一种基于时钟自适应的改进缓存替换算法*

魏文国，赵慧民，庄林凯，许鸿俊
（广东技术师范学院电子与信息学院，广东 广州 510665）

摘 要：缓存算法在存储系统、数据库、Web 服务器等计算机领域有很广泛的应用，缓存命中率是衡量缓存算法优劣的指标之一，对经典的缓存页面替代算法 LRU、CLOCK、ARC 和 CAR 进行了比较和分析，提出了一种基于时钟自适应的改进缓存替换算法——ICAR，它能更精确地对读请求的“频率”特性进行管理。实验结果表明，在几种典型的概率分布（例如随机分布、泊松分布和正态分布）的读请求进入缓存的情况下，ICAR 在大部分情况下都比 CAR 和 LRU 算法有更高的缓存命中率。但是当缓存命中率相当高（高于 80%）或者比较低（低于 30%）的情况下，ICAR 算法并不能总是表现出更好的性能，值得进一步研究。

关键词：缓存；替换算法；命中率
中图分类号：TP391 **文献标志码**：A **文章编号**：0529－6579（2012）06－0054－05

An Improved Clock Adaptive Cache Replacement Algorithm

WEI Wenguo, *ZHAO Huimin*, *ZHUANG Linkai*, *XU Hongjun*
(College of Electronics and Information, Guangdong Polytechnic Normal University, Guangzhou 510665, China)

Abstract: The caching algorithm has a very wide range of applications in the field of storage systems, database system and Web server, the cache hit ratio is one of cache measure indicators. Based on analysis of classic cache page replacement algorithms—LRU, CLOCK, ARC and CAR, the improved clock adaptive cache replacement algorithm—ICAR is proposed, which can more accurately manage read requests “frequency” characteristics. The experimental results show that if read requests fulfill several typical probability distribution (for example, random distribution, the Poisson distribution and normal distribution), ICAR can get higher cache hit rate than CAR and LRU algorithm in the majority of cases. But when the cache hit rate is very high (above 80%) or low (less than 30%) case, ICAR algorithm can not get better performance than CAR, it's worthy of further research.

Key words: cache; replacement algorithm; the hit rate

1 研究背景分析

缓存算法是存储系统、数据库、Web 服务器等计算机领域的核心技术之一[1－5]，几种经典的缓存页面替代算法 LRU、CLOCK、ARC 和 CAR 各有优缺点[6]，其中 LRU 是代替最近最少使用的页面的算法[7－8]，Oracle 数据库使用该算法管理缓存。LRU 实现极其简单，具有常数的时间和空间复杂度，能捕捉很多工作负载的“最近期”特性。但是 LRU 有 4 个很明显的缺点：

1）用进程锁去保证数据正确和统一，这是在高性能、高吞吐量环境下所不能接受的，如虚拟缓存管理、数据库和文件系统。

2）在每次命中缓存时，它必须移动到最多最近使用（MRU）的位置。在异步计算环境下，多个

* **收稿日期**：2012－04－28
基金项目：国家自然科学基金资助项目（61272381）；广东省自然科学基金资助项目（10151063301000000）
作者简介：魏文国（1968 年生），男，博士，教授；E-mail：wgwei@21cn.com

