混合架构下多请求模式的缓存替换模型研究

曹 旻 刘文中
(上海大学计算机学院 上海 200444)

摘 要 针对多类型多访问模式应用的需求,在GDSF算法的基础上,引入平均访问间隔和最近访问间隔两个特性以增强算法的适应性;建立缓存结构模型,通过双关键字索引机制,快速索引缓存对象,降低系统开销;对超过一定大小的文件采取后缀预取策略以增加缓存中数据对象的个数。在课题应用背景下,与传统算法的对比实验表明,该方法能够减少缓存的平均请求等待时间,提高对象命中率和字节命中率,增强了缓存替换算法对多类型多请求模式应用的适应性。

关键词 访问模式,缓存策略,缓存模型,索引,平均请求等待时间,预取
中图法分类号 TP315.69 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.6.038

Research on Cache Replacement Model Based on Multi-request Mode under Hybrid Architecture Model

CAO Min LIU Wen-zhong
(College of Computer Science, Shanghai University, Shanghai 200444, China)

Abstract For the social needs of many types of access modes and multiple applications, based on GDSF algorithm, this paper introduced two features of average access interval and recent access interval to enhance the adaptability of the algorithm. Cache structure model was built by double keyword indexing mechanism to index buffer object quickly and reduce system overhead. The suffix blocks of big file were prefectched to increase the number of data objects in the cache. In the background of the subject application, comparative experiments with the traditional method show that this method can make the average waiting time of the request, cache object hit rate and byte hit ratio get a comprehensive improvement and improve the adaptability of cache replacement algorithm for multi-type multiple requests mode application.

Keywords Access mode, Caching strategies, Cache model, Index, Average waiting time, Prefetching

1 引言

随着高效能计算的提出和发展,传统的高性能计算越来越不能满足现在纷繁多样的应用需求,同时也暴露了它适应性差、高能耗、利用率低的弊端,因此适应多种应用需求的可重构新型计算机体系结构的研究应运而生。国家863重点项目——基于认知的可重构新型计算机体系结构(Proactive Re-configurable Computing Architecture, PRCA)的研究与实现,采取具有重构特性的FPGA计算组件,结合传统GPU和CPU组成可动态重构的计算单元,除了要满足常规的高性能计算(如气象观测、图像匹配、密码破解等)外,还要能够实现大多数应用的代理服务,比如Web服务、视频点播等。在PRCA系统中,各种应用的计算任务所使用的资源都集中在专门的存储设备上,由于FPGA的执行速度比存储设备的读写速度高出多个数量级,就形成了"存储墙",它严重影响了FPGA的执行效率,成为系统性能提升的瓶颈。

PRCA系统的资源请求特点如下:

1)资源文件有小到kB级的图片文件,大到TB级的mkv等视频文件;

2)请求资源的访问频率差别较大,比如用于密码破解的字典,短时间内访问频率较高,而图像匹配中的相关图像资源可能仅被请求一次;

3)资源总量较大,请求模式涉及循环访问模式,且大小不定,易出现抖动现象;

4)由于FPGA的顺序执行任务的特性,对请求的平均请求等待时间要求较高,否则就会终止当前的任务,造成不必要的浪费。

PRCA系统的代理缓存的物理结构模型如图1所示。

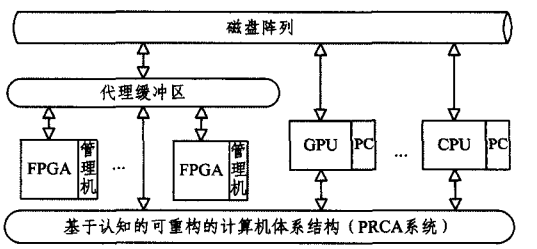


图1 PRCA系统代理缓存的物理结构模型

到稿日期:2014-07-15 返修日期:2014-11-18 本文受国家"863"计划基金资助项目(2009AA012201-CFA2009SHDX01),国家自然科学基金(61001163)资助。

曹 旻(1966—),女,副教授,主要研究方向为软件体系结构,E-mail:mcao@shu.edu.cn;刘文中(1990—),男,硕士,主要研究方向为分布式计算软件架构、文件缓存,E-mail:wzliu27@126.com。

综上可知，在 PRCA 系统中建立的缓冲区要满足多种应用的需求，因此需要设计适应多种应用的缓存替换算法。目前流行的缓存替换算法广泛应用于商业应用中，比如流媒体服务中有基于最小效用的流媒体缓存替换算法[1]、基于混合模式的流媒体缓存调度算法[2]、P2P 流媒体点播系统缓存策略[3]等；Web 应用中有基于预测的 Web 缓存替换算法[4]、基于页面内容的缓存替换算法[5]等；网络存储中有基于网络存储服务器的缓存替换策略[6]等。在上述主流的应用中，缓存替换算法有效地提高了对象命中率，减少了请求等待的时间，减少了服务的压力，提高了任务执行的效率。同样地，缓存替换算法对于本课题所遇到的“存储墙”问题也具有类似的作用，因此研究基于特定应用背景和需求的缓存替换策略是非常必要的。

