**海量存储论文**

|  |
| --- |
| **实际系统和虚拟系统中改进地址转换的技术** |

|  |  |
| --- | --- |
| 院（系）名 称 ： | 计算机学院 |
| 专 业 名 称 ： | 计算机技术 |
| 学 生 姓 名 ： | 余云龙 |
| 学号： | 2017282110247 |

2017年12月

摘 要

本文综合了物理机系统和虚拟操作系统中提升地址转换一致性、转换的覆盖面积的方案，分析修改TLB（快表）对地址转换的影响。

在虚拟操作系统中，为了提升性能，操作系统需要经常进行修改虚拟地址到物理地址转换的活动。操作系统能在物理页面之间迁徙数据以管理异构内存设备，也就是页面重映射，但是开销很大。其中地址转换一致性占了很大一部分开销，特别是在虚拟操作系统上。转换一致性是专门用于转换的缓存（例如，TLBs，MMU caches等）与页表映射保持最新的手段，于是针对虚拟系统上转换一致性的大开销，就提出了硬件转换无效性和一致性（HATRIC），这是一种硬件机制，将该机制用在现有的cache一致性协议上，可以在KVM的虚拟化环境中测试发现，HATRIC能提升30%的性能，并减少10%的能耗，且只占用0.2%的CPU面积。

研究发现，在非统一的内存系统中，使用大页面会降低性能。现在每种技术都有适应自身的最佳内存块大小，但是在实际操作系统中不同的内存分配一致性往往不能给该技术提供最佳内存块。于是针对多样的内存分配方式，就提出了一种硬件、软件混合转换结构。这是软件和硬件混合的机制，可以有效适应不同的内存映射。该技术会将内存块连续的信息记录在页表项的一个子集中，称之为锚定项。使用TLB进行地址转换时，锚定项为其后的连续页面提供转换。因为少量的锚定项就可以覆盖大部分的虚拟地址空间，所以能大大提高TLB的效率。混合转换技术可以在硬件变动较小的前提下提升转换覆盖面积，同时还支持内存灵活分配。

关键词：地址转换；硬件软件混合；硬件转换一致性；TLB合并；虚拟内存

# 引言

为了应对大内存工作负载的系统，系统架构师开始使用异构内存进行架构。例如，英特尔在其Knight的Landing芯片中集成了高带宽的封装内存，并在多个产品里集成了3D Xpoint内存。AMD和Hynix即将发布高带宽内存（HBM）同样地，Micron的混合内存块和字节寻址内存正快速发展。厂商们将这些高性能内存和传统的DRAM相结合，促进了对异构内存架构的研究。

异构内存管理要求操作系统以不同的延迟、带宽、开销在内存设备之间重映射页面。页面重映射在迁移物理页面的场景下用的较少，但是在使用异构内存时就变得频繁了。因为在应用程序将数据迁移到内存设备必须用到页面重映射。因此，现在IBM和Redhat公司已经在部署Linux补丁集以便在兼容的异构内存设备之间重映射页面。

现在的问题是页面重映射有性能缺陷，且开销较大。原因有两方面，一是复制数据的开销大，二是转换一致性的代价大。我们着重关注第二个方面。已有的研究表明，转换一致性会消耗10-30%的系统运行时间，且在虚拟化系统上的开销会达到40%，因为在虚拟化系统中需要使用两个页表。转换一致性是专门用于转换的缓存（例如，TLBs，MMU caches等）与页表映射保持最新的手段。现在转换一致性需要更好的硬件支持，共需要三个属性的支持：精确的失效性、精确的目标识别、轻量级的目标端处理。现在供应商已经解决了这些需求，但是在虚拟操作系统中的转换一致性还没有实现这些目标。于是就提出了硬件转换失效性和一致性（HATRIC），一种关于本地TLB一致性的硬件机制。

