMM 进行时钟同步处理;处理结束后,CPU 触发 VMEntry,恢复虚拟机上下文信息,进入虚拟机,继续执行指令.和 X86 不同的是,申威架构具有最高权限的硬件模式,该模式不受根/非根虚拟化模式的限制,任何模式都可以通过受限的系统调用陷入硬件模式.

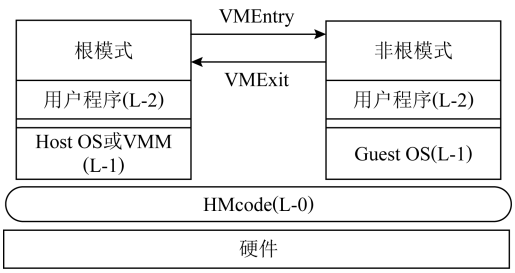



Fig. 2 The virtualization mode of Sunway CPU

图 2 申威 CPU 虚拟化模式

### 2.3 申威特权程序可编程接口(HMcode)

HMcode 运行在最高特权的硬件模式(如图 1、图 2 所示),该接口处于内核和硬件之间,属于固件.代码保存在 flash 部件中,形式上类似于 RAM BIOS.特权程序可以以物理地址直接访问全局地址空间,也可以直接访问寄存器.特别地,申威架构是软件管理的 TLB,这为实现内存虚拟化提供了必要支持.TLB miss 之后,特权入口 TLB_MISS_ENTRY 接管 TLB miss 处理,进行页表查询和 TLB 填充.我们在 TLB_MISS_ENTRY 中实现嵌套页表的相关逻辑,具体实现在第 3 节进行详细阐述.HMcode 为上层应用提供丰富的系统调用接口,但不同 CPU 模式具有不同的调用权限,例如内核进程可使用如进程上下文切换、读寄存器、TLB 刷新等功能的系统调用;而用户进程无法使用此类内核级调用.

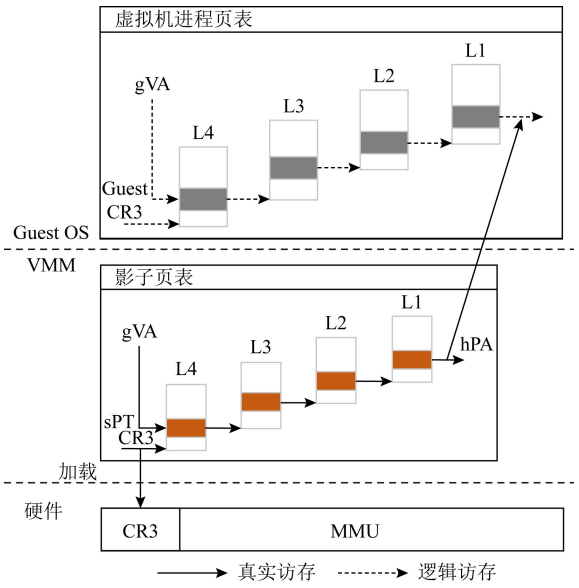
## 3 申威架构下的软件平滑嵌套页表

### 3.1 传统内存虚拟化模型

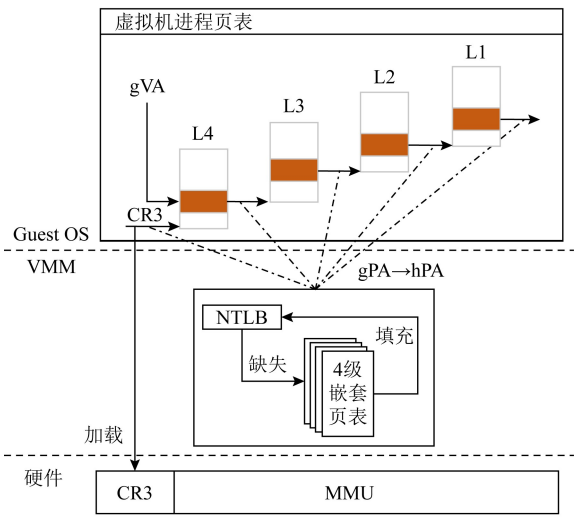
现有的完全内存虚拟化模型包括影子页表模型和嵌套页表模型.图 3 展示了 2 种模型的工作原理.