线程都试图将页面移动到 MRU 位置是不可接受的。

3）LRU 捕捉工作负载的“最近期”特性，但是没有捕捉“频率”特性。

4）LRU 容易被内存扫描影响，即一系列一次使用的页面导致系统的低性能。

CLOCK 缓存算法是 LRU 的一种单位元量近似实现[9-10]，它在每个页面包含“单位引用次数”，当页面第一次被插入缓存时，引用次数将设为 0。页面在缓存里以一个钟的形式循环存放，当缓存命中时，页面的引用次数设为 1。钟的指针用来替代缓存，当缓存容量满了的时候，指针会从起始位置开始扫描整个缓存，当指针指向引用次数为 1 的页面时，会将引用次数设为 0，如果指向引用次数为 0 的页面，则将此页面替换出缓存。CLOCK 解决了 LRU 的缺点 1 和缺点 2，它完全去除了进程锁，这样使得缓存算法能更快地进行缓存的插入和清理，大大提高了缓存工作的效率。CLOCK 算法第一次提出 page reference bit，每次缓存命中后只需将页面引用次数设为 1，不用像 LRU 一样每次缓存命中都将页面转移到 MRU 位置。但是缺点 3 和 4 导致 CLOCK 算法的实际命中率比 LRU 并没有明显的提高。这是因为两种算法都只捕捉“近期使用”的特性，而不能捕捉“频率”特性，导致了此类算法及其变体的命中率并没有实质性的提高。

2003 年，IBM 研究中心的 Nimrod Megiddo 等人[11]提出了 ARC 算法如图 1 所示，该算法创造性地运用四条 LRU 链：T1，T2，B1 和 B2 分别捕捉近期和频率特性。T1、T2 存放缓存中的数据，B1、B2 的数据并不放在缓存中。T1 管理近期使用页面，T2 管理多次使用的页面，B1、B2 则分别接收从 T1、T2 淘汰的页面并进行后续管理。ARC 算法将多次命中的页面从 T1 转移到有关频率的 T2 中，使得缓存能同时管理“近期”和“频率”两个重要的特性。ARC 解决了缺点 3 和缺点 4，将高频使用的页面与最近使用页面分开管理，从而捕捉到高频使用的页面并进行区分管理。同时也避免了在一次一系列浏览后，缓存页面被全部破坏的问题。这两点使得 ARC 缓存算法相对于 LRU，CLOCK 等类似的缓存算法在命中率方面有很明显的提高，ARC 思想的提出对当代缓存算法有十分重要的意义，后面涌现出众多类似的改良算法都基于此算法。ARC 算法虽然解决了缺点 3 和缺点 4，但是该算法的实现完全基于 LRU 算法，从而暴露出 LRU 几个严重的缺陷，其中上面所阐述的缺点 1 和 2 就是很关键的两点，LRU 自身的缺点导致了 ARC 在缓存替代效率上还有提高的空间。

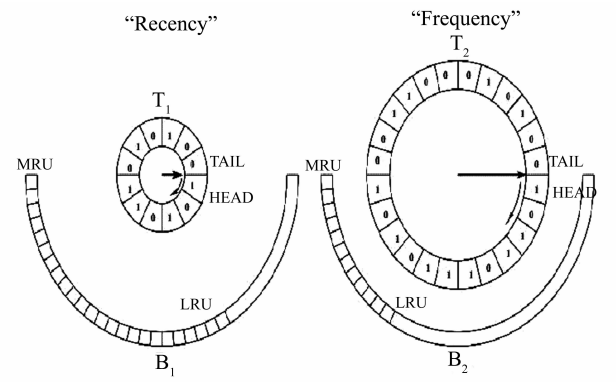

图 1　ARC 缓存管理算法数据结构示意图

Fig. 1　Data structure of ARC cache management algorithm

IBM 研究中心的 Dharmendra Modha[12-13]不久又提出了一种改良 ARC 的缓存算法 CAR：Clock with Adaptive Replacement。CAR 缓存算法是当前较经典的几个缓存算法之一，运用广泛。CAR 将 ARC 与 CLOCK 结合，同时解决了 LRU 的 4 个缺点，T1，T2 用 CLOCK 算法实现，B1，B2 用 LRU 实现。能同时捕捉“近期”和“频率”两个特性，并完全解决了 LRU 算法的缺陷。

虽然 CAR 对缓存算法进行了较充分的改良，但是它并不能很完善地捕捉缓存使用的“频率”特性。我们在文献［14］中提出了对缓存页面的“频率”特性进行更加精细化管理的缓存管理算法 GCAR，该算法在集群协作缓存环境下被验证能取得更好的缓存命中率，本文在此基础上进行改进，改变 GCAR 算法的部分流程，并在 3 种典型的概率分布（例如随机分布、泊松分布和正态分布）的读请求进入缓存的情况下进行对比测试，都能取得更好的缓存命中率，证明该 ICAR 缓存管理算法的适用性更好。本文还研究基于时钟自适应的改进缓存替换算法 ICAR 的思想与流程；在 3 种典型的概率分布（例如随机分布、泊松分布和正态分布）的读请求进入缓存的情况下进行对比测试 LRU、CAR 和 ICAR 的缓存命中率；最后给出结论和阐述将来的工作。