随着大内存需求的应用程序对内存的需求越来越大，虚拟内存中的地址转换效率成为其性能发展的瓶颈。现阶段有2种提升地址转换效率的方法：第一种是增加转换覆盖面积，具体是一定的开销范围内扩大TLB转换覆盖面积。第二种方法是减少TLB未命中时的后续操作，也就是减少当内存在TLB种请求未命中时对未命中情况的处理延迟。方法二并不实用，因为当TLB命中时，处理延迟完全可以忽略。故最好是按照方法一，扩大转换面积直接提升性能。

在商业领域内，有一系列技术扩大转换面积，常用的方法有增加页面的大小，使用2MB的页面替换4KB的页面，这样就使得TLB单元的覆盖面积变为原来的512倍，同时系统还要给每个页面分配2MB的物理内存。其它的方法有在硬件层面的合并技术，如果多个页面在物理位置是连续的，CoLT和cluster TLB就可以将它们合并一个单独的TLB单元。但是硬件层面的合并技术完全是看运气合并页表的，且不能保证合并后的单元大小一样。

上述的方法在分配灵活性和转换覆盖面积的扩展性两方面是不能兼得的，页面大，页面数量就少，覆盖面积的扩展性也受限。硬件合并技术因为完全是由硬件完成的，它分配的灵活性较高，但是覆盖面积的扩展性限制在4-8个页面。

这些方法的共同点就是操作系统必须提供合适的连续内存块。但是在实际情况下，这种连续性较大的内存块分配非常复杂。因为内存结构并不是统一的，操作系统也就没办法针对不同的覆盖面积给出最佳内存分配策略。即使是同一个程序，也会因为系统状态的变化而改变内存分配状态。因此，能适应不同内存分配情况的硬件软件混合合并技术就提出来了。

# 研究背景

本节详细介绍硬件软件混合合并技术，以及硬件转换一致性机制。

## 硬件软件混和合并技术的动机

### 扩大转换覆盖面积

扩大转换覆盖面积主要有三种方法：

（1）使用多种页面大小

在x86系统中，TLB支持2MB和1GB的页面大小。如果2MB可用，操作系统就会给应用程序分配2MB的页面。使用多种大小的页面在TLB设计时不会产生明显的复杂性，最新的架构可以在L2 TLB中同时支持4KB和2MB的页面大小，也就不用给每种大小的页面分配单独的TLB了。但是，大页面的缺点就是它的覆盖面积仍然是有限的，而且覆盖面积的扩展性也会受限。

（2）使用多种大小的段

这种方法能大幅度提升转换覆盖面积。直接段和冗余内存映射（RMM）支持一种或多种大小的段。操作系统要给每个段区分配一个连续内存块，只要这种连续内存分配一直可行，单个段的转换覆盖面积就能扩展到非常大的虚拟地址空间，也就能消除大部分的地址转换开销。但是每个进程只能同时使用一定数目的段，而且这种基于段的转换非常依赖大的内存块。

（3）基于硬件的合并技术

因为操作系统使用伙伴算法来减少内存碎片，所以在内存分配方面有一定的连续性。基于硬件的合并技术利用TLB控制器找到这些连续分配的页面，这种搜索方式较看运气，故转换覆盖面积的扩展性相当有限。且在TLB单元中它们仅能做到4-8页的合并能力。

总的来说，三种方法在硬件成本、分配灵活性、覆盖面积的扩展性上各有长短。因此，为了能充分利用三者的优势，就提出了硬件软件混合合并技术。

### 提升内存中的不一致性

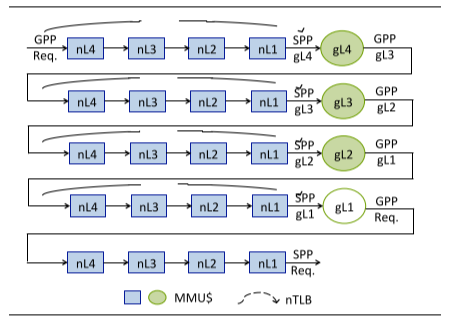
扩大转换覆盖面积要求分配连续的内存块，为了使用大页面就需要提升内存架构中的不一致性。大多数的数据中心系统使用多socket NUMA节点来提高系统密度，在NUMA系统中，线程调度往往与线程正在访问的位置不一致。因此就需要使用大页面防止不必要的内存访问。Gaud等人发现，使用2MB大小的页面会降低多线程应用的性能，而选用更大的页面就能改进这一现象。Kwon等人发现，在实际系统中，进程经常接收到不一致的大页面，降低性能。两项发现表明，当内存不一致性存在时，即使是2MB大小的页面也微不足道。因为大页面的使用可以通过提高内存不一致实现。