图 3(a)是影子页表模型的工作原理.VMM 为每个虚拟机进程建立一套影子页表,用于保存客户机虚拟地址到宿主机物理地址之间的映射.在地址转换过程中,一旦发生 TLB 缺失,CPU 触发页表查询过程.实际加载到页表基地址寄存器 CR3 中的是影子页表基地址,所以 MMU 对影子页表进行页表查询.影子页表模型的优点是通过 1 维页表缓存 2 维地址转换,提高了地址转换的效率.然而,VMM 需要维护影子页表的正确性,一旦客户机操作系统修改了客户机进程页表,影子页表也需要进行相应的修正.这就是影子页表模型最大的开销所在,即如何同步客户机进程页表和影子页表.客户机修改进程页表不是系统级事件,VMM 无法感知这一操作.传统的方式是写保护机制,VMM 对所有的客户机进程页表页施加写保护,一旦进程页表页被修改,就会触发写保护错误,进而触发 VMExit 而陷入 VMM.所以,影子页表模型的缺点就是不适应于高频页表修改的应用程序(如 gcc).过度的页表修改会导致大量的 VMExit,每一次 VMExit/VMEntry 都需要 TLB 清空、上下文切换等操作,开销十分显著.

图 3(b)是嵌套页表模型的工作原理.嵌套页表模型的基本思想是通过嵌套页表保存 gPA 到 hPA 的映射,虚拟机在进程页表查询时获取 gPA,然后访问嵌套页表进行第 2 维的地址转换(gPA→hPA).相较于影子页表模型,嵌套页表模型消除了因为影子页表同步造成的虚拟化开销,并且有效降低了内存虚拟化的复杂程度.然而,嵌套页表模型会导致更多的页表查询开销.64 位系统具有 4 级页表,每一级




(a) 影子页表模型




(b) 嵌套页表模型

Fig. 3　The principles of shadow page table model and nested page table model

图 3　影子页表模型和嵌套页表模型原理

进程页表查询都需要进行 1 次嵌套页表的查询,每一级嵌套页表的查询都需要 1 次访存操作.页表基地址寄存器(CR3)、4 级进程页表及数据页地址各 1 次转换,因此在无硬件辅助支持的条件下共需 24 次访存[15].为了缓解这一开销,研究者们提出了 NTLB 的硬件部件,用于直接保存第 2 维的地址映射,其基本功能等同于 CPU TLB.因为是硬件部件,因此具有极高的查询效率(数个 CPU 周期).NTLB 可以显著提高嵌套页表的查询效率.

但是,目前的申威芯片仍处于不断发展与完善阶段,很多相关的硬件支持(比如 DMA 等)均未实现,嵌套页表所需的相关硬件同样也未提供.硬件支持的缺乏意味着如果在申威处理器上实现嵌套页表模型,每次地址转换都需要 24 次访存,相较于非虚拟化环境的 4 次访存,显然这一开销是难以接受的.因此,我们改进了传统的嵌套页表模型,基于申威架构的 HMcode,在申威处理器上实现了首个平滑嵌套页表的内存虚拟化解决方案.一方面为申威虚拟机实现真正意义的内存虚拟化,另一方面我们也希望通过这一方案为新一代申威处理器的硬件辅助虚拟化设计提供相应的理论和实验支持.

### 3.2　平滑嵌套页表

申威架构缺少硬件上的内存虚拟化支持,因此在地址转换开销上难以接受.为了解决这一问题,我们采用平滑(1 级)嵌套页表机制来缓解开销.图 4 展示了平滑嵌套页表基本原理.平滑嵌套页表由一段连续的物理内存存储,任何对于该页表的查询都只需要根据“基地址+偏移”的模式进行访问.平滑嵌套页表以客户机物理页帧号(guest-physical frame number, GFN)为索引,其中每个条目保存着对应的宿主机物理页帧号(physical frame number, PFN).宿主机物理页帧号和客户机物理地址的页内偏移组成宿主机物理地址.相较于 4 级嵌套页表,访问 1 次平滑嵌套页表仅需要 1 次访存,整个 2 维嵌套页表查询最多需要 9 次访存.然而平滑嵌套页表的设计需要重构查询嵌套页表的 MMU,这在 X86 服务器上无法实现.而申威架构 HMcode 提供了底层支持的软件可编程性,我们可以在 HMcode 中软件编程实现嵌套页表所需的一些底层支持.