2 相关工作

2.1 缓存策略评价标准

缓存替换算法有多种评价标准，主要有对象命中率、字节命中率和平均请求等待时间，这 3 个评价标准都对缓存替换算法进行了有效的评价，但很多缓存替换算法只重视前两种，而忽略了平均请求等待时间。

定义 1(对象命中率) 即命中次数和任务请求次数的比率，是最早被提出来用于评价缓存替换算法好坏的标准，它的值直接决定了缓存区的缓存效果。在一段时间内，任务请求总次数为 N，命中总次数为 K，则该缓存替换算法的对象命中率为：

$$H=K/N \tag{1}$$

由于缓存中对象文件大小的不同，字节命中率逐渐成为评价缓存替换算法好坏的标准。

定义 2(字节命中率) 即命中的总字节数与任务请求的总字节数的比率。在一段时间内，任务请求的总体字节数为 W，命中的总次数为 n，每次命中的对象的字节数为 $w(i)$，从而得出缓存替换算法的字节命中率为：

$$R=\frac{\sum_{1}^{n}w(i)}{w} \tag{2}$$

定义 3(平均请求等待时间) 即从请求发起到请求响应所经历的时间。任务请求经历的步骤如图 2 所示。任务请求的时间消耗点主要包括以下 3 个方面：

1)在缓冲区中，查找请求对象命中或缺失的效率；

2)查找是否有空闲缓存空间的效率；

3)查找要替换出的缓存对象的效率。

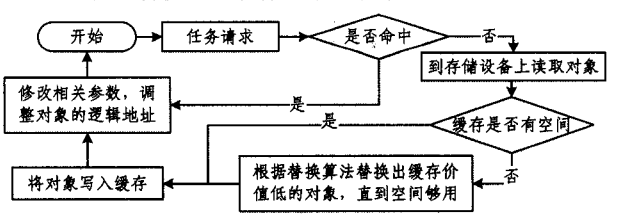


图 2 任务请求流程图

很多缓存替换算法没有对这些方面进行改善，仅仅以算法的对象命中率和字节命中率作为评价算法好坏的标准。然而从图 2 的流程中可以看出，平均请求等待时间直接影响任务请求的响应时间，进而影响缓存的性能。因此，平均请求等待时间也是衡量缓存替换算法好坏的一个重要标准。

2.2 任务请求的访问模式

不同应用类型执行任务的请求模式往往区别很大，普通缓存算法对请求模式的适应性较低，会造成缓存性能短板。因此，本文从数据访问模式的角度出发，来提高算法的适应性和总体缓存效果。Oracle 发布的标准测试集 TPC-H 中包含了多种数据访问模式，简要分类如下。

1)连续访问模式，对任务所需资源进行顺序访问。

2)循环访问模式，对资源进行规模大小不一的循环访问，比如图像识别应用的图像匹配。

3)短暂集中区域性访问模式，该模式下的任务请求符合局部性原理。

4)全局概率性访问模式，资源具有稳定的访问可能性，各数据块没有必然的联系。

5)完全随机访问模式，资源中每个对象被访问的概率是随机的。

2.3 传统的替换算法比较和分析

目前与缓存技术相关的算法比较多，从缓存对象的各种角度出发，分析其缓存的价值。缓存得到的性能的提升也随着实践的增多，更加地被业界认可，特别在流媒体应用、Web 应用、网络存储等服务中。目前典型的缓存替换算法主要分为 4 类，第一类是基于时间因素的，代表算法有 LRU(Lease Recently Used)[7]、LRU-K、Clock 算法[8]；第二类是基于频率因素的，代表算法有 LFU(Least FreQuently Used)算法[9]、2Q(Two queue)算法[10]、LIRS(Low Inter-reference Recency Set)算法[11]、ARC(Adaptive Replacement Cache)算法[12]；第三类是基于缓存对象大小的，代表算法有 Size、GDS 算法；第四类是混合类型的算法，综合了多个替换因子，代表算法有 LRFU(Least Recently Used/Least Frequently Used)算法、GDSF(Greedy Dual Size Frequent)算法[13]，还有针对流媒体、Web 缓存、网络存储等应用的专用算法。下面对一些典型算法进行分析。

基于时间因素的代表算法为最近最少使用(Lease Recently Used，LRU)[7]替换算法，它优先选择缓存中最近最少使用的数据块替换，实现简单，额外开销少，比较适合具有高局部性的数据访问模式，但对多种访问模式的适应性不好，如 PRCA 系统中的图像匹配涉及的循环扫描访问模式，当扫描的数据大小超过缓存大小时，LRU 几乎起不到任何缓存作用，反而消耗一定系统资源。它的衍生算法 LRU-K 和 Clock 算法也有同样的问题。

