Linux 2.6 内存管理研究*

谢长生，刘志斌
(华中科技大学 计算机科学系，湖北 武汉 430074)

摘 要：Linux 得到越来越广泛的应用，Linux 2.6 作为最新的内核在各方面都有很大的改进。针对 Linux 内存管理系统，介绍了 2.6 版所采用的新技术，同时提出了改进的途径。

关键词：Linux 2.6；内存管理
中图法分类号：TP316 **文献标识码**：A **文章编号**：1001-3695(2005)03-0058-03

Research on Linux Memory Management

XIE Chang-sheng, LIU Zhi-bin
(Dept. of Computer Science, Huazhong University of Science & Technology, Wuhan Hubei 430074, China)

Abstract: Linux is widely used nowadays. Linux 2.6 has made many progresses as the newest kernel. The article aims for memory management, points out the new technologies have been adopted by the 2.6 version, and expresses some improving methods at the same time.

Key words: Linux 2.6; Memory Management

1 Linux 发展概述

Linux 已经是一个以往操作系统的实用的替代品，在市场上表现出了强大的竞争力。越来越多的政府机构和 IT 巨头的注意力正在转向 Linux。从嵌入式设备到服务器，Linux 现在几乎可以用于所有的地方。Linux 在中国也发展得如火如荼，前不久中日韩共同签署《开放源代码合作备忘录》，将合作致力于形成共同认可的 Linux 标准以实现信息交流与研究成果的共享。

Linux 内核始于 1991 年由 Linus Torvalds 为他的 386 开发的一个类 Minix 的操作系统。Linux 1.0 的官方版发行于 1994 年 3 月，包含了 386 的官方支持，仅支持单 CPU 系统。Linux 1.2 发行于 1995 年 3 月，它是第一个包含多平台支持的官方版本。Linux 2.0 发行于 1996 年 6 月，它是第一个支持 SMP 体系的内核版本。Linux 2.2 在 1999 年 1 月到来，它带来了 SMP 系统上性能的极大提升，同时支持更多的硬件。Linux 2.4 于 2001 年 1 月发布，它进一步地提升了 SMP 系统的扩展性，同时它也集成了很多用于支持桌面系统的特性。Linux 2.6 于 2003 年年底发布，无论是对于企业服务器还是对于嵌入式系统，Linux 2.6 都是一个巨大的进步。对高端的机器来说，新特性针对的是性能改进、可扩展性、吞吐率，以及对 SMP 机器 NUMA 的支持。对于嵌入式领域，添加了新的体系结构和处理器类型——包括对那些没有硬件控制的内存管理方案的 MMU-less 系统的支持。并且与往常一样，为了满足桌面用户群的需要，添加了一整套新的音频和多媒体驱动程序。

Linux 2.6 在进程调度、内存管理、文件系统、网络、系统安全、设备驱动等方面都有很大的改进，本文将集中分析 Linux 2.6 的内存管理。内存管理与 CPU 的体系结构密切相关，本文以 Intel 的 i386 系列 CPU 作为目标进行描述。

收稿日期：2004-05-08；修返日期：2004-07-25
基金项目：国家自然科学基金资助项目(69773074)

2 Linux 内存管理的基本框架

2.1 i386CPU 内存管理的硬件设施

i386 系列 CPU 发展初期采用段式管理，出于兼容性的要求，段式管理被延续下来。CPU 内部设置了 CS, DS 等段寄存器，以及段描述表寄存器 GDTR(全局)和 LDTR(局部)，两种寄存器中内容的组合找到段描述项，段描述项描述了段的地址、大小以及状态和权限等一些信息。随着计算机技术的发展，i386CPU 实现了页式管理，段内分页，用页面表寻址，为此增加了页面目录的基地址寄存器 CR3。

2.2 Linux 对虚拟地址的组织

Linux 程序使用的是 32 位虚拟地址，编程时无须考虑物理地址，虚拟到物理地址的转换由映射机制完成。32 位地址可以形成 4GB 的虚拟空间，Linux 将 4GB 空间分成两部分，最高 1GB 为内核空间，低 3GB 为用户空间，内核空间为不同用户共享，虚拟内存的组织如图 1 所示。

