页迁移系统中反向页表技术的设计与实现*)

杜 静 戴华东 杨学军
(国防科技大学计算机学院 长沙410073)

摘 要 页迁移技术是实现CC-NUMA存储优化的一种重要策略,它动态开发了数据的局部性。页迁移策略的实现涉及到虚存系统中物理地址到虚拟地址的转换,传统做法需要遍历所有进程的虚拟地址空间,效率低、开销大。针对此问题,本文介绍了一种能够高效实现物理地址到虚拟地址转换的技术——反向页表技术,着重介绍了反向页表的设计、实现和维护方法。

关键词 页迁移/复制,页表一致性,时间开销,反向页表

The Design and Implementation of Reverse Page Table in Page Migration System

DU Jing DAI Hua-Dong YANG Xue-Jun
(School of Computer Science, National University of Defense Technology, Changsha 410073)

Abstract Page migration is an important policy to exploit memory optimization in CC-NUMA systems, and it is a way to dynamically exploit data locality. The realization of page migration policy involves translation from physical addresses to virtual addresses in virtual memory area, it needs travel through all the process address space with traditional approaches, which is inefficient and costly. Aiming at this problem, an efficient technology to realize translation from physical addresses to virtual addresses-reverse page table is introduced in this paper. Emphasis is put on the design, realization and maintenance for reverse page table

Keywords Page migration/replication, Consistency of page table, Time overhead, Reverse page table

1 引言

CC-NUMA(Cache Coherent Non-Uniform Memory Access)系统结合了SMP系统的易编程性和分布式存储系统的高可扩展性等优点,受到了用户的欢迎。同传统SMP系统相比,虽然CC-NUMA系统支持对远程存储器中数据的透明访问,但由于系统的远程存储器访问延迟远高于本地存储器访问延迟,因此应用访存局部性的优劣成为严重地影响应用整体性能的关键因素之一。优化应用数据和指令访问的局部性无法仅依靠用户在应用中进行优化,一方面实现困难,同时优化效果有限。基于CC-NUMA计算机系统的硬件支持,在操作系统层通过动态页迁移和页复制技术实现动态的访存优化是目前主要的技术途径。

操作系统的页迁移和页复制技术涉及到页表的管理和TLB的管理。操作系统通过页表实现虚拟地址到物理实址的虚实转换,TLB则是该转换的一种硬件加速手段。操作系统页迁移和页复制过程中对页表和TLB的操作过程如下:当硬件产生页迁移请求时,内核获得要迁移或复制页的物理地址,并进入页迁移和页复制中断处理过程。操作系统在完成页的迁移或复制后,需要修改页表和TLB表项,将原来对应该物理地址的所有虚实映射关系转化为新的映射关系。当应用访问该地址时,通过原先的虚拟地址可转换到新的迁移或复制后的物理地址。

在实现页迁移中,页迁移过程本身的开销十分重要。开销过大将导致访存的长时间停顿,反而加大了访存的延迟,抵消页迁移所带来的数据局部性优点。为了减少页迁移和复制开销,快速定位和修改页表、TLB表项十分重要,因为页迁移和复制中断要根据所获得的页物理地址定位所有使用该地址的页表和TLB表项。传统操作系统仅支持从虚拟地址到物理地址的映射,根据物理地址映射虚拟地址需要遍历所有的页表,包括内核页表,以及所有用户进程的页表,开销巨大。因此,通过物理地址映射虚拟地址的反向页表映射技术在CC-NUMA计算机系统中的页复制和迁移中占有十分重要的地位。

本文结合实际工程需求,介绍了一种能够在操作系统内核中高效实现物理地址到虚拟地址转换的方法——反向页表技术,并基于CC-NUMA体系结构和Linux 2.4.18内核进行了设计和实现。

2 面向页迁移的反向页表设计

2.1 设计思想

反向页表是操作系统中实现快速实虚转换、支持页迁移的关键部件,它的实现有利于提高页迁移的效率。通常,一个迁移的候选物理页可能被多个进程所共享。迁移过程中,需要找到所有映射到该物理页的进程,这将多次涉及虚实地址的相互转换。Linux的页表技术完成了虚存中虚实地址的映射;我们所研究的反向页表技术,弥补了仅仅依靠正向页表实现实虚映射所带来的开销,能够通过迁移页的物理地址快速定位需要修改的进程页表项。

