[文章编号] 1007－7405(2015)05－0396－05

子页感知的闪存页面置换算法

刘君玲

（福建信息职业技术学院计算机工程系，福建 福州 350003）

[摘要] 根据闪存的独特物理特性，提出了子页感知的闪存页面置换算法. 该算法引入了子页技术和基于相似概率的部分更新机制，既可以提高闪存存储系统的性能，又可计算每个内存页的置换值，并选择了置换值最小的内存页为牺牲页. 实验结果表明，新算法在页面命中率、读/写操作次数、运行时间方面均具有优势.

[关键词] 闪存；页面置换算法；子页技术；企业；存储；最近最少使用算法

[中图分类号] TP 15 [文献标志码] A

Subpaging Aware Page Replacement Algorithm for Flash Memory

LIU Jun-ling

(Fujian Polytechnic of Information Technology, Fuzhou 350013, China)

Abstract: Due to distinctive physical characteristics, this paper proposes a subpaging aware page replacement algorithm for flash memory. The proposed algorithm introduces a subpaging technology and a partial update scheme based on similar probability in order to improve the performance for flash memory storage system. At the same time, the proposed algorithm calculates the replacing value of each main memory page and selects the one with the least replacing value as the eviction page. Experimental results show that the proposed subpaging aware page replacement algorithm is better than current page replacement algorithms on page hit ratio, the number of write/read operations and runtime.

Key words: flash memory; page replacement algorithm; subpaging technology; enterprise; storage; least recently used algorithm

0 引言

由于闪存具备访问速度快、质量轻、抗震性强以及能耗低等优点，闪存已经成为智能手机和嵌入式系统的重要存储介质[1]. 随着闪存技术的不断发展和成熟，闪存的容量不断增大且价格不断降低，使得闪存成为代替传统机械硬盘的重要存储介质之一[2]. 国内的华为、奇虎以及阿里巴巴等大型科技公司已经开始采用闪存代替部分传统机械硬盘，以提升企业存储系统的性能. 操作系统中现有的页面置换算法是针对传统机械硬盘的机械特性进行优化和设计的[3]. 然而，闪存表现出与传统机械硬盘完全不同的物理特性. 例如，闪存的读写操作性能是不一致的，其写操作延迟远高于读操作延迟[4]. 如果将现有的页面置换算法直接应用于闪存存储设备，势必会造成闪存存储系统的能耗居高不下.

最近最少使用算法（LRU）是大部分现代操作系统为最大化页面命中率而广泛采用的一种页面置换算法[5]. 为了减少写操作数量，文献［6］提出了第一个适用于闪存存储设备的页面置换算法

[收稿日期] 2014－12－22 [修回日期] 2014－05－22
[作者简介] 刘君玲（1973—），女，实验师，从事计算机应用研究.

——CFLRU. 该算法改进了原有的 LRU 算法，在 LRU 页面将队列分成干净页优先区域和工作区域两部分. 干净页优先区域包含将来最有可能被置换出去的最近最少使用页面. 工作区域包含最近最常使用页面，所以大部分的缓存命中情况发生在该区域内. CFLRU 算法优先置换干净页优先区域内的干净页，当干净页优先区域内没有干净页时，CFLRU 算法按 LRU 顺序置换出剩余的脏页.

文献［7］引入写顺序重新排序策略对原有的 LRU 算法进行扩展，提出了一种新的页面置换算法 LRU-WSR. LRU-WSR 按照 LRU 顺序对 LRU 页面队列中的页面进行扫描并检查. 如果该最近最少使用页面是干净页，不管该页面是否是热页面，LRU-WSR 算法将该干净页回收相应的页框；如果该最近最少使用页面是热脏页，LRU-WSR 算法将其设置为冷脏页，并将其插入 LRU 队列中 MRU 端；如果该最近最少使用页面是冷脏页，LRU-WSR 算法将该页写回闪存存储设备并回收相应的页框.

文献［8］针对闪存读写操作成本不对称的特点，提出了一种基于概率的页面置换算法 APB-LRU. 该算法根据数据访问频度，将缓冲区分为冷区和热区两部分，APB-LRU 以较高的概率置换出冷区的干净数据页，以较低的概率置换出热区的脏数据页.