## 虚拟系统中地址转换一致性的动机

### 虚拟化中的硬件和软件支持

虚拟系统使用硬件或软件的方式来实现虚拟地址到物理地址的转换。一般地，虚拟机管理器使用影子页表将客户机虚拟页面（GVP）映射到系统物理页面（SPP），让它们与客户机操作系统页表保持同步。但是保持页表同步的开销比较大，因此大多数系统都是使用二维页表的。如图1所示，是二维页表遍历流程图

图1 虚拟系统中二维页表的遍历



客户页表将GVP映射到客户物理页面（GPP），嵌套页表再将GPP映射到SPP。x86-64系统使用4级正向树映射页表，按照从4（根级）到1（叶级）进行分级。当客户虚拟机中的进程进行内存查找时，其GVP转换为SPP。当查看访客页表（gL4-gL1）时，就要将GPP转换为页表所在的SPP。可以使用GPP来搜索嵌套页表（nL4-nL1）以找到SPP gL4。然后再生成下一个客户页表（gL3）的GPP。这个过程要重复24次才完成遍历，所以开销很大。

### 虚拟化中的页面重映射

本节介绍在虚拟系统中触发转换一致性。页面重映射可以按照迁移的数据和启动迁移的软件代理进行分类。

按照重映射的数据划分：系统重映射的页面中存储的内容有：a).客户页表；b)嵌套页表；c)非页表数据。大多数重映射的页面内容是c，只有不到1%的内容对应a和b。所以我们就注重HATRIC（硬件转换失效性和一致性）对c的操作。

按照重映射启动器划分：分为页面被客户机操作系统映射和被虚拟机管理器映射两种。客户机操作系统映射页面时，可以通过软件方法降低客户页表一致性的开销。但不能用于降低虚拟机管理器的页表重映射转换一致性开销。对于异构内存（如芯片堆栈寄存器），我们实际希望用嵌套页表重映射动态迁移数据以获取更好的性能。

# 技术介绍

本节主要介绍两种技术具体的实施方案和步骤，以及两者要解决的问题。

## 混合TLB合并技术简介

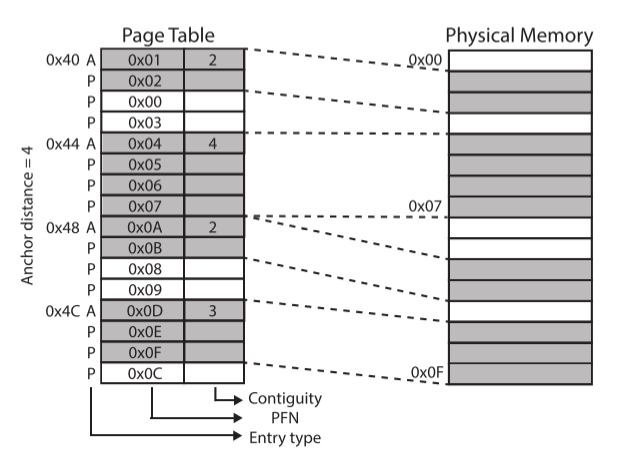
### 锚定页表

大页表、段和硬件合并技术可以在硬件资源有限的情况下扩展地址转换的覆盖面积。在这一节中，我们将平衡操作系统和结构组件的使用，使其内存分配更加灵活。同时得到更好的覆盖面积扩展性。

在该混合技术中，操作系统不再依赖硬件逻辑来确定连续页面，而是使用内存分配信息将连续状态记录到页表单元中。技术使用MMU（内存管理单元）中的组件来最小化TLB查找逻辑中所需要的硬件改变。为了记录连续块信息，每N个页表单元就被设计成一个锚点项。锚点放置在第N个单元上，N表示两个相邻锚点单元之间的距离。每个锚点项都包含从锚点项开始的后续页面有多少个是连续的信息，锚点项和锚点都像是一个常规的页表单元一样运转，。