实际上,FreeBSD内核通过对页号的索引,已经实现了空间独立的反向页表结构,保存着体系结构相关的内存映射信息。Linux 2.5.x内核也实现了反向页表结构。但是这两种反向页表的应用目的主要是针对swap过程的优化,没有实现

*)基金项目:国家863服务器操作系统内核(2003AA1Z2026)。杜 静 硕士生,研究方向:并行、分布式操作系统及并行计算机体系结构;戴华东 博士,讲师;杨学军 教授,博士生导师。

对页迁移的支持。参考它们的基本思想，结合页迁移的需求，我们研究并实现了基于 Linux 2.4.18内核和 Intel 的 IA64处理器的反向页表技术，该技术对页迁移的优化提供了有效的支持。

2.2 反向页表在页迁移中的作用

页迁移的设计目标是通过动态地开发数据局部性，来减小远程存储开销。实现页迁移的关键技术之一是控制页迁移本身的开销，否则页迁移所带来的数据局部性好处可能被页迁移本身的开销抵消。

进程执行时，操作系统要为进程的正文和数据分配实际的物理页面，所以，当物理页面移动后，内核需要在全局地址空间内修改虚拟地址到物理地址的映射，并刷新 TLB 表。传统的实虚地址映射方法需要遍历整个虚存空间，使得维护页表一致性的开销成为页迁移的主要开销。为了避免传统方法所带来的维护开销，我们在页迁移实现策略中引入了能快速实现实虚地址转换，有效支持页迁移系统的反向页表技术。该页迁移的实现过程如下：

当页迁移策略被触发后，内核将在迁移目标结点的内存空间中分配一个新页，同时分配该页的反向页表。然后根据被迁移源页的物理地址，查询其反向页表，找到所有保存老的虚实转换的 PTE 项，该 PTE 项将映射到迁移后的新页。接着将新页插入到合适的数据结构中，修改旧页所映射的进程页表入口项，并刷新所有处理机的 TLB 表，对新页的反向页表进行维护。最后将数据拷贝到新页，并释放旧页，删除其反向页表。过程如图1所示。

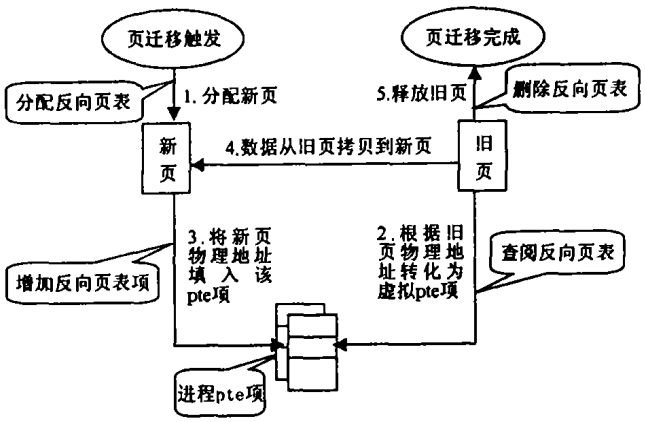

图1 页迁移技术中反向页表的应用

反向页表在页迁移中的作用是根据中断给出的物理地址，迅速定位到进程 PTE 表项的虚地址。从而在页迁移过程中能够快速将迁移后新页的物理地址填入该 PTE 表项。当多个进程共享迁移页时，通过查找反向页表能够同时定位多个进程的页表项，同传统页迁移技术相比，不需遍历整个进程地址空间，有效地降低了迁移开销。