2　基于时钟自适应的改进缓存替换算法 ICAR

通过对 CAR 算法以及经典缓存算法的研究，我们发现：CAR 用于捕捉频率特性的 CLOCK T2 并不能很完善地捕捉“频率”特性。虽然多次使用的数据会从 T1 传递到 T2 并进行分开管理，但是

CLOCK T2 里的引用次数只是 0 或 1。这样导致即使该数据块经过了多次访问，但是它的引用次数依然只是 1。当缓存进行替代操作的时候，多次访问的数据块与少量访问的数据块在 T2 中引用次数都为 1，导致两者被清理出缓存的几率相同。所以，CAR 算法只能粗线条地对 T2 数据进行管理，并不能很精确地捕捉数据“频率”特性。

针对 CAR 不能精确捕捉数据块“频率”特性的缺陷，我们试图改变 CLOCK 的数据结构，使 CLOCK 的引用次数不局限于 0 和 1，当每次缓存命中的时候我们将引用次数进行“加 1”处理。这样可以使 CLOCK T2 能对“频率”特性进行更精确地管理。通过对 CLOCK 算法的改变，并对 CAR 算法的适应性调整，我们提出 ICAR 缓存算法，其流程如图 2 所示。与 ARC 和 CAR 算法类似，同样 T1 和 T2 采用 CLOCK 类型的缓存数据结构，他们包含了缓存里的所有数据。B1 和 B2 只是简单的 LRU 列表的实现，包含那些近期从缓存淘汰出的数据。

CLOCK T1 捕捉“近期”的页面，并且 T1 是按照缓存页面使用由新到旧排序头进尾出的循环链表；CLOCK T2 则捕捉“长期”的页面（即被频繁访问的页面），即 T2 是头进尾出的循环链表。从 T1 淘汰的数据进入 B1，即 B1 是按照页面使用由新到旧排序头进尾出的循环链表。从 T2 淘汰的数据进入 B2，即 B2 也是头进尾出的循环链表，B1、B2 的数据不存放在缓存中。算法限制 $|T1|+|T2|$ 的大小不超过缓存容量 c，将 B1 的大小限制与 T2 大小一致，B2 与 T1 大小一致。其中 T1 和 T2 的大小此消彼长，并且由自适应变量 p 调节：当 B1 内的数据命中时，T1 的大小将增加 1；反之，当 B2 内的数据命中时，T1 的大小将减小 1。

当缓存未命中时，新的数据将被插入至 T1，同时引用次数将被设为 0。当缓存命中任何在 T1 或 T2 的数据时，引用次数都将 +1。

“整理缓存”子模块的流程如图 3 所示，缓存清理时，CLOCK T1 的指针从尾部移动到头部。当 CLOCK T1 指针指向的页面引用次数为 0 时，页面将清理到 B1 的 MRU（最近使用）位置，当引用次数为不为 0 时，页面将插入至 CLOCK T2 的头部并使指针指向下一个页面；当 CLOCK T2 指针指向的页面引用次数为 0 或 1 时，页面将清理到 B2 的 MRU 位置，当引用次数大于 1 时，将页面引用次数 −1 并移动至 CLOCK T2 的头部并使指针指向下一个页面。