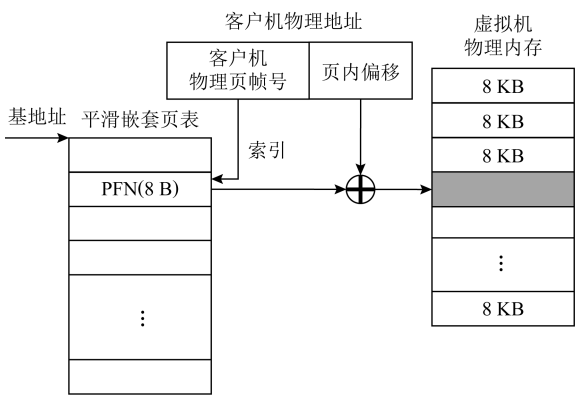



Fig. 4　The principle of flat nested page table

图 4　平滑嵌套页表原理

平滑嵌套页表的大小由其对应的虚拟机物理内存大小决定.平滑嵌套页表需要完整覆盖虚拟机物理内存的页式映射,即页表中条目数等于虚拟机物理页数.申威架构的页面大小是 8 KB,若虚拟机内存

为 $x$(单位为 GB),则平滑嵌套页表共有$\frac{x}{8\,\mathrm{KB}}$个条目.在平滑嵌套页表中每个条目大小为 8 B,则页表存储开销为 $x$(单位为 MB),这一开销(1/1000)是可以接受的.

### 3.3 基于申威架构的平滑嵌套页表设计

平滑嵌套页表模型主要分为 3 部分:页表初始化、页表查询和页表缺页处理.当前的申威服务器以 QEMU/KVM 为基本虚拟化框架,底层的页表查询模块实现在 HMcode 中.平滑嵌套页表将依托于申威架构特性来设计,图 5 展示了申威架构下平滑嵌套页表的基本框架.

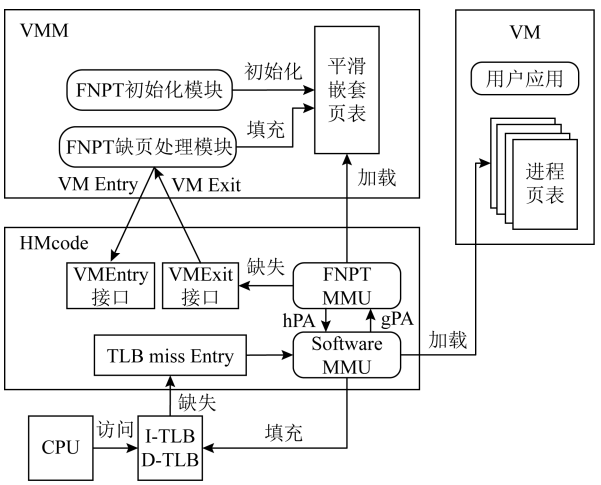



Fig. 5 The framework of flat nested page table under Sunway architecture

图 5 申威架构平滑嵌套页表框架

1) 平滑嵌套页表的初始化.VMM 需要根据虚拟机的物理内存大小来申请相应的连续内存保存平滑嵌套页表.

2) 平滑嵌套页表的查询.查询平滑嵌套页表的 MMU(FNPT MMU)应实现于 HMcode.目前申威的 MMU 是以软件形式实现在 HMcode 中.Software MMU 在访问每一级客户机进程页表时需要通过 FNPT MMU 访问平滑嵌套页表将 gPA 转换成 hPA.

3) 平滑嵌套页表缺页处理.FNPT MMU 查询页表时可能触发缺页中断,通过 VMExit 接口进入 VMM 中进行缺页处理.缺页处理结束之后,通过 VMEntry 接口重新进入客户机完成地址转换.