3 反向页表的实现

结合工程需求，我们的具体实现基于 Intel 的 IA64处理器和 Linux 2.4.18内核。IA64是 Intel 的64位多处理器体系结构。它克服了传统体系结构的许多限制，并提供了极大的可扩展空间。通过推断、预测、显式指令并行等技术，IA64较好地实现了指令级并行性，并且64位存储器地址为高性能服务器提供了巨大的存储空间。而 Linux 2.4.18内核加上 NUMA Patch 已经较好地实现了不连续内存设计、NUMA 系统的结点间处理器的设计，有利于相关工作的展开。

3.1 反向页表数据结构设计

反向页表技术的主要设计思想是，在传统页表基础上，为每个物理页增加一个反向指针集合，指向使用该物理页的各个页表项，形成物理地址到虚地址的反向映射。如图2所示，其中虚线部分表示反向页表结构。

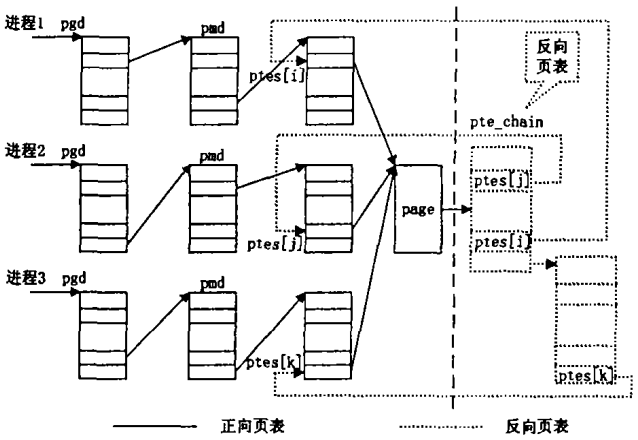

图2 反向页表及其映射关系

(1)对 page 数据结构进行修改

```
typedef struct page {
    ……
    union {
        struct pte_chain * chain;
        pte_addr_t direct;
    } pte;
    ……
}
```

当虚地址到物理地址的映射关系是一对一时，PTE 为一个直接(direct)指针，直接指向使用本物理地址的唯一 PTE。

当虚地址到物理地址的映射关系是多对一时，我们通过 pte_chain 结构所构成的链表 chain 来记录所有使用本物理地址的 PTE。

(2)实现 pte_chain 数据结构

·pte_chain 是一个多个数组链接而成的链表，数组中的每一项对应指向映射到该物理页的一个 PTE 项地址，并返回一个指向该页的 PTE 集合的链表结构。其数据结构如下：

```
struct pte_chain {
        unsigned long next_and_idx;
        pte_addr_t ptes[NRPTE];
}
```

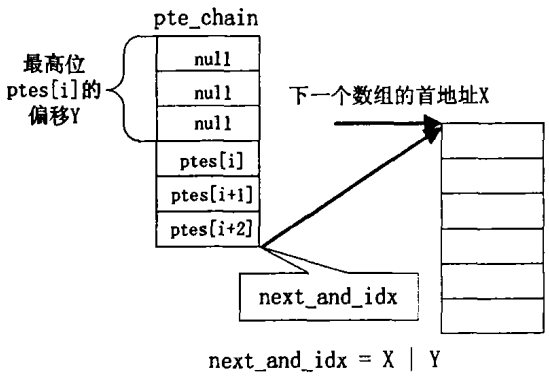

图3 反向页表结构

ptes[NRPTE]数组的元素个数 NRPTE 由系统一级 cache 大小决定。其尺寸满足 pte_chain 结构的 cache 边界对齐需求。当共享该物理页的 PTE 数目大于 NRPTE 时，扩展生成新的 pte_chain 结构。通过与 cache line 的边界对齐使 struct pte_chain 数据结构可以一次加载到一个 cache line

中，大大提高存储访问效率。为了提高 PTE 空间使用的合理性，next_and_idx 指针除包含下一个 pte_chain 结构的地址信息外，还包含本 pte_chain 结构中 ptes 数组空闲项索引信息，如图3所示。在分配和回收 ptes 数组项时，保证 pte 紧致存放，使用空闲索引来界定空闲项和占用项，以便于快速查找。