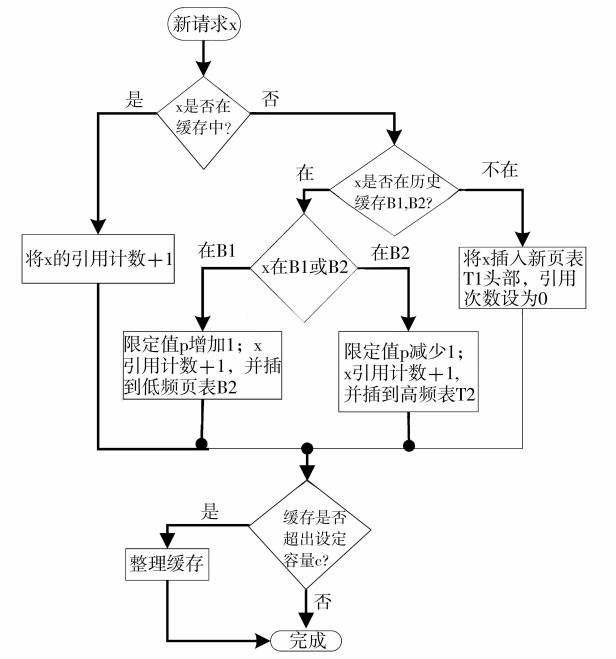

图 2 缓存替换算法 ICAR 流程图

Fig. 2 Flow chart of ICAR cache replacement algorithm

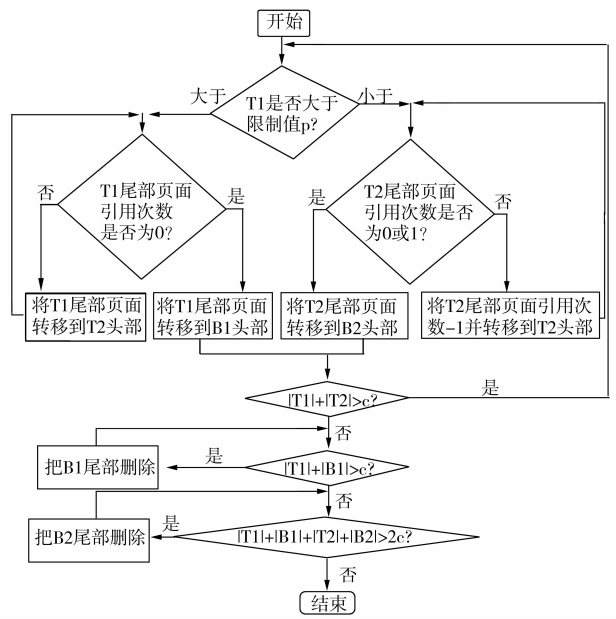

图 3 ICAR 子模块“整理缓存”流程图

Fig. 3 Flow chart of ICAR submodule finishing cache

3 仿真实验与测试结果

实验环境与条件：不失一般性假设，缓存容量为 1 024 个页面；主要测试读请求到达的时间服从随机分布、泊松分布和正态分布等三种代表性的情况；若读请求太大，超过缓存容量的一定比例之后，命中缓存的几率很低，研究的意义不大，所以假设每次读请求的大小在［1，20］之间随机产

生。使用 Java 语言开发仿真程序，模拟三种缓存管理算法：LRU、CAR 和 ICAR 在读请求按照上述概率分布到达时的缓存命中率。

第一组实验在读请求以随机分布到达时对比测试 LRU、CAR 和 ICAR 三种缓存管理算法的缓存命中率如图 4 所示。请求大小在［1，20］中随机产生，即读请求的平均大小为 10。缓存容量为1 024，表示缓存大约能够容纳（1 024 /100）约 100 个请求。当概率模型为随机分布时，读请求进入缓存的概率没有明显的高低之分，没有固定的高频页面，难以捕捉频率的特性，所以三种缓存算法的命中率都近似相同，当不同读请求的个数较少（例如图 4 最右边的只有 128 个不同读请求的测试案例），能够在一定程度反映读请求的“频率”特性时，ICAR 的缓存命中率较其他两种略高。