### 3.4 平滑嵌套页表在申威下的软件实现

我们在申威 1621 服务器上实现了平滑嵌套页表模型.在虚拟机运行过程中,CPU 根据客户机虚拟地址查询 TLB 进行地址转换.系统发生 TLB 缺失,立刻陷入 TLB 缺失处理入口进行 TLB 缺失处理.MMU 加载客户机进程页表开始进行 2 维页表查询,客户机进程页表保存了 gVA 到 gPA 的映射关系,每一级页表的页表项都保存了 1 个客户机物理页帧号和对应的权限位信息.首先,MMU 访问 FNPT MMU 尝试将进程页表基地址(gPA)转换成 hPA;然后,4 级进程页表的每一轮查询都需要访问 FNPT MMU 将 gPA 转化成 hPA;最后,MMU 将 gVA 到 hPA 的映射使用软件填充 TLB.至此,1 次虚拟化环境下的地址转换完成.

在嵌套页表模型中,系统可能发生 2 种缺页中断:1)客户机进程页表不完整导致的缺页中断;2)嵌套页表不完整导致的嵌套页表缺页中断.当 MMU 查询客户机进程页表时,会检查页表项的权限位,若有效位为 0,则触发 1 次客户机缺页异常处理,系统进入虚拟机内核缺页处理程序进行缺页处理,填充进程页表.缺页中断处理结束后,系统再次执行产生 TLB miss 重新开始地址转换.当 FNPT MMU 查询平滑嵌套页表时,有可能发生缺页中断,此时触发 1 次嵌套页表缺页处理.

系统通过 VMExit 接口进行上下文切换,CPU 进入 VMM 进行嵌套页表缺页处理.首先根据传入的 GFN 调用 *gfn_to_pfn_memslot*()接口将 GFN 转化成对应的 PFN.该接口首先将虚拟机物理页帧号转化成宿主机虚拟页号,因为虚拟机物理地址空间和虚拟机管理进程(QEMU 进程)的虚拟地址空间都是连续的,因此两者之间通过“基地址+偏移”的方式直接转换;其次,该接口查询宿主机进程页表,将虚地址转化成物理地址;最后 VMM 以 GFN 为索引,将对应的 PFN 插入嵌套页表.至此,1 次嵌套页表的缺页处理完成,VMM 调用 VMEntry 接口切换上下文,重新陷入虚拟机,CPU 再次执行产生 TLB 缺失的指令,重新进行地址转换.

## 4 实验与分析

### 4.1 实验环境

我们在真机环境下评估了申威架构下的平滑嵌套页表模型.实验组物理机是申威 1621 服务器,该服务器具有 16 个物理核心,主频约为 1.6 GHz,搭载深度操作系统 15.02.对照组物理机是基于 X86 架构的 Intel 服务器,主频约为 2.1 GHz,搭载的是 Ubuntu 16.04 服务器版操作系统.具体参数如表 1 所示.

**Table 1　Experimental Machine Configuration**

**表 1　实验机参数配置**

| 软硬件参数 | 处理器配置 | |
|---|---|---|
| | 申威 | X86 |
| CPU | Sunway 1621 | Intel® Xeon® Silver 4216 |
| 核数 | 16 | 16 |
| 主频/GHz | 1.6 | 2.1 |
| ITLB | 32-entries | 64-entries |
| DTLB | 64-entries, 512-entries | 64-entries 1536-entries |
| CPU 缓存 | L1 I/D 32 KB L2 512 KB | L1 I/D 32 KB, L2 1 MB L3 22 MB |
| 内存/GB | 32 | 32 |
| 虚拟机内存/GB | 20 | 20 |
| 硬盘/TB | 2 | 2 |
| Linux 内核 | 4.4.15 | 4.4.15 |
| VMM | QEMU-KVM 2.5.0 | QEMU-KVM 2.5.0 |

### 4.2　实验设计

因为现有的申威服务器不具有真正意义的内存虚拟化功能,所以对照组机器选择相同软件配置的 Intel X86 服务器.在 X86 架构下,我们对影子页表模型(X86-SP)和拓展页表模型(X86-EPT)分别进行了测试.

嵌套页表模型需要预热处理(warm-up).在每轮测试程序运行前,需要首先运行一个较大工作集程序进行预热处理,保证嵌套页表完成缺页处理.若在虚拟机启动初期直接运行测试程序,系统会产生大量的嵌套页表缺页处理中断,降低程序性能.这种处理是考虑到云服务器的应用都具有较长工作周期,嵌套页表的缺页处理一般只发生在虚拟机启动前期,预热操作之后的实验结果更能体现虚拟机在一个长周期内的真实性能.