3.2 反向页表的维护

所述反向页表的维护技术是指为了与系统正向页表保持一致性而对反向页表进行建立和动态增删的维护过程。在系统运行过程中，正向页表将不断改变虚实地址的映射关系，反向页表是维护实虚地址映射关系的页表，故反向页表将随正向页表的变化而动态改变。反向页表的维护时机是在任何一个 PTE 中的物理地址发生改变时，此时反向页表都需要进行修改，以正确反映当前的实虚映射关系。

反向页表的维护点，即正向页表项(PTE)变化点主要发生在页失效中断、页迁移中断和换页和交换过程中。具体如下：

3.2.1 页失效中断

当逻辑页面没有对应的物理页面时，系统会进入页失效中断处理。页失效中断处理例程将为该逻辑页面准备合适的物理页面，并将该物理页面的地址填写到 PTE 中。页失效发生的情况有以下几种：

1) 写时拷贝(COW)

写时拷贝的基本思想是将一个物理页面(旧页)标记为只读，而将其所包含的虚地址访问位(VMA)标识为可写。任何对页面的写操作都会与页级保护相冲突，然后触发一个页失效。当页失效处理程序注意到页级保护和虚地址访问位(VMA)的保护不一致时，它将创建该页的一个可写拷贝(新页)作为替代页。此时需要修改写进程的页表和新旧物理页的反向页表，删除旧页反向页表中有关写进程的项，并为新页分配反向页表，在该表中填加写进程中指向它的页表项地址。

此时，我们使用反向页表建立例程为物理页建立反向页表数据结构。

2) 文件页失效

文件页是指映射到磁盘文件的虚拟页，当对该类虚拟页进行访问发生失效时，页失效处理例程负责分配一个新物理页，将所缺页内容从磁盘载入内存物理页中，而后将物理页地址添写到虚拟页的 PTE 表项中。

此时，我们使用反向页表建立例程为物理页建立反向页表数据结构。

3) 匿名页失效

匿名页是指供进程堆栈使用的页，当对该类虚拟页进行首次访问时，会发生页失效。页失效处理例程负责为其分配新页，并初始化为全零，而后将物理页地址添写到虚拟页的 PTE 表项中。

此时，我们使用反向页表建立例程为物理页建立反向页表数据结构。

4)交换页失效

交换页是指被暂时交换到交换设备上，不在内存物理页中的虚拟页。页失效处理例程分配新的物理页，并将缺页的内容从交换设备上换入到新分配的物理页中，而后将物理页地址添写到虚拟页的 PTE 表项中。

此时，我们使用反向页表建立例程为物理页建立反向页表数据结构。

上述四种情况中都存在分配新物理页的情况，我们这些物理页创建反向页表结构，并填写正确的 PTE。

3.2.2 页迁移和复制

页迁移和页复制过程中，根据中断给出的需要迁移的物理页地址，通过查阅反向页表，可以快速、准确地定位所有的 PTE。

页迁移过程中，需要释放旧页，删除其反向页表，并为迁移后的新页重新分配反向页表，且把所有映射到该物理页的页表项虚地址写入反向页表中。我们将旧物理页中的反向页表结构赋予给新物理页，旧物理页指向该反向页表结构的指针随物理页的释放而删除。

页复制过程中，需要产生原物理页的拷贝页，此时一个多对一映射分裂为两个多对一映射，每个物理页的反向页表，都应仅包含实际访问它的 PTE。此时根据新生成的 pte 调用反向页表建立例程为新物理页建立反向页表结构，调用反向页表删除例程删除旧物理页中对应的 pte。

3.2.3 交换和换页

为了提高内存的有效利用率，操作系统通过换页和交换机制将暂时不用的进程页写到交换设备，释放占用的物理页。换页和交换机制通过系统中运行的守护进程 swapper 实现。在守护进程将暂时不用的进程页写到交换设备，释放占用的物理页时，反向页表随物理页的释放而进行删除操作。

