文件 Cache 自适应策略研究

高薇姣,蒋泽军,王丽芳
GAO Wei-jiao,JIANG Ze-jun,WANG Li-fang

西北工业大学 计算机学院,西安 710129
College of Computer Science,Northwestern Polytechnical University,Xi'an 710129,China
E-mail:gaoweijiao@gmail.com

GAO Wei-jiao,JIANG Ze-jun,WANG Li-fang.Study on Page Cache self-adjusting scheme.Computer Engineering and Applications,2009,45(24):67-69.

Abstract: The read and write performances of Linux vary while accessed by different file record sizes.The read and write performances of Linux are low at certain accessing file record sizes.The algorithm in Page Cache is one of the causes of this problem.Based on the analysis of the influence of different accessing file record sizes on the performance of Page Cache,this paper developes an efficient self-adjusting scheme to improve read and write performances of Linux.The scheme considers prefetching policy's influence on replacement algorithm and increases replacement algorithm's adaptability to vary accessing file record sizes.Experimentally,the scheme is efficient to improve read and write performances of Linux.

Key words: Linux;Page Cache;replacement algorithm;prefetching

摘 要: Linux 系统在被不同大小的数据块访问时,系统读写性能有差异。在少数特定访问数据块大小的应用中,Linux 系统读写性能较差。文件 Cache 算法的性能是导致该问题的原因之一。在分析访问数据块大小对文件 Cache 算法性能的影响的基础上,提出了一种文件 Cache 自适应策略。该策略考虑了预取算法对于页面置换算法的影响,增强了页面置换算法对访问数据块大小变化的适应性,达到了提高 Linux 系统读写性能的目标。Linux 系统读写性能测试实验表明,该策略可以使 Linux 系统在被不同大小的数据块访问时保持稳定且更优的读写性能。

关键词: Linux;文件 Cache;页面置换算法;预取算法
DOI:10.3778/j.issn.1002-8331.2009.24.021 **文章编号:**1002-8331(2009)24-0067-03 **文献标识码:**A **中图分类号:**TP303

1 引言

Linux 系统中的文件 Cache 基于局部性原理，解决了 CPU 与硬盘速度差异的问题,提高了计算机系统的性能。但是,因为文件 Cache 的性能与访问数据块的大小有关,所以在访问数据块偏大或偏小的情况下,整个计算机系统的读写性能较差。因此,若要提高 Linux 系统读写性能,必须尽量降低数据块大小对文件 Cache 的性能影响。文件 Cache 的性能与其相关算法的性能有直接的关系,所以,优化文件 Cache 的相关算法是提高 Linux 系统读写性能的途径之一。

文件 Cache 中涉及的主要算法为页面置换算法和预取算法。页面置换算法将最不需要的页面置换出文件 Cache,以保证文件 Cache 中缓存的是最近且最常用的页面。预取算法预测将来的数据请求,将其提前取入文件 Cache,以保证文件 Cache 中缓存了将要存取的数据。该文系统地分析了 Linux 系统文件 Cache 页面置换算法的性能与数据块大小的关系,结合预取算法对于页面置换算法的影响，提出一种文件 Cache 自适应策略。该策略使得 Linux 系统在各种大小数据块访问的情况下,均能发挥较优的性能。实验证明,该策略可以提高 Linux 系统的读写性能。

2 文件 Cache 的性能与数据块大小的关系

文件 Cache 的性能衡量标准是 Cache 系统加速比。假设 Cache 的访问周期为 Tc,主存储器的访问周期为 Tm,Cache 的命中率为 H。根据文献[1],Cache 系统的加速比 Sp 满足以下公式:

$$Sp=\frac{1}{(1-H)+H\times\frac{Tc}{Tm}} \quad (1)$$

由公式(1)可知,Cache 的访问周期 Tc 和主存储器的访问周期 Tm 均受到所用器件的限制,因此,提高 Cache 系统的加速比 Sp 的最好方法是提高命中率 H。命中率 H 主要与 Cache 容量、Cache 页面置换算法和 Cache 预取算法等因素有关[1]。