我们选择了 SPEC CPU 2006,SPEC CPU 2017,Graph500,Memcached 等应用作为测评程序.SPEC CPU 2006 采用 ref 集进行测试.SPEC CPU 2006 测试程序工作集普遍较小(均低于 3 GB),为进一步验证模型的效果,我们从 SPEC CPU 2017 测试集中挑选了 8 个具有较大工作集的测试程序.表 2 展示了这 8 组程序的工作集信息.我们也选用了较为典型的云服务程序 Memcached 和 Graph500 测试 swFNPT 模型的性能.最后我们使用 STREAM 评估虚拟机系统内存带宽损失.

**Table 2　Benchmarks of SPEC CPU 2017**

**表 2　SPEC CPU 2017 测试程序**　GB

| 程序 | 工作集 |
|---|---|
| 602.gcc | 7.8 |
| 603.bwaves | 11.6 |
| 605.mcf | 3.8 |
| 607.cactuBSSN | 6.9 |
| 619.lbm | 3.3 |
| 631.deepsjeng | 7.0 |
| 649.fotonik3d | 9.8 |
| 657.xz | 15.7 |

### 4.3　实验结果及分析

#### 4.3.1　内存虚拟化整体性能测试

SPEC CPU 2006 测试集具有多个测试子程序,不同程序具有不同的访存特征,能够较为全面地反映系统访存性能.表 3 展示的是 swFNPT,Intel X86-EPT,Intel X86-SP 这 3 组模型使用 SPEC CPU 2006 测试集的实验结果.实验结果表明,适配 swFNPT 模型的申威虚拟机整体性能良好,SPEC CPU 2006 全集平均性能开销仅为 3.24%.申威服务器内存虚拟化与 Intel X86 下 2 组模型的平均性能开销基本相当.

如第 3 节所述,相较于传统的影子页表模型,嵌套页表的主要优点是消除了页表同步的开销.这一优点展现在具有频繁修改页表特性的测试程序(如 403.gcc)上.如表 3 所示,基于嵌套页表的 X86-EPT 和 swFNPT 在 403.gcc 程序的开销均低于基于影子页表的 X86-SP.

在 SPEC CPU 2006 的实验结果中 429.mcf 具有超过 20% 的性能开销.且 X86 架构下的 Intel X86-EPT 同样具有高达 8.39% 的虚拟化开销,但 Intel X86-SP 的开销低于 5%,这是由该程序的访存特性所决定的.429.mcf 是典型的局部性较差的程序,这导致该类程序的 TLB 缺失率高,MMU 压力更大.在地址转换过程中,该类程序需要频繁进行页表查询操作.64 位操作系统采用 4 级页表,基于影子页表模型的地址转换最多产生 5 次存储访问,考虑到 PWC 硬件支持,这一开销会更小[9].相反地,采用嵌套页表模型,无论是 Intel EPT 还是 swFNPT 都需要 2 维页表查询,这大大加剧了页表查询的开销.这一特性在之前的研究工作中也有明确体现.Pham 等人[14]指出使用嵌套页表模型的虚拟化系统中运行 429.mcf,地址转换在整个程序运行周期中占比

高达 40%.我们在 Intel X86 机器上通过硬件计数器统计地址转换开销并建模实验获得了近似的实验结果.此外,在非虚拟化环境下 429.mcf 仍有 22%的执行时间用于地址转换.汪小林等人[16]的工作也明确指出 429.mcf 在使用嵌套页表模型时具有明显的内存虚拟化开销.