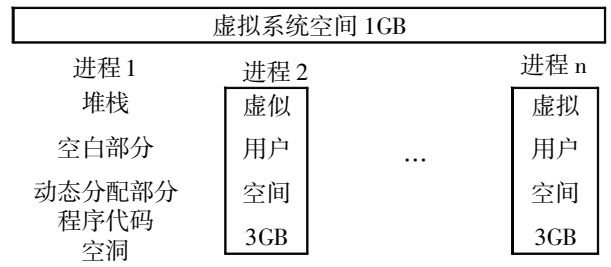

图 1 Linux 虚拟空间组织

每一个进程包括一个 mm_struct 结构，指向 vm_area_struct 结构的链表，表示不同的虚存段，整个链表就表示进程使用的所有虚存。出于效率的考虑，vm_area_struct 同时保持 AVL 树的结构。

2.3 Linux 对物理空间的组织

物理空间包括两部分，即物理内存空间和磁盘空间，均按

页组织。内存页面用 Page 结构描述, 所有页面形成 mem_map 数组。磁盘页面比较简单, 只是说明被几个进程使用, 被组织成文件的形式, 用 swap_info_struct 描述。

内存空间被分成不同的管理区, 对于 i386CPU 而言, 0MB ～16MB 定义为 ZONE_DMA, 16MB ～896MB 为 ZONE_NORMAL, 如果有多于 896MB 的内存则被定义为 ZONE_HIMEM。

内存页是物理内存的基本组织, 部分定义如下:

```
struct page {
struct list_head list;
struct address_space* mapping;
      …
atomic_t count;
      …
struct list_head lru;
…
}
```

这几个元素都与内存页面的组织有关。对于 List 来说, 如果页面空闲(Count 等于 0) , 则 List 用来链入所属管理区的多个 free_area_struct 队列中的一个, 不同队列的空闲块大小不同, 不过均为 2 的整数次幂个页面; 如果页面不空闲(Count 不为 0) , 则 List 或者为空(页面未建立盘上映射) , 或者用来链入 Mapping 中三个页面队列 clean_pages, dirty_pages, locked_pages 中的一个(分别表示未被改写页面、被改写页面和锁定页面) 。对于 Lru 来说, 如果页面活跃, 则 Lru 用来链入全局的活跃队列 active_list; 如果不活跃, 则 lru 用来链入全局的 inactive_dirty_list 或者所属管理区的 inactive_clean_list。

对于内核缓冲区来说, 按页分配不是很好的方式。Linux 采用了 Slab 管理方式, 一个 Slab 可以由 1, 2, 4… 最多 32 个连续的物理页面组成, 一个 Slab 块可以包含多个对象。

2.4　Linux 的内存映射和管理机制

程序的虚拟空间必须映射到物理空间才能运行。在 i386CPU 上, 首先要进行段式映射, Linux 没有用到段式管理, 它的做法是把 GDT 中段描述符段的大小定义为 4GB, 也就是说只分了一段, 从而使段式映射没有作用。

接着进行页式映射, 从 CR3 中取出页面目录的指针, 通过查表获得物理地址。Linux 的虚拟页面映射模型如图 2 所示。

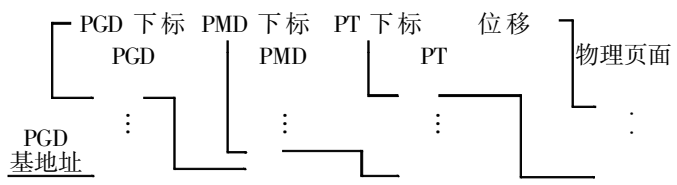

图 2　Linux 的虚拟页面映射模型

具体到 i386 来说, 实际上是两层映射, 跳过中间的 PMD 层次。对于程序来说, 并非所有虚存都映射到物理空间, 而是动态映射, 如果程序运行时内核发现虚拟页面没有映射或映射的是磁盘页面, 会作相应的处理——分配内存页面并建立映射, 然后恢复程序运行。

在程序运行的过程中, 涉及到的内存操作主要有内存分配、内存使用、内存回收、内存页面换出、页面换入。内存分配会在管理区的空闲区进行, 通过 Buddy 算法在管理区的 free_area 中获得需要的内存块。如果内存不足, 则会启动 Kswapd 这个守护进程腾出部分物理内存。除了被调用, Kswapd 进程还会定时启动。Kswapd 的工作分两部分:

(1) 检测物理内存剩余的情况, 如果短缺, 则按 LRU 策略断开 active_list 队列中部分可交换页面的映射, 使页面变为不活跃状态, 链入 inactive_clean_list 队列或者 inactive_dirty_list 队列, 为换出做准备。