如图2所示，是一个锚点项的页表。在该图中，锚点距离N是4，因此每4个单元就被指定成移动锚点项。锚页表使用未被使用的位来记录有多少后续页面是连续映射的。锚点距离N能确定反映进程中的内存连续性状态。

图2 页表中标记锚点项（锚点距离=4）每个锚记录从锚定地址开始连续映射

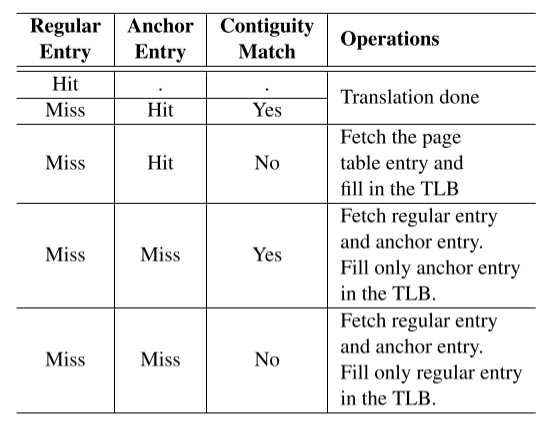


### 使用锚页表转换

要使用锚页表进行转换，只需要对MMU（内存管理单元）进行很小的更改就可以了。在不修改TLB结构的前提下，每个单元增加几个bit位以存储连续字段即可。锚点页表和常规页表共享相同的TLB。因为L1 TLB与核心联系紧密，并且性能受访问延迟的影响较大，因此对锚页表的支持其实是对L2 TLB的支持。

如图3所示，图3总结了所有情况下L2 TLB的操作流程。前2行分别表示常规TLB命中和锚命中，第3行表示锚查找成功的情况，但是对应的VPN不属于锚的连续块，故出现未命中。

图3 L2 TLB的操作



接下来就是查找锚记录了，对给定的VPN进行转换，将锚定距离与VPN对齐计算AVPN（锚点虚拟页号）。连续的AVPN都被映射到TLB的不同集合中，将VPN与相应AVPN之间的距离，与锚点TLB单元的连续值进行比较，确定给定VPN是否是其中的锚定子块，若是，则使用锚记录成功完成转换。

## 硬件转换一致性的详细设计

### 共同标签

当管理程序不知道GVP时，共同标签允许我们通过逻辑标签扩展来精确识别转换机制。换句话说，每个TLB，MMU缓存和nTLB条目都有自己的共同标签。共同标签存储嵌套页表项的系统物理地址。例如，GVP 1-SPP 2就使用了存储在系统物理地址为0x1010的嵌套页表单元，该地址下存储着共同标签。

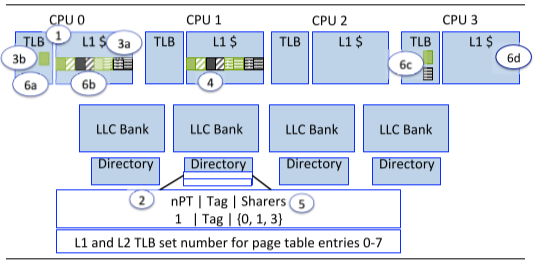
共同标签不仅允许更精确的转换识别，而且还可以搭载到现有的缓存一致性协议上。当虚拟机管理器修改嵌套页表转换时，缓存一致性协议检测对页表项的系统物理地址的修改。通过共同标签，HATRIC扩展了缓存一致性，以前仅限于私有缓存的一致性消息现在也被转发给转换结构。共同标签用于识别哪个TLB、MMU caches、nTLB单元对修改的嵌套页表cache队列作出了响应。