在 Cache 容量一定的时候,Cache 命中率与数据块大小的关系如图 1 所示,当数据块偏小的时候,命中率 H 很低,随着

基金项目:国家科技支撑计划(the National Scientific and Technical Supporting Programs of China under Grant No.2007BAH08B08)。
作者简介:高薇姣(1984-),女,硕士,主要研究领域为安全存储,嵌入式系统等;蒋泽军(1964-),男,教授,主要研究领域为安全存储、嵌入式系统与网络安全等;王丽芳(1965-),女,教授,主要研究领域为网络安全与电子商务等。
收稿日期:2008-10-15 **修回日期:**2008-12-22

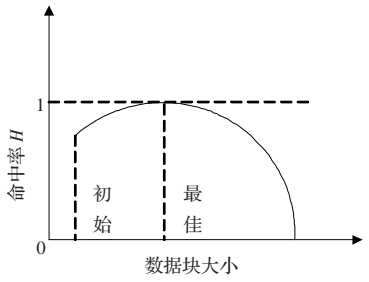

图 1 Cache 命中率与数据块大小的关系

数据块大小的增加,Cache 命中率增加，这种增加趋势在某个数据块大小处达到最大值。此后,命中率随着块大小的增加反而减小。

在 Cache 容量一定的时候,命中率主要和 Cache 页面置换算法和 Cache 预取算法有关。因此,命中率与数据块大小的关系是由 Cache 页面置换算法和 Cache 预取算法的性能共同决定的。在数据块偏小的时候,Cache 的页面置换等待时间长同时预取效果不大,因此系统访问时间长,Cache 命中率低;当数据块偏大的时候,系统可能预取进大量无用数据,同时 Cache 的页面置换速度低于数据块更新速度,Cache 命中率也随之降低。

综上所述，提高 Cache 命中率可以提高系统的读写性能,而通过合理优化 Cache 页面置换算法和 Cache 预取算法,可以使 Cache 命中率在各种数据应用中都能尽量接近最佳值。

3 Linux 内核文件 Cache 相关算法的分析与改进

3.1 页面置换算法

Linux 系统的页面置换算法结合最久未使用算法(LRU)、时钟算法(CLOCK)和 2Q 算法[2]的优点,实现了简化的 LRU 2Q 算法[3]。算法实现过程如图 2 所示。

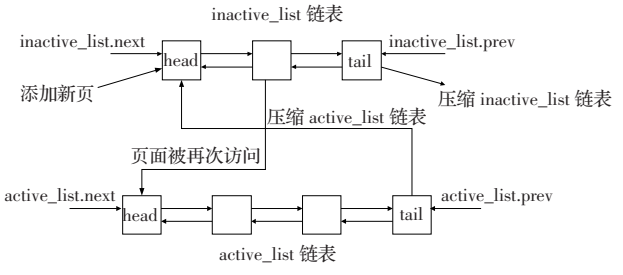

图 2 Linux2.6.25 内核页面置换算法实现过程

Linux2.6.25 内核中分别维护了 active_list 和 inactive_list 两个链表。active_list 链表保存访问频率高的页面,inactive_list 链表保存访问频率相对较低的页面。第一次分配的页面会被放置在 inactive_list 链表中，只有当 inactive_list 链表中的页面被再次访问时,页面才能被移至 active_list 链表中。inactive_list 链表扮演着过滤器的角色,将访问频率低的页面屏蔽在 active_list 链表之外，保证了 active_list 链表中页面均是访问频率高的页面,充分体现了在内存中保存最近且最常用页面的原则。

当系统内存不足时,即 active_list 链表和 inactive_list 链表超过指定长度时，系统将调用页面回收进程压缩 active_list 链表和 inactive_list 链表。active_list 链表中被压缩的页面被移入 inactive_list 链表,inactive_list 链表中被压缩的页面被回收或写入磁盘。