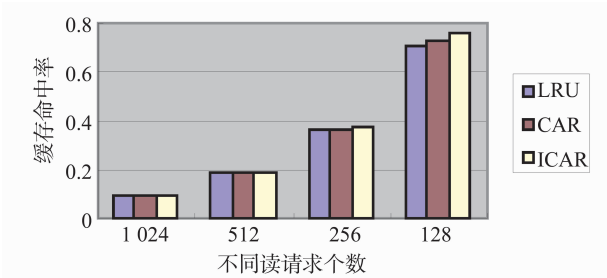

图 4　读请求按照随机分布到达的缓存命中率对比

Fig. 4　Cache hit ratios compared of random distribution read request

第二组实验在读请求以正态分布到达时对比测试 LRU、CAR 和 ICAR 三种缓存管理算法的缓存命中率如图 5 所示。高频率的读请求会更长时间停留在缓存中，结果表明：当命中率较低时（低于 30%），页面“频率”的特性难以捕捉，ICAR 的命中率与 CAR 命中率近似，但比 LRU 命中率高；当命中率适中时（高于 30% 且低于 80%），页面“频率”特性体现得较为明显，ICAR 较 CAR 命中率有比较明显的增加，平均增幅约为 12.5%；当命中率较高时（高于 80%），CAR 与 ICAR 两者缓存命中率相对于 LRU 都有非常明显的提高，但 ICAR 与 CAR 命中率几乎相同，因为“高频率”页面长期驻留缓存，不能体现差异。

第三组实验在读请求以泊松分布到达时对比测试 LRU、CAR 和 ICAR 三种缓存管理算法的缓存命中率。测试结果与正态分布有相似的规律。

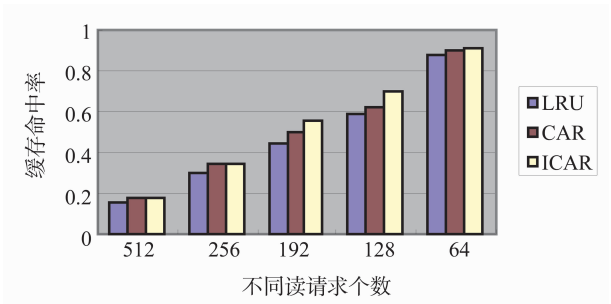

图 5　读请求按照正态分布到达的缓存命中率对比

Fig. 5　Cache hit ratios compared of normal distribution read request

4　总结与展望

本文比较和分析了 4 种经典的缓存管理算法 LRU、CLOCK、ARC 和 CAR，提出基于时钟自适应的改进缓存替换算法 ICAR，仿真实验验证该算法在大多数情况下能取得更高的缓存命中率。文献［14］中提出的 GCAR 缓存管理算法对集群协作缓存能取得较好的缓存命中率，本文的改进在于 ICAR 的适用性更好。进一步的研究可在缓存命中率相当高（高于 80%）或者比较低（低于 30%）的情况下改进和优化 ICAR 算法。

参考文献：

［1］　CHRIS G, ALI R, BUTT Y, et al. Program-counter-based pattern classification in buffer caching［C］// 6th Symposium on Operating Systems Design and Implementation, 2004: 321 – 349.

［2］　ALI R. The performance impact of kernel prefetching on buffer cache replacement algorithms［C］// Proceedings of the 2005 ACM Sigmetrics International Conference on Measurement and Modeling of Computer Systems, 2005: 157 – 168.

［3］　JEONG J. Simple penalty-sensitive cache replacement policies［J］. Journal of Instruction-Level Parallelism, 2008, 10: 1 – 24.

［4］　王欣，周南，邱小彬. JCS 数据缓存技术在动态 Web 系统中的应用［J］. 中山大学学报：自然科学版，2009（S1）: 356 – 357.

［5］　陈华竣，郑智，倪德明. 一种面向分层访问的目录结构在 RDBMS 中的存储方法［J］. 中山大学学报：自然科学版，2005（S1）: 138 – 141.

［6］　BANSAL S, MCKENNEY P E , MODHA D S. Apparatus and system for dynamically allocating main memory among a plurality of applications: US Patent, 7 487 320［P］. 2009 – 02 – 03.