### HATRIC的整体操作

如图4所示，是HATRIC操作的细节。最初，CPU 0的TLB和L1 caches是空的。当内存访问时，在TLB中未访问到CPU 0①。无论何时从L1 cache中的页表队发出的M、E、S状态的请求，都不需要再初始化一致性事务了。但是，假设页表中的最后一个内存引用在L1 cache中不存在。在步骤②就会有一条读取请求中发送到一致性目录。

有两种可能的情况。第一种情况是，转换可能在私有cache中未缓存，并且也没有一致性目录单元，那么就会分配一个目录单元并设置nPT位。第二种情况是（如图9）该请求与现有的目录单元匹配。且nPT位已经设置了，HATRIC读取识别CPU 1和CPU 3的共享列表，同时也缓存处于共享状态的理想的转换。

图4 一致性目录识别缓存了页表单元的转换结构



作为响应，具有理想转换的cache队列被发回到CPU 0（从CPU 1,3或存储器，无论哪个速度最快），更新L1 cache 3a和TLB 3b。随后，共享者列表将添加CPU 0。

现在假设CPU 1运行虚拟机管理程序，并在步骤④中从嵌套页表中取消固定的绿色转换。为了将L1 cache 队列转换为M状态，高速缓存一致性协议将消息中继到一致性目录。步骤⑤识别相应的目录单元。6a是CPU 0无效化它的TLB单元，6b表示L1 cache中有8个转换，6c是CPU 3无效化2个匹配共同标签的TLB单元，6d是由于伪特定，CPU 1的L1 cache收到虚假的无效化信息。

# 思考总结

两篇论文所提出的技术都与地址转换、TLB有关。一篇是在不同内存分配方案下提升TLB转换覆盖面积的硬件软件混合TLB合并技术，该技术提出的目的是为了提升虚拟内存下的地址转换效率。传统的扩大TLB转换覆盖面积的方法在分配灵活性和扩展性方面不可兼得，但是该方案能兼得两者。实验的结果表明该技术只需稍微修改MMU，就可以进行高度合并的地址转换。另一篇是提出了虚拟系统中硬件转换一致性的技术以降低页面重映射的开销，并提高系统的性能。该技术是通过对转换结构（TLB，MMU caches和nTLBs）进行修改，在已有的硬件cache一致性协议上实现了硬件转换一致性技术。实验是在Xen和KVM的虚拟化系统测试的，测试结果表明硬件转换一致性能提升系统性能，降低能耗，且在系统上容易实现，是一种硬件机制。

在地址转换中，快表（TLB）的作用很大，在虚拟地址向物理地址转换寻址时，如果在TLB中命中了，就能大大降低地址转换的开销了。因此可以发现两种技术都在TLB上做了一些修改，混合合并技术就是在页表上设置锚点来记录连续内存信息，并在增加TLB单元的位数变成锚定TLB。而硬件机制中每个TLB有自己的共同标签，共同标签存储着嵌套页表单元的物理地址，而共同标签使得嵌套页表是在硬件层面改变。

两种技术都是改变一方面以达到预想的目的，但是都没有用传统的方法。也就是说他们提出的技术都是创新的。也就是说我们在做实验，思考解决问题的方法时，可以查阅前人的资料并学习他们的方法。但是前人的方法不尽完美，也不一定可信。比如混合合并技术中，前人的方法在分配灵活性和转换覆盖面积的扩展性方面不能兼得，正如“鱼和熊掌不可兼得”的道理一样，但是研究者就想出了硬件软件混合的TLB合并技术，提供了最佳性能。当然以后也许会有更好的方法。

最后是发现问题的能力，对于地址转换这方面的问题，研究者们不仅能考虑到实际操作系统也能考虑到虚拟操作系统中。当然这是与虚拟化有关的话题，研究面广，我们也应该多多思考，发现前人没有发现的问题。

**参考文献**

[1]Zi Yan Ján Veselý Guilherme Cox. Hardware Translation Coherence for Virtualized Systems. ISCA 17, June 24-28, 2017, pp.430-443

[2]Chang Hyun Park Taekyung Heo Jungi Jeong. Hybrid TLB Coalescing: Improving TLB Translation Coverage under Diverse Fragmented Memory Allocations. ISCA 17, June 24-28, 2017, pp.444-456