系统回收页面有两个特点：首先系统并不是全局扫描链表,从而选择最久没有使用的页面回收。系统仅局部扫描链表并进行页面回收工作。假设系统所需页面为 *n_total*,系统仅仅从链尾开始扫描 *n_total* 个页面;其次,系统并非一次性回收 *n_total* 个页面,而是逐步回收页面。系统单次扫描的页面数 *n_shrink* 为:

$$n\_shrink=\frac{L}{2^{DEF\_PRIORITY}} \tag{2}$$

公式(2)中 L 表示链表长度,*DEF_PRIORITY* 表示优先级,其初始值为 6[3]。

如果系统一次没有回收足够的页面,系统会逐渐减小优先级，以增加每次扫描页面的数量 *n_shrink*，直至系统回收到 *n_total* 个页面。

虽然每次计算出的扫描页面的数量为 *n_shrink*,但是系统并不立即执行扫描工作。如果 *n_shrink* 小于 *SWAP_CLUSTER_MAX*，系统将等待并累加 *n_shrink* 的值，直至 *n_shrink* 达到 *SWAP_CLUSTER_MAX* 才开始执行页面回收工作。如果 *n_shrink* 大于 *SWAP_CLUSTER_MAX*,系统将先回收 *SWAP_CLUSTER_MAX* 个页面,再逐步回收剩余的页面。

3.2 预读算法

Linux2.6.25 系统的预读算法采用按需预读,并将预读分为同步预读和异步预读。在文件首次读请求时,系统采用同步预读,从磁盘预取少量页面放入文件 Cache 中。如果第二次读请求命中 Cache,则系统开始采用异步预读,即扩大预读的页面数。如果第二次读请求没有命中 Cache,则系统继续采用同步预读。当系统采用异步预读时,系统会动态调整预读的页面数目。在异步预读过程中,预读页面数的初始值是读大小的 2~4 倍。后续的预读页面数将逐次倍增,直到达到系统设定的最大预读大小 *read_ahead_kb*。

3.3 改进策略

通过以上分析可知,Linux2.6.25 系统一次预取的最大页面数为 *read_ahead_kb*。执行一次回收页面扫描所回收的页面数是 *SWAP_CLUSTER_MAX*。在 Linux2.6.25 系统中 *read_ahead_kb* 为 128 K,*SWAP_CLUSTER_MAX* 的值是 32。因此,当访问数据块远小于 128 K 时,文件 Cache 的页面置换算法中 *n_shrink* 小于 32 的可能性较大。此时系统只能等待直至 *n_shrink* 达到 32 个页面才能进行页面回收的工作。当访问数据块远大于 128 K 的时候,Cache 页面置换算法中 *n_shrink* 大于 32 的可能性较大。此时系统只能将页面回收工作分部进行,每次回收 32 个页面。同时系统预取页面最大值为 128 K,所以当数据块远大于 128 K 的时候,Cache 中数据的更新速度将低于访问数据的更新速度,因此 Cache 命中率也随之降低。

在上述两种情况下,Cache 命中率相对低，系统读写性能相对较差。因此,若要使系统在被不同大小数据块访问时均保持稳定良好的性能,必须保证 Cache 中数据的更新频率与访问数据的更新频率相匹配。在算法实现时,必须突破系统页面置换算法中单次回收页面数是 *SWAP_CLUSTER_MAX* 的限制。通过动态调整单次扫描时回收页面的页面数,使得单次回收的页面数与 *n_shrink* 相匹配,从而使 Cache 中数据的更新频率能够根据数据块的大小动态变化。同时,应该考虑预取算法对于页面置换算法的影响。通过扩大 Linux 系统中预取粒度的选择

范围,使得在调整回收页面的速率的同时可以动态改变预取页面的数量。

4 性能测试与分析

根据提出的改进策略,设计了实验进行测试,实验系统的配置如表 1 所示。

表 1 实验环境

项目	内容
CPU	Inter® Pentium® Dual CPU 1.7 GHz
硬盘	希捷 160 GB
内存	1 GB
操作系统	Fedora 9
文件系统	Ext3
交换分区	200 MB
内核版本	Linux2.6.25.15

实验采用系统 IO 的基准测试工具 Iozone 来测试修改前后 Linux 系统的读写性能。为全面反映整个计算机系统的读写性能,Iozone 所测试的读写必须涉及读写硬盘操作。因此,在用 Iozone 测试 Linux 系统时,写测试文件的大小必须大于系统交换分区大小,读测试文件的大小必须大于系统缓存大小。因为所设交换分区大小为 200 MB,所以选取写文件大小为 256 MB。系统内存为 1 GB,所以选取读文件大小为 2 GB。