文献［9］提出了一种新的闪存页面置换算法 PT-LRU，该算法维护三条主要列表：冷干净 LRU 列表、冷脏 LRU 列表以及混合 LRU 列表. PT-LRU 优先从冷干净 LRU 列表中置换出 LRU 页，如果冷干净 LRU 列表为空，PT-LRU 则以较高的概率置换出冷脏 LRU 列表中的页，以较低的概率置换出热干净页.

通过分析，作者发现现有的闪存页面置换算法通过优先置换出干净页的策略，可以有效地减少写操作次数，从而达到降低闪存存储系统能耗的目的. 但内存中的脏数据页通常含有大量的干净数据，置换出这类脏数据页会引发大量的冗余写操作. 为了减少冗余写操作次数并保持较高的页面命中率，本文试提出一种子页感知的闪存页面置换算法.

1　子页感知的闪存页面置换算法

1.1　子页技术

在操作系统中，为了保持数据一致性，当页面置换算法选择的牺牲页为脏页时，页面置换算法先要把脏页写回存储系统中，最后才回收对应的页框. 现代操作系统中页的大小普遍为 4 KB，而闪存中页的大小为 512 B，也就是说写回一个脏页会触发 8 个写操作.

为此，本研究在 Linux 操作系统上独立运行 6 个常见的应用程序，分别是 Apache OpenOffice Writer，Apache OpenOffice Calc，Firefox，Viewnior，Xmms 以及 Foxit Reader. 通过测试发现，内存中的脏页通常含有大量的未修改数据，即干净数据. 定义内存页平均脏度为内存页中已修改数据的数量与内存页总的存储容量比率.

从表 1 可以看出，只有 Viewnior 的脏度相对较高，Firefox 和 Xmms 的脏度低于 50%，说明这两种软件运行时内存页中含有大量的干净数据.

表 1　不同工作负载下脏内存页的平均脏度

Tab. 1　The average dirty degree of the dirty main memory page in each workload

应用程序 Applications	描述 Description	脏度 Dirty degree/%
Apache OpenOffice Writer	文字处理软件 Word processing software	84.21
Apache OpenOffice Calc	电子表格软件 Sheet software	82.35
Firefox	Mozilla 公司开发浏览器 Bowser developed by Mozilla	46.53
Viewnior	Linux 操作系统下的图像查看工具 Image viewer tootl for Linux	96.07
Xmms	Linux 操作系统下的控制台音频播放软件 Console audio player for Linux	41.67
Foxit Reader	Linux 操作系统下的 PDF 阅读器 PDF reading tool for Linux	65.16

如图 1 所示，为了识别内存页中的干净数据，本研究采用网络系统中常用的子页技术（Subpaging Technique）将脏内存页划分成一定数量的子页，子页的大小跟闪存的大小一致[10]. 本文假定操

作系统的页大小为 4 KB，而闪存的页大小为 512 B，所以每个脏内存页可以划分成 8 个子页．每个子页都要关联一个脏位，如果子页包含已修改数据，说明该子页为脏子页，其脏位可设置为 1．如果子页内的数据都是未修改数据，说明该子页为干净子页，其脏位值为 0.

图 1 子页技术

Fig.1 Subpaging technology

1.2 算法设计

根据子页技术划分的结果，内存页 P 的置换成本定义为：a_p/b_p，其中，a_p 表示内存页 P 中脏子页的数量，b_p 表示内存页 P 中所有子页数量．从该定义可以看出，内存页中脏子页数量越高，内存页的置换成本也就越高．页面置换算法的命中率与牺牲页的最近访问时间和访问频率有关．为此，本研究综合了最近访问时间和访问频率两个因素，把内存页的热度定义为：$\sum_{i=1}^{n} D(t_c - t_{pi})$，其中：$D(x)$ 是一个权重函数，$D(x) = (1/2)^{x\ln 2}$，$\{t_{p1}, t_{p2}, \cdots, t_{pi}, \cdots, t_{pn}\}$ 是内存页最近 n 次的访问时间；t_c 表示当前时间．内存页的热度可以表明该页在将来被访问的概率．所以内存页的热度越小，那么它在将来被访问的概率也就越小.