（下转第 62 页）

[5] RUBINSTEIN R, ELAD M. Double sparsity: learning sparse dictionaries for sparse signal approximation[J]. IEEE Transactions on Signal Processing, 2010, 58(3): 1553 – 1564.

[6] AHARON M, ELAD M, BRUCKSTEIN A M. The K-SVD: an algorithm for designing of overcomplete dictionaries forsparse representation[J]. IEEE Transactions on Signal Processing, 2006, 54(11): 4311 – 4322.

[7] RUDELSON M, VERSHYNIN R. Sparse reconstruction by convex relaxation: Fourier and Gaussian measurements[C]// Proc CISS 2006 (40th Annual Conference on Information Sciences and Systems), 2006.

[8] KIM S J, KOH K, LUSTIG M, et al. A interiorpoint method for large-scale L1-regularized least-squares problems with applications in signal processing and statistics[J]. Journal of Machine Learning Research, 2007, 7(8): 1519 – 1555.

[9] TROPP J, GILBERT A. Signal recovery from random measurements via orthogonal matching pursuit[J]. IEEE Tram Inform Theory, 2007, 53(12): 4655 – 4666.

[10] 杨海蓉, 张成, 丁大为, 等. 压缩传感理论与重构算法[J]. 电子学报, 2011, 39(1): 142 – 148.

[11] RAUHUT H, SCHNASS K, VANDERGHEYNST P. Compressed sensing and redundant dictionaries[J]. IEEE Transactions on Information Theory, 2008, 54(5): 2210 – 2219.

[12] CHEN S, DONOHO D, SAUNDERS M. Atomic decomposition by basis pursuit[J]. SIAMJ Science Computer, 1999, 20: 33 ~ 61.

[13] BARANIUK R G. Compressive sensing[J]. IEEE Signal Processing Magazine, 2007, 24(4): 118 – 121.

[14] NEEDELL D, VERSHYNIN R. Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit[J]. Found Comput Math, 2009, 9(3): 317 – 334.

[15] 赵慧民, 郭一缜, 丁晓艳, 等. 用于视频多播传输的压缩传感实现方法研究[J]. 中山大学学报: 自然科学版, 2012, 51(1): 45 – 49.

[16] 练秋生, 周婷. 结合字典稀疏表示和非局部相似性的自适应压缩成像算法[J]. 电子学报, 2012, 40(7): 1416 – 1422.

[17] BLUMENSATH T, DAVIES M E. Iterative thresholding for sparse approximations[J]. Journal of Fourier Analysis and Applications, 2008, 14(5/6): 629 – 654.

(上接第57页)

[7] LEE D, CHOI J, KIM J, et al. On the existence of a spectrum of policies that subsumes the Least Recently Used (LRU) and Least Frequently Used (LFU) policies[C]// Proceeding of 1999 ACM Sigmetrics Conference, 1999: 134 – 143

[8] LI Zhansheng. CRFP: A novel adaptive replacement policy combined the LRU and LFU policies[C]// IEEE 8th International Conference on Computer and Information Technology, 2008: 72 – 79.

[9] BANSAL S. Method and system of clock with adaptive cache replacement and temporal filtering: US Patent App, 10/955 201[P]. 2006.

[10] JIANG S, CHEN F, ZHANG X. CLOCK-Pro: an effective improvement of the CLOCK replacement[C]// Proc of USENIX '05, 2005: 121 – 130.

[11] MEGIDDO N, MODHA D. ARC: a self-tuning, low overhead replacement cache[C]// Proceedings of the 2nd USENIX Symposium on File and Storage Technologies, 2003: 115 – 130.

[12] BANSAL S, MODHA D. CAR: clock with adaptive replacement[C]// Proceedings of the 3nd USENIX Symposium on File and Storage Technologies, 2004: 203 – 215

[13] SHAMSHEER S M. A throughput analysis on page replacement algorithms in cache memory management[J]. International Journal of Engineering Research and Applications, 2012, 2(2): 126 – 130.

[14] 魏文国, 陈潮填, 闫俊虎. 集群协作缓存机制研究[J]. 计算机科学, 2008 (1): 282 – 284.