**Table 3 Experimental Results of SPEC CPU 2006**

**表 3 SPEC CPU 2006 测试结果** %

| 测试程序 | 虚拟化开销 | | |
|---|---|---|---|
| | X86-SP | X86-EPT | swFNPT |
| 400.perlbench | 3.24 | 2.94 | 1.28 |
| 401.bzip2 | 1.85 | 1.18 | 0.04 |
| 403.gcc | 7.34 | 4.90 | 2.40 |
| 410.bwaves | 2.15 | 5.81 | 8.10 |
| 416.gamess | 1.84 | 0.17 | 0.48 |
| 429.mcf | 2.80 | 8.39 | 20.71 |
| 433.milc | 6.29 | 5.68 | 2.55 |
| 434.zeusmp | 2.48 | 4.96 | 6.73 |
| 435.gromacs | 1.32 | 1.32 | 0.08 |
| 436.cactusADM | 2.45 | 0.82 | 0.17 |
| 437.leslie3d | 3.43 | 1.56 | 0.42 |
| 444.namd | 1.98 | 0.20 | 0.46 |
| 445.gobmk | 1.52 | 1.02 | 2.38 |
| 447.dealII | 5.66 | 7.01 | 0.56 |
| 450.soplex | 1.64 | 1.23 | 3.72 |
| 453.povray | 6.04 | 3.85 | 0.97 |
| 454.calculix | 1.51 | 0.47 | 0.55 |
| 456.hmmer | 1.85 | 0.21 | 0.11 |
| 458.sjeng | 1.11 | 2.38 | 2.32 |
| 459.GemsFDTD | 2.62 | 4.19 | 4.60 |
| 462.libquantum | 1.32 | 0.00 | 1.12 |
| 464.h264ref | 8.79 | 8.63 | 0.21 |
| 465.tonto | 1.58 | 0.00 | 0.86 |
| 470.lbm | 4.04 | 0.93 | 4.91 |
| 471.omnetpp | 4.45 | 7.19 | 9.81 |
| 473.astar | 2.90 | 8.32 | 9.17 |
| 481.wrf | 1.74 | 0.00 | 2.45 |
| 482.sphinx3 | 1.64 | 0.00 | 2.28 |
| 483.xalancbmk | 5.00 | 4.17 | 4.63 |
| 平均值 | 3.12 | 3.02 | 3.24 |

429.mcf 实验结果表明,Intel X86-EPT 比申威 swFNPT 内存虚拟化开销更小.这一差距主要原因分为 2 个方面:1)硬件支持(PWC,NTLB)和 TLB;2)Cache 容量.PWC 和 NTLB 可以有效减少 2 维页表查询中的访存次数,申威服务器暂未实现这 2 个功能部件.尽管 swFNPT 将每次页表查询中的访存次数控制在 9 次,但是相较于具有硬件辅助的内存虚拟化实现,swFNPT 平均的访存次数仍然较高.由表 1 可知,Intel 服务器的 1 级 TLB 容量为申威服务器的 2 倍,2 级 TLB 容量为申威服务器的 3 倍.Intel 服务器的 2 级 Cache 容量为申威服务器的 2 倍,且 Intel 服务器具有第 3 级 Cache.更大的 TLB 容量将大大降低程序的 TLB 缺失率,缓解 MMU 压力.更大的 2 级 Cache 容量将有助于提升地址转换中访存效率.

### 4.3.2 大工作集程序虚拟化性能测试

为进一步验证平滑嵌套页表模型的性能,我们在申威服务器上运行测试 SPEC CPU 2017 中 8 个内存用量较大的程序.实验结果如表 4 所示,平滑嵌套页表模型的平均开销仅为 4.12%,最大开销不超过 9.2%.这一实验结果说明该模型在较大内存压力的情况下仍有较好的性能.特别地,结合 4.3.1 节 SPEC CPU 2006 的实验结果,我们发现 605.mcf 程序虚拟化开销明显低于 429.mcf 程序的开销.这是因为后者局部性更差,这一特性也有相关研究给出[17].

**Table 4 Experimental Results of SPEC CPU 2017**

**表 4 SPEC CPU 2017 测试结果**

| 测试程序 | 内存/GB | 虚拟化开销/% | | |
|---|---|---|---|---|
| | | X86-SP | X86-EPT | swFNPT |
| 602.gcc | 7.8 | 7.96 | 4.72 | 3.43 |
| 603.bwaves | 11.6 | 3.67 | 2.31 | <1 |
| 605.mcf | 3.8 | 1.83 | 6.62 | 4.26 |
| 607.cactuBSSN | 6.9 | 1.72 | 1.02 | 3.87 |
| 619.lbm | 3.3 | 0.13 | 1.84 | 3.79 |
| 631.deepsjeng | 7.0 | 3.55 | 7.28 | 4.39 |
| 649.fotonik3d | 9.8 | 2.80 | 3.93 | 9.20 |
| 657.xz | 15.7 | 6.79 | 10.49 | 4.50 |
| 平均值 | | 3.56 | 4.78 | 4.12 |