如图 2 所示，子页感知的闪存页面置换算法维护两条 LRU 页面队列，分别是干净页面队列和脏页面队列．该算法为每一个内存页分配一个置换值，当内存中空闲页框不足时，页面置换算法选择置换值最小的内存页．为了减少写操作数量，同时保持较高的页面命中率，该置换值（Replacing value）$R(p)$ 定义为：$R(p) = \lambda \times a_p/b_p + (1-\lambda) \times \sum_{i=1}^{n} D(t_c - t_{pi})$．该置换值的定义考虑了内存页的热度和置换成本两个因素，这两个因素通过加权系数 λ 加权得到页面的置换索引值．这两个因素的比重与加权系数 λ 相关，会随着加权系数 λ 的变化而变化.

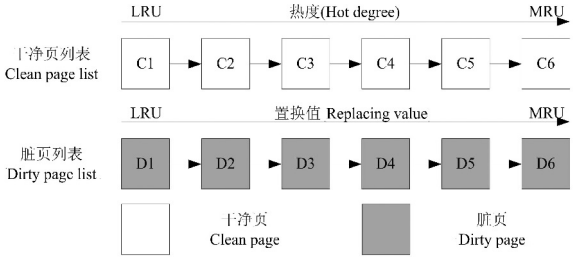

图 2 子页感知的闪存页面置换算法

Fig.2 Subpaging aware flash memory page replacement algorithm

为了避免过度优先置换出置换成本低的页（包括干净页）而影响命中率，所以加权系数 λ 定义成一个随 δ 增大而减小的单调递减函数：$\lambda = 2/(1+e^{\delta/8})$，其中：$\delta = C_w/C_r$ 且 $\delta \geqslant 2$，C_w 代表写操作成本，C_r 代表读操作成本，因此 δ 代表写操作成本与读操作成本的比率．从图 3 可以看出，加权系数 λ 随着 δ 的增大而减小．当 $2 \leqslant \delta < 8$ 时，$\lambda > 0.5$，内存页的热度成为影响牺牲页选择的重要因素．当 $\delta = 8$ 时，$\lambda \approx 0.5$，选择牺牲页的时候需要同时考虑这两个因素．当 $\delta > 8$ 时，$\lambda < 0.5$，内存页的置换成本成为影响牺牲页选择的重要因素．当 δ 趋向于无限大时，T 近似等于 1，干净页可以被优先置换出去，而不影响算法的整体性能.

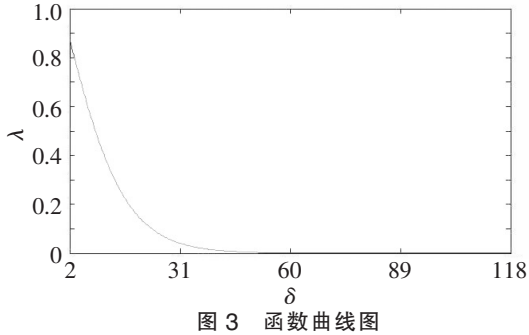

图 3 函数曲线图

Fig.3 Function curve graph

因为干净内存页不含有脏数据，所以根据置换值定义可以得到干净内存页的置换成本为 0，也就是说干净页队列上每个内存页的置换索引值等于其热度．因此，干净页队列上的内存页按照其热度值从左至右递增排列，而脏页队列上的内存页按照其置换索引值从左至右递增排列.

1.3 基于相似概率的部分更新机制

为了进一步减少写操作次数和降低垃圾回收额外开销，本文引入基于相似概率的部分更新机制.

首先，计算子页写回闪存存储器后变成不正确页（Invalid Page）的概率：1）假定闪存存储器由 T 个物理块组成，每个块包含 N_p 个页，闪存存储器中脏页的数量为 N_d；2）假设从上次子页写操作结束到当前的时间间隔，共产生了 K 次用户写操作；3）那么子页被更新的概率，即变成不正确页的概率为 $8/(T \times N_p - N_d)$. 然后，将具有相似概率的子页聚类到同一个簇：1）设定一个概率阈值 P_t；2）如果两个子页的概率差小于等于概率阈值 P_t，那么这两个子页就具有相似概率；3）将具有相似概率的子页聚类到同一个簇，直到簇的大小为 8. 最后，将大小为 8 的簇内子页写回闪存存储器.