在上述反向页表维护点(一、二、三)，根据情况调用反向页表的建立和删除例程，完成维护工作。针对页迁移策略进行测试，结果表明，反向页表能有效地支持页迁移工作。

4 性能分析与改进

4.1 性能分析

反向页表提供了进程页表的反向映射。也就是说，它能够指出哪个进程用到了该物理页，及其对应的虚地址。通过运用反向页表，与传统方法相比，页迁移的实现存在以下优势：

· 因页迁移而引起对页表项的修改时，不需要遍历所有进程的虚存，仅仅搜索页面的反向页表。这意味着页迁移代码仅仅做很少的内存搜索工作。对于频繁的迁移复制来说，减少了页迁移/复制开销。

· 当多个进程共享同一物理页时，可以迅速通过物理页的地址找到所有的 PTE 项。

· 当负载很大时，内存管理可以花费较少的 CPU 时间，提高页迁移/复制的效率。

这种方法的不足之处是存在链表存储空间的开销。系统中的每一个物理页结构都要维护一个额外的链表结构 pte_chain。一个内存大小为256MB 的系统中物理页的数目为64k，则需要为反向页表分配64×(sizeof (struct pte_chain))kB 大小的物理内存，与页表所占据的空间相比，这个数目比较可观。而且为反向页表分配内存空间时，可能存在溢出或越界的问题。

尽管如此，经过测试，在高端系统和负载很大的情况下，反向页表支持的页迁移系统性能明显优于传统页迁移系统。

4.2 技术改进

在页迁移中增加反向页表的支持后，使得页迁移开销减小。但是反向页表长期占用了大量内存，降低了一些需要作用于大量页面的操作的效率。所以，需要改进技术来降低反向页表的开销。

Linux 系统中一些页面来自文件映射，包含程序代码数据和 mmap( )所映射的文件。这些页面存在另外指向页表项

的路径，如果这些路径可以被使用，那么反向页表的开销则可以避免，如图4所示。

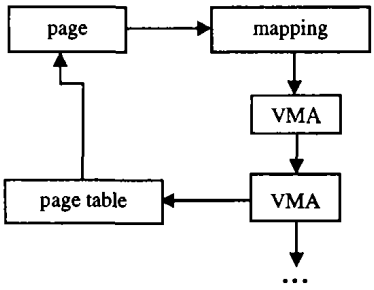

图4 文件映射页面的一种反向映射路径

page 结构中有一项指向 address_space 结构的域 mapping，该域描述了备份这个页面的文件。address_space 结构包括备份文件的 inode、文件页面的数据结构和两个 vm_area_struct(VMA)的链表。VMA 链表说明了特殊进程地址空间的映射关系。文件/proc/pid/maps 列出了不同 PID 号进程的 VMA 映射。当我们需要获得非 anonymous 页面的虚地址时，可以通过 address_space 和 VMA 结构找到对应的页表项。

虽然这种方法比直接查找反向页表要花费更多的时间，但是，因为可以不用维护非 anonymous 页面的反向页表结构，所以能够节省大量空间开销。我们考虑在今后的实现中将两种反向映射方法结合起来，并在不同的情况下使用不同的地址转换方法来获得更高的页迁移效率。

结束语 页迁移技术是实现 CC-NUMA 系统存储优化的重要方法，它动态地解决了数据局部性问题。由于迁移过程中频繁涉及到实虚地址的转换，其开销影响了页迁移的效率。以优化页迁移技术为出发点，本文提出了操作系统中实现快速实虚转换、支持页迁移的关键部件——反向页表技术。经过测试，在高端系统和负载很大的情况下，反向页表支持的页迁移系统性能明显优于传统页迁移系统。

当然，目前这种技术还面临着许多问题，主要体现在反向页表的空间开销上，而且 fork( )系统调用将会对进程地址空间中的每个页面增加一个新的反向页表项，大大增加了反向页表的维护开销。这些都是今后需要研究和解决的内容。