写测试结果如图 3 所示。

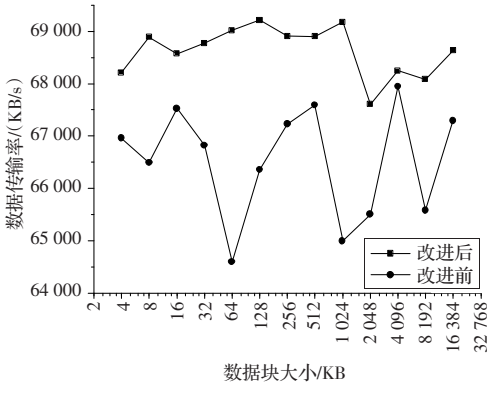

图 3 写性能测试结果

读测试结果如图 4 所示。

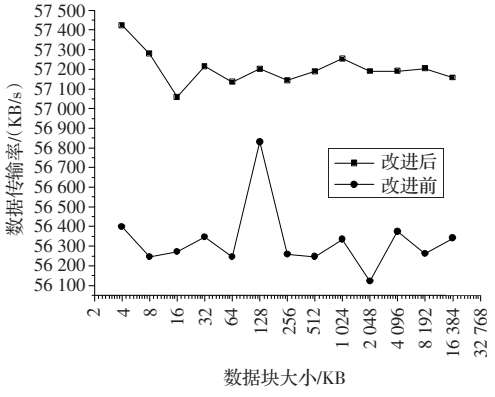

图 4 读性能测试结果

以上实验结果表明:

(1)改进前的系统在读写文件时,系统性能随着文件数据块大小的变化波动较大,在部分访问数据块大小下,系统读写性能较差。改进后的系统的读写性能随着文件数据块大小的变化波动较小,在不同数据块大小下系统读写性能差异小。

(2)改进后的系统的读写性能整体优于改进前的系统。

因此,文件 Cache 自适应策略有效地提高了 Linux 系统的读写性能。因为策略的重点在于保证 Linux 系统在被不同大小数据块访问时保持稳定的读写性能。所以,针对特定数据块大小,策略所得到的系统读写性能并非是系统最优值,而是较优值。针对访问数据块大小相对单一的系统,可以根据前文对于页面置换算法和预取算法的分析,合理修改相关参数来获得最佳值。

5 结语

提出了一种提高 Linux 系统读写性能的文件 Cache 自适应策略。该策略根据不同大小的访问数据块的应用特点,动态改变系统的预取策略和页面置换策略,提高了文件 Cache 的命中率,从而提高了 Linux 系统的读写性能。文件 Cache 自适应策略的优点在于考虑了预取算法对于页面置换算法的影响,提高了页面置换算法对于访问数据块大小的适应性。实验证明该策略能使 Linux 系统在被不同大小数据块访问时保持稳定且更优的读写性能。

参考文献:

[1] 郑纬民,汤志忠.计算机系统结构[M].2 版.北京:清华大学出版社,2002:193.

[2] Johnson T,Shasha D.2Q:A low overhead high performance buffer management replacement algorithm[C]//Bocca J B,Jarke M,Zaniolo C.Proceedings of 20th International Conference on Very Large Data Bases,Santiago de Chile,VLDB'94.1994.San Fransisco:Morgan Kaufmann,1994:439-450.

[3] Gorman M.Understanding the Linux virtual memory manager[M].London:Prentice Hall,2005:150-160.

[4] 钮俊清,郑浩然,李恒,等.一种基于有限记忆多 LRU 的 Web 缓存替换算法[J].小型微型计算机系统,2008,29(6):1010-1014.

[5] 田小波,陈蜀宇.基于最小效用的流媒体缓存替换算法[J].计算机应用,2007,27(3):733-736.

[6] 罗益辉,谢长生,张成峰.存储系统的集中式 Cache 替换算法[J].华中科技大学学报:自然科学版,2006,34(11):41-43.

[7] 倪继利.Linux 内核分析及编程[M].北京:电子工业出版社,2005:114-147.

[8] Bovet D P,Cesati M.Understanding the Linux Kernel[M].3rd ed.Beijing:O'Reilly,2005:150-160.

[9] Wei Qin-qi,Xie Chang-sheng,Li Xu.Stripe-Cache:An efficient cache scheme for building multimedia oriented RAID system[C]//LNCS 4553:HCI Applications and Services,Human-Computer Interaction,Beijing,2007.Germany:Springer-Verlag,2007:1130-1139.