2　实验分析

实验的硬件环境是 Intel Core i5-4200 3.5GHz CPU 和 4GB DDR3 内存，实验的运行环境为 Windows XP 操作系统. 本文的实验采用中国科技大学开发的 Flash-DBSim[11] 闪存仿真软件模拟闪存存储系统. Flash-DBSim 是一个可配置的闪存仿真平台，可以根据上层应用程序的需要模拟出不同特性的闪存存储系统，现有的很多研究都采用了 Flash-DBSim 进行算法性能的测试实验.

本文采用 Flash-DBSim 闪存仿真软件模拟一个存储大小为 256 MB 的 NAND 闪存存储系统，该系统的每个数据块包含 64 个数据页，且每个数据页大小为 2KB.

本文采用真实轨迹进行轨迹驱动模拟实验，将本文提出的子页感知的闪存页面置换算法（SPF）与现有的 CFLRU、LRU-WSR、APB-LRU 和 PT-LRU 算法进行性能对比. 实验过程中采用的真实轨迹是通过磁盘驱动数据跟踪器 DiskMon 收集访问磁盘的 I/O 数据请求.

模拟实验采用的性能评价指标为页面命中率、写操作次数、读操作次数以及运行时间. 页面命中率为从缓存中读取数据的次数与所有访问数据次数之比；写操作次数定义为闪存存储系统向闪存写入数据的次数；读操作次数定义为闪存存储系统从闪存读取数据的次数；运行时间定义为物理读操作时间、物理写操作时间和闪存中擦除操作的时间和.

从图 4 显示的实验结果可以看到，本文提出的子页感知的闪存页面置换算法的页面命中率高于现有的闪存页面置换算法. 这是因为本文算法在选择牺牲页的时候考虑了每个内存页的热度，防止经常访问的内存页被置换出去，可以有效地提高页面命中率.

图 5 显示的实验结果表明，本文提出的子页感知的闪存页面置换算法的写操作次数低于现有的闪存页面置换算法，这是因为本文算法在选择牺牲页的时候考虑了每个内存页的置换成本，且只将牺牲页中的脏子页写回闪存存储设备.

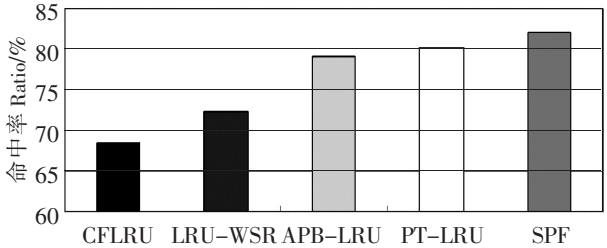

图 4　页面命中率

Fig.4　Page hit ratio

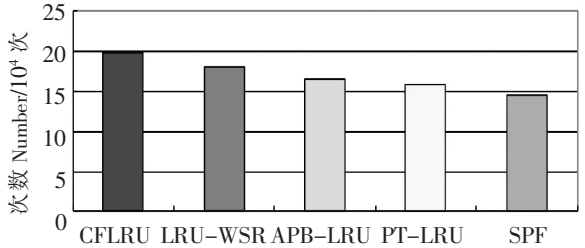

图 5　写操作次数

Fig.5　The number of write operations

图 6 显示的实验结果表明，本文提出的子页感知的闪存页面置换算法的读操作次数少于现有的闪存页面置换算法，这是因为本文算法的页面命中率最高.

图 7 显示的实验结果表明，本文提出的子页感知的闪存页面置换算法的运行时间低于现有的闪存页面置换算法. 页面置换算法的运行时间与其页面命中率和读操作次数有关. 页面命中率越高，其运行时间越低；读操作次数越少，其运行时间越低.