### 4.3.3 典型云服务应用内存虚拟化性能测试

表 5 展示了 Memcached 和 Graph500 的 2 组典型云服务应用的实验结果.Memcached 使用 memcslap 脚本进行测试,其中并发参数为 100;Graph500 使用 OpenMPI 框架运行节点数为 $2^{20}$.结果表明 Memcached 测试中,swFNPT 和 X86-EPT 虚拟化开销均低于 X86-SP.这是因为 Memcached 程序具有较为频繁的进程切换,而影子页表模型在进程切换时都需要产生 VMExit 进行页目录表寄存器的

更新,开销较大.Graph500 是较为典型的大规模并行程序,我们使用 OpenMPI 框架运行.该程序在 Intel X86-EPT 模型上的开销约为 5%,在 Intel X86-SP 模型上开销约为 6%,在 swFNPT 模型上开销低于 8%.

**Table 5　Experimental Results of Memcached and Graph500**

**表 5　Memcached 和 Graph500 测试结果**　%

| 测试程序 | 虚拟化开销 | | |
|---|---|---|---|
| | X86-SP | X86-EPT | swFNPT |
| Memcached | 10.64 | 2.98 | 5.20 |
| Graph500 | 5.71 | 5.14 | 7.72 |

#### 4.3.4　虚拟机内存带宽

STREAM 是广泛使用的用于测试系统内存带宽的工具.我们使用 STREAM 2.0 测试了虚拟化系统中的内存带宽.试验结果表明,相较于非虚拟化环境,swFNPT 产生的内存带宽损失低于 3%.我们在相同的软件环境下分别进行了 Intel X86-EPT 和 X86-SP 的实验,带宽损失分别为 2.88%和 3.27%.

#### 4.3.5　实验总结

4.3.1～4.3.4 节的实验结果表明:swFNPT 在大多数应用程序上表现良好,诸如 400.perlbench 等 18 个程序虚拟化开销均低于 3%.特别地,401.bzip2,435.gromacs,603.bwaves 等程序的虚拟化开销低于 1%.SPEC CPU 2017 的实验表明 swFNPT 模型在较大内存压力下也能表现出良好的性能.在内存带宽方面,STREAM 测试结果表明 swFNPT 内存虚拟化造成的带宽损失不高于 3%;Memcached 和 Graph500 的实验结果表明,swFNPT 在支持进程频繁切换和大规模并行方面表现良好.但是申威 swFNPT 在执行局部性较差的程序(如 429.mcf)时虚拟化开销较大,这主要受限于申威服务器硬件支持上的不足,这一实验结果为申威服务器下一步的发展提供切实有效的实验支持.

## 5　总结与展望

本文在国产申威架构上设计实现了首个软件平滑嵌套页表模型 swFNPT,在申威 1621 服务器上的实验结果表明,swFNPT 整体性能良好.这一工作为国产申威架构的硬件辅助虚拟化下一步发展提供有价值的参考.一方面,申威架构处理器要不断完善 MMU,NTLB,PWC 等的硬件支持;另一方面,申威处理器要保留特权程序可编程接口以提供底层软件灵活性.结合 NTLB 等硬件支持,未来软硬件结合的 swFNPT 将更能充分发挥申威架构的独特优势.

**作者贡献声明**:沙赛进行了该论文相关的系统设计、代码编写、论文撰写等工作;杜翰霖进行了系统测试;罗英伟进行了论文结构设计和论文修改;汪小林制定了论文相关的实验方案;王振林参与进行了系统设计和论文修改.

## 参　考　文　献

[1] Chengdu Shenwei Technology Co, Ltd. Sunway processer [EB/OL]. [2021-02-07]. http://www.swcpu.cn (in Chinese)
(成都申威科技有限责任公司. 申威处理器[EB/OL]. [2021-02-07]. http://www.swcpu.cn)

[2] Dongarra J. Report on the Sunway Taihulight system [EB/OL]. [2021-02-07]. http://www.netlib.org/utk/people/JackDongarra/PAPERS/sunway-report-2016.pdf

[3] The Intel Open Source Technology Center. System Virtualization: Principles and Implementation [M]. Beijing: Tsinghua University Press, 2009 (in Chinese)
(英特尔开源软件技术中心. 系统虚拟化——原理与实现[M]. 北京: 清华大学出版社, 2009)