参 考 文 献

1 Verghese B, Devine S, Gupta A, Rosenblum M. Operating system Support for Improving Data Locality on CC-NUMA Computer Servers. In: proc. Architectural Support for Programming Languages and Operating Systems, 1996. 279～289
2 Nikolopoulos D, et al. Scheduler-Activated Dynamic Page Migration for Multiprogrammed DSM Multiprocessors. In Journal of Parallel and Distributed Computing, 2002
3 Laudon J, Lenoski D. The SGI Origin: A ccNUMA Highly Scalable Server[A]. In: Proc. of the 24th Annual Int'l Symp on Computer Architecture[C]. 1997
4 van Riel R. Towards an O(1) VM: Making Linux virtual memory management scale towards large amounts of physical memory. In Linux Symposium 2003
5 LWN The object-based reverse-mapping VM. http://lwn.net/Articles/23732/?format=printable

(上接第209页)

$\frac{1-0.6561}{1-0.9945327825}\approx62.9$倍；而使用完全复制的方法，只能将单点容错性能和整体容错性能分别提高$\frac{1-0.9}{1-0.99}=10$倍和$\frac{1-0.6561}{1-0.9509900499}\approx7.01$倍。由此可见，在增加同样的存储空间的情况下，环形 DCR²S 可以比完全复制更加有效地增加一组文件的单点和整体容错性能。

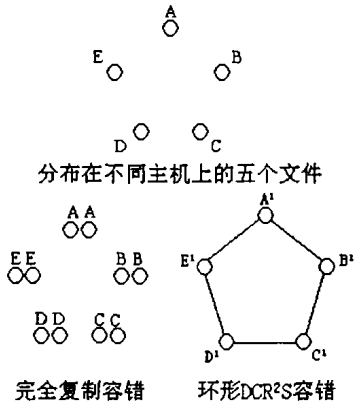

图6 环形 DCR²S 容错和完全复制容错

以上的分析是基于 $k=1$ 的假设，即环形 DCR²S 单元中每个点对应的文件复本只有一个。当 $k>1$ 时，根据公式计算可知环形 DCR²S 结构的容错性能仍然高于完全复制，且提高的幅度比 $k=1$ 时更大。由此可知，一组分散的文件在结成环形 DCR²S 单元后，若还想进一步提高单个文件和整体的容错性能，只需为每个文件增加复本，就可以获得比占用同样存储空间的完全复制方法更好的容错性能。

结论和未来的工作 本文提出了一种新的分布式协作冗余复制存储机制 DCR²S，这种机制通过校验文件实现了多个文件之间的相互协作，不仅可以提高单个文件的容错性能，更可以使得一组文件的整体容错性能得到大大提高。本文用图论的概念准确地描述了 DCR²S 的容错性质，定量地分析了链形和环形 DCR²S 结构的容错性能，并给出了这两种特殊结构的单点和整体可得概率的计算公式。通过计算比较，环形 DCR²S 结构在占用同样存储空间的条件下，可以实现比完全复制更高的容错性能，证明了 DCR²S 是一种十分有效的分布式文件容错机制。

如前所述，DCR²S 有多种结构，而本文只分析了链形和环形两种，在以后的工作中将对其他一些 DCR²S 结构的容错性能进行进一步的分析，找出容错效率最高、最易于实现的 DCR²S 单元结构，并通过实验给予验证。

参 考 文 献

1 Dingledine R, Freedman M, Molnar D. The freehaven project: Distributed anonymous storage service. In: Proc. of the Workshop on Design Issues in Anonymity and Unobservability, July 2000
2 Bolosky W, Douceur J, Ely D. Theimer M. Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. In: Proc. of Sigmetrics, June 2000
3 Kubiatowicz J. et al. Oceanstore: An architecture for global-scale persistent storage. In: Proc. of ASPLOS, Nov. 2000, ACM
4 魏青松，卢显良，雷宇. FTDSS：高容错分布式共享存储机制. 计算机科学，2003，30(8)：172～175
5 Druschel P, Rowstron A. Storage management and caching in PAST, a large-scale, persistent peerto-peer storage utility. In: Proc. of ACM SOSP (2001)
6 Blomer J, Kalfane M, Karp R, Karpinski M, Luby M, Zuckerman D. An xor-based erasure-resilient coding scheme. Technical report, Intl. Computer Science Institute, Berkeley, California, 1995
7 Weatherspoon H, Kubiatowicz J D, E C vs. Replication: A Quantitative Comparison, IPTPS '02