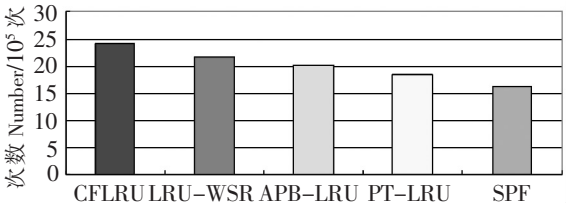

图 6 读操作次数

Fig.6 The number of read operations

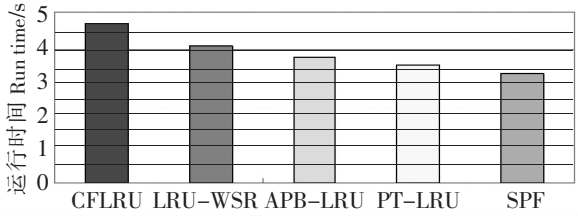

图 7 运行时间

Fig.7 Runtime

3 结论

本文针对闪存的物理特性，提出了子页感知的闪存页面置换算法. 先是引入子页技术将每个脏内存页划分成 8 个大小相同的子页，接着结合内存页的置换成本和热度计算出每个内存页的置换值，然后选择置换值最小的内存页作为牺牲页，最后引入基于相似概率的部分更新机制将具有相似概率的子页聚类到同一个簇并将该簇内的子页写回闪存存储器. 实验结果表明，本文提出的子页感知的闪存页面置换算法的性能高于现有的闪存页面置换算法.

[参 考 文 献]

[1] WEI Y T, SHIN D K. NAND flash storage device performance in Linux file system [C] //Proceedings of the 6th International Conference on Computer Sciences and Convergence Information Technology, Washington: IEEE Computer Society, 2011: 574-577.

[2] NO JAECHUN. Hybrid file system using NAND-flash SSD [C] //Proceedings of 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Piscataway, USA: IEEE Computer Society, 2011: 380-385.

[3] ASIT DAN, DON TOWSLEY. Approximate analysis of the LRU and FIFO buffer replacement schemes [C] //Proceedings of the 1990 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, New York: Association for Computing Machinery, 1990: 143-152.

[4] CHANIK PARK, JEONG-UK KANG, SEON-YEONG PARK, et al. Energy-aware demand paging on NAND flash-based embedded storages [C] //Proceedings of the 2004 International Symposium on Low Power Electronics and Design, New-Port Beeth, USA: Association for Computing Machinery, 2004: 338-343.

[5] ELIZABETH J O'NEIL, PATRICK E O'NEIL, GERHARD WEIKUM. The LRU-K page replacement algorithm for database disk buffering [C] //Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Fort Collins, USA: Association for Computing Machinery, 1993: 297-306.

[6] SEON-YEONG DAWOON JUNG, JEONG-UK KANG, JIN-SOO KIM, et al. CFLRU: a replacement algorithm for flash memory [C] //Proceedings of the 2006 International Conference on Compliers, Architecture and Synthesis for Embedded Systems, New York: Association for Computing Machinery, 2006: 234-241.

[7] HOYOUNG JUNG, HYOKI SHIM, SUNGMIN PARK, et al. LRU-WSR: Integration of LRU and writes sequence reordering for flash memory [J]. IEEE Transactions on Consumer Electronics, 2008, 54(3): 1215-1223.

[8] 林子雨, 赖明星, 邹权, 等. 基于替换概率的闪存数据库缓冲区替换算法 [J]. 计算机学报, 2013, 36(8): 1568-1581.

[9] CUI J H, WU W G, WANG Y F, et al. PT-LRU: A probabilistic page replacement algorithm for NAND flash-based consumer electronics [J]. IEEE Transactions on Consumer Electronics, 2014, 60(4): 614-622.

[10] LI H L, YANG C L, TSENG H W. Energy-aware flash memory management in virtual memory system [J]. IEEE Transactions on Very Large Scale Integration Systems, 2008, 16(8): 952-964.

[11] SU X, JIN P Q, XIANG X Y, et al. Flash-DBSim: A simulation tool for evaluating flash-based database algorithms [C] //2009 2nd IEEE International Conference on Computer Science and Information Technology, Piscataway, USA: IEEE Computer Society, 2009: 185-189.

(责任编辑 朱雪莲 英文审校 黄振坤)