(2) 每次都执行, 把 inactive_dirty_list 中的页面写入交换设备, 并且回收一部分 inactive_clean_list 中的页面。

3　Linux 2.6 内存管理方面的新技术

Linux 2.6 在内存管理方面采用了很多新的技术, 特别是针对多处理器大内存机器方面, 下面介绍一些主要的改变。

3.1　反向映射

反向映射技术为每一个内存页面的 Page 结构增加了如下内容:

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

Pte 为一个链表或者指针, 包含了指向该页的每一个进程的页表条目(Page-Table Entries, PTE) 的指针。

反向映射的采用给页面换出带来的好处很明显: 不需要遍历每一个进程的页表来判断本页是否被这个进程使用; 可以优先换出希望换出的内存页面; 只需要扫描不活跃的页面。

不过反向映射也存在一些缺点, 首先是内存空间上的开销, Linux 2.6 采用两种方法减少空间开销: ①如果某个页面只是由一个唯一的进程映射, 就不用链表, 而设置 Direct; ②链表采用 Cacheline 分组, 一个 Pte 链表的节点可以同时指向多个节点, 这样不仅减少了内存使用, 而且有利于数据的局部性。其次是节点的 Next 指针采用 32 位, 所以不能使用高端内存, 对于服务器的应用不利。最后还有时间上的开销, 使用反向映射需要加锁, 时间消耗增多, 特别是对于那些执行很多派生和退出的应用程序。

尽管有一些折中, 但可以证明反向映射是对 Linux 内存管理器的一个颇有价值的修改。通过这一途径, 查找定位映射某个页的进程这一严重瓶颈被最小化为只需要一个简单的操作。当大型应用程序向内核请求大量内存和多个进程共享内存时, 反向映射帮助系统继续有效地运行和扩展。

3.2　大内存页和高端存储页表

以前在 i386CPU 上内存页大小为 4 KB, 对大部分用途来说, 内存管理器以这样大小的页来管理内存是最有效的。不过, 有一些应用程序要使用特别多的内存(如大型数据库) , 如果又同时存在很多进程, 在这种情况下页表占的空间不可忽视, 甚至可能超过应用程序请求使用的内存数量。

Linux 2.6 在这种情况下采用的办法是大内存页, i386CPU 支持 4MB 的大内存页。采用大内存页可以显著减少页表的大小, 同时还可以通过减少变换索引缓冲(Translation Lookaside Buffer, TLB) 的失败次数来提高性能, 因为内存页越大, 就有越多的内存可以通过 TLB 引用。

大内存页可以缩小页表, 不过如果进程很多的话还是会过多占用低端内存, 而内核又不能直接使用高端内存。所以在 2.6 内核中有一个配置选项称为 Highmem PTE, 让页表条目可以存放在高端内存中, 释放出更多的低端内存区域给那些必须放在这里的其他内核数据结构。作为代价, 使用这些页表条目的进程会稍微慢一些。不过, 对于那些在大量进程中运行的系统来说, 将页表存储到高端内存中可以在低端内存区域挤出更多的内存。

3.3 热页面和冷页面

热页面和冷页面是一项提高数据缓冲效率的技术。这项技术对于每一个 CPU 的每一个管理区,都会提供两个数据页队列:热队列(Hot Queue) 和冷队列(Cold Queue) 。这两个队列只保存单个页面,多个数据页的管理仍然通过 Buddy 系统。

使用热页面和冷页面来收集和释放内存会提供比 Buddy 系统更高的效率,可以在一个锁定的条件下对多个数据页进行操作,默认的情况是一次 16 个页面。队列会存在一个高水位和一个低水位,如果页面数少于低水位,队列会被重新充满,如果高于高水位,队列会被清空,对于热队列来说,值为 32 ~96,而冷队列为 0 ~32。

热队列的管理方式是后进先出,它收集的是通过 free_pages( ) 释放的内存页。冷队列收集的是 CPU 请求可是没有被马上使用的页面(如 DMA 操作一次会读入一些页面) ,它保留了热页面中有价值的页面以及一个缓冲行中暂时没有使用的页面,这些页面来自 shrink_list( ) , shrink_cache( ) 和 refill_inactive_zone( ) 释放的页面。

据统计采用热页面和冷页面技术可以使内存访问的平均周期数降低 17. 5% 。

3.4 分节点的页面管理

Linux 2.6 中页面队列从全局变成从属于某个节点的管理区,这样有利于释放特定管理区的内存,不需要使用全局锁,而且减少了 NUMA 机器节点间缓冲。内存收集也被设计成可以针对某个节点。

与此相对应,如果内存回收守护进程 Kswapd 仍然是全局处理的,处理的数据可能会非常多,从而对性能产生非常不利的影响。所以 2.6 中的这个守护进程设计成分节点,只需要扫描自己的节点,这样大大提高了效率。