[4] Barham P, Dragovic B, Fraser K, et al. Xen and the art of virtualization [J]. ACM SIGOPS Operating Systems Review, 2003, 37(5): 164-177

[5] Yao Jianguo, Lu Qiumin, Wang Xingyan, et al. SWVM: A light-weighted virtualization platform based on Sunway CPU architecture [EB/OL]. [2021-02-07]. https://link.springer.com/article/10.1007%2Fs11432-019-2769-4

[6] Bhargava R, Serebrin B, Spadini F, et al. Accelerating two-dimensional page walks for virtualized systems [J]. ACM SIGPLAN Notices, 2008, 43(3): 26-35

[7] Gandhi J, Basu A, Hill M D, et al. Efficient memory virtualization: Reducing dimensionality of nested page walks [C] //Proc of the 47th Annual IEEE/ACM Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2014: 178-189

[8] Gandhi J, Hill M D, Swift M M. Agile paging: Exceeding the best of nested and shadow paging [C] //Proc of the 43rd ACM/IEEE Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2016: 707-718

[9] Bhattacharjee A. Large-reach memory management unit caches: Coalesced and shared memory management unit caches to accelerate TLB miss handling [C] //Proc of the 46th Annual IEEE/ACM Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2013: 383-394

[10] Sunway 1621 RD Department. Sunway 1621 processor software interface manual [EB/OL]. [2020-02-01]. http://www.swcpu.cn/uploadfile/2018/0709/20180709030859351.pdf

[11] Ahn J, Jin S, Huh J. Revisiting hardware-assisted page walks for virtualized systems [J]. ACM SIGARCH Computer Architecture News, 2012, 40(3): 476-487

[12] Guo Fei, Kim S, Baskakov Y, et al. Proactively breaking large pages to improve memory overcommitment performance in VMware ESXi [J]. ACM SIGPLAN Notices, 2015, 50(7): 39-51

[13] Gaud F, Lepers B, Decouchant J, et al. Large pages may be harmful on NUMA systems [C] //Proc of the 14th USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2014: 231-242

[14] Pham B, Veselý J, Loh G H, et al. Large pages and lightweight memory management in virtualized environments: Can you have it both ways? [C] //Proc of the 48th Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2015: 1-12

[15] Merrifield T, Taheri H R. Performance implications of extended page tables on virtualized X86 processors [C] //Proc of the 12th ACM SIGPLAN/SIGOPS Int Conf. New York: ACM, 2016: 25-35

[16] Wang Xiaolin, Zang Jiarui, Wang Zhenlin, et al. Selective hardware/software memory virtualization [C] //Proc of the 7th Int Conf on Virtual Execution Environments. New York: ACM, 2011: 217-226

[17] Panda R, Song Shuang, Dean J, et al. Wait of a decade: Did SPEC CPU 2017 broaden the performance horizon [C] //Proc of the 24th IEEE Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2018: 271-282

**Sha Sai**, born in 1995. PhD candidate. His main research interests include system software, operating system, virtualization and memory management.

**沙 赛**,1995 年生.博士研究生.主要研究方向为系统软件、操作系统、虚拟化和内存管理.

**Du Hanlin**, born in 1999. Bachelor. His main research interests include system software, virtualization, and cloud computing.

**杜翰霖**,1999 年生.学士.主要研究方向为系统软件、虚拟化和云计算.

**Luo Yingwei**, born in 1971. Professor, PhD supervisor. His main research interests include system software, operating system, virtualization, and cloud computing.

**罗英伟**,1971 年生.教授,博士生导师.主要研究方向为系统软件、操作系统、虚拟化和云计算.

**Wang Xiaolin**, born in 1972. Professor, PhD supervisor. His main research interests include system software, operating system, virtualization, and cloud computing.

**汪小林**,1972 年生.教授,博士生导师.主要研究方向为系统软件、操作系统、虚拟化和云计算.

**Wang Zhenlin**, born in 1970. Professor, PhD supervisor. His main research interests are broadly in the areas of compilers, operating systems and system virtualization.

**王振林**,1970 年生.教授,博士生导师.主要研究方向为编译、操作系统和系统虚拟化.