3.5 UKVA 技术

在多 CPU 系统中,内存很大,意味着大部分内存被当作高端内存,由于内核只能通过映射的方式访问高端内存,会影响性能。UKVA(User-Kernel Virtual Addressing) 技术将每个 CPU 的虚拟空间映射到不同的物理内存,这一点很像用户空间的分配(U) ,不过这些区域作为内核空间保护(K) ,这样可以提高效率。

3.6 小结

Linux 2.6 内核中内存管理的改进远远不只本文中提到的这些特性,很多变化是细微的,却相当重要。在性能提高的基础上, Linux 2.6 内存管理经过测试表明也具有很好的稳定性。

4 Linux 内存管理改进研究

前面我们分析了 Linux 2.6 内存管理的一些新技术,不过 Linux 的内存管理还在不停地发展中,下面提出一些可能的改进方法。

4.1 部分页面回收

现有的页面回收机制在小内存的机器上工作状况良好,但是对于大内存机器的支持不是很完善。大内存机器意味着各个页面队列都将很大,如果在页面回收过程中仍然扫描所有的非空闲页面,那么时间的消耗可能会难以忍受。在这种情况下可以采用的策略有在淘汰页面和写出 Dirty 页面时都只扫描部分页面。

对于页面淘汰过程,我们可以建立多个 active_list 队列,每个队列具有不同的活跃状态,页面被访问得越多,就进入活跃状态越高的队列,每次淘汰过程只扫描最不活跃的队列。

对于淘汰页面的写入过程,可以多增加一个缓存队列,例如可以称为 inactive_laundry_list,换出的 Dirty page 先进入 inactive_dirty_list,不过并不马上写入交换设备,当这个队列满了之后,会将队头的页面移到 inactive_laundry_list 队列,进入这个队列的页面才会被写入交换设备,然后进入 inactive_clean_list 等待回收。

采用这样的策略可以使页面回收的过程加快,不至于造成很大的延时。

4.2 保持缓存和代码的比例协调

在活跃的页面中,一部分被用作文件缓存,另一部分被用于进程的代码。在一些情况下,文件缓存占用的空间可能过多,而实际上文件缓冲的利用率并不如代码利用率高,这样会造成性能的降低。

为了避免这种情况,我们可以在这两部分空间之间设定一个比例,运行中保持比例的协调。

在实现中,我们可以把 active_list 按内容分成两个队列,即缓存队列和代码队列。页面淘汰时会检查这两个队列的比例,如果比例高于某一个值则只从缓存队列淘汰,低于某一值就只从代码队列淘汰,两个值之间则两个队列都有页面被淘汰。

4.3 基于对象的反向映射

Linux 2.6 采用的反向映射是基于页面的,这种方法对于页面换出可以获得很高的效率,但是它同时会对一些过程产生影响,如页面错异常、进程复制退出等。对于大于一个页面的对象(如 VMA) 在这些过程中会降低性能,这时如果采用基于对象的反向映射,会获得比较好的性能。不过基于对象的反向映射同时存在缺点,如有的页面不属于对象,对这样的页面处理会比较麻烦。

基于页面和对象的反向映射都存在各自的优缺点,适用于不同的情况。理想的方式是将两者结合起来,相信这也是未来发展的方向。

5 总结

Linux 2.6 的内存管理部分做了很大的改进,在性能和稳定性方面都表现得不错,对高端机器和嵌入式同时提供了更好的支持。发展的同时, Linux 的使用范围也在不断地扩大,需求带动发展,如何设计更好的内存管理系统未来仍然是一个挑战。

参考文献:

[1] 毛德操, 胡希明. Linux 内核源代码情景分析[M] . 杭州: 浙江大学出版社, 2001.

[2] Martin J Bligh, David Hansen. Linux Memory Management on Larger Machines[EB/ OL] . http: / /archive. Linuxsymposium. org/ols2003/ Proceedings/All-Reprints/ Reprint-Bligh-OLS2003. pdf, 2003-07.

[3] Rik Van Riel. Towards an O (1) VM[EB/ OL] . http: / /www. surriel. com/lectures/Ols2003/, 2003.

作者简介:

谢长生(1957-) , 男, 湖北武汉人, 教授, 博士生导师, 主要研究方向为新型计算机外存储体系结构、网络海量存储技术、网络多媒体技术; 刘志斌(1977-) , 男, 湖北宜昌人, 硕士研究生, 主要研究方向为计算机系统结构。