基于频率因素的代表算法为最少访问频率(Least Frequently Used，LFU)[9]替换算法，它优先选择缓存数据块中最少被访问的数据块进行替换，比较适合全局概率性访问模式，但适应的访问模式有限，对于瞬间请求频率较大而以后不再使用的数据块，可能会常驻内存。PRCA 系统中口令破解采用字典模式时，用到字典资源，该文件的使用频率会瞬间升高，造成缓存污染。在大规模的资源请求下，它的衍生算法 LIRS 和 ARC 算法也会出现同样的问题。

基于对象大小的代表算法为 Size 算法，它把请求对象的大小作为替换因子，优先选择替换缓存中较大的对象。该算法结构简单，额外开销少，在有限的缓存空间里能够存储更多

的小文件，因而提高了算法的对象命中率。但由于比较小的文件会长期占用内存而造成缓存污染，降低缓存的字节命中率。单独基于对象大小的算法，如它的衍生算法 GDS 也有同样的问题。

混合类型算法的代表算法为 GDSF(Greedy Dual Size Frequent)算法[13]，它虽综合了对象大小和访问频率等因素，但没有考虑时间因素，较小对象且短时间访问频率较大的缓存对象会常驻内存；基于预测的 Web 缓存替换算法[4]在 GDSF 算法的基础上加入预测机制，根据 Web 日志构造预测模型，通过预测模型对请求进行有效的预取。其优点是提高了 GDSF 算法的对象命中率和字节命中率，缺点则是使用范围狭窄，其预测模型是基于日志信息的，无法很好地应用到其他领域。

基于最小效用的流媒体缓存替换算法[1]是为了避免 LFU 和 LRU 算法中流媒体文件被连续替换的问题，通过流媒体文件块的访问次数、每次的播放时间和文件块的大小，考虑文件最近 K 次访问情况及相关参数，增加了替换对象所考虑的因素，提高了缓存的效果，但没有考虑对象的流行度和请求模式，缺乏必要的预取机制。

Aging 算法[5]参考 LIRS(Low Inter-reference Recency Set)算法[11]中缓存对象的访问间隔，利用时长记录的数据块的历史访问信息作为替换数据块的标志，替代了传统的 Recency。该算法在逻辑上将缓存分为 H 集合和 C 集合，只有 C 集合中的数据可以被替换，集合不采用固定大小。它把时长信息作为替换数据块的关键因素，虽然提升了缓存命中率，缩短了程序的执行时间，提升了计算机的整体性能，但采用判断位和标志位需要进行大量的计算和对比工作，需要足够的额外计算资源支持，且没有考虑请求访问模式、对象大小和访问频率等因素。

上述算法中非混合算法由于考虑因素单一，具有较大的性能缺陷；而混合算法大多是针对特殊应用的专用算法，依赖应用的相关数据(如 Web 的页面信息、日志信息等)，无法直接应用到 PRCA 系统中。虽然非混合算法对单个应用类型具有比较好的缓存效果，但对于 PRCA 系统中多任务类型多访问模式的缓存需求，无法从对象命中率、字节命中率和平均请求等待时间多个角度取得较好的综合缓存效果。

结合缓存替换算法，设计出良好的缓存模型是减少平均请求等待时间的有效途径。CPU 三级缓存模型是为了缓解 CPU 执行速度与内存的读写速度的差异而设计的，有效提高了 CPU 指令的执行效率，其中三级缓存中的数据是重叠的，虽然有效保证了数据的安全性，却降低了空间利用率；同时为了保证数据的一致性，在数据回写时也会带来较大的时间开销。这种绝对的安全性和数据的重叠机制对于 CPU 的设计是必须的，而在内存中建立缓存模型，就不需要再提供过多的安全机制，因此本文取消了数据重叠机制，根据缓存对象的访问时间、访问频率、对象大小等因素分配其存储方式，减少了维护数据一致性带来的开销。

本文在 GDSF 算法的基础上吸取了 Aging 算法中双集合和 CPU 三级缓存模型的思想，建立了三级结构的缓存模型，增加了流行度比较高的对象的生存周期，间接增加了对象被替换的参考因素；在原有的替换因子的基础上增加了一些因素，分别是替换对象的访问次数、大小、最近访问时间和最近 K 次访问情况；较大文件只缓存前 K 段数据，对后续数据采取预取机制。通过以上 3 方面的结合，使算法适应多类型多访问模式的应用，在对象命中率、字节命中率和平均请求等待时间上得到了较好的综合缓存效果。

3 算法设计

3.1 GDSF 算法原理及改进

GDSF 算法的主要原理是计算每个缓存对象的缓存价值，它充分考虑了对象的大小、访问频率、延迟代价等因素，当缓存空间满时，替换出缓存价值最小的对象。其计算公式为

$$K(i)=L+F(i)*\left(\frac{Value(i)}{\log Size(i)}\right) \tag{3}$$

其中，$K(i)$代表对象的缓存价值；L 为膨胀因子，初始值为 0，每次有缓存对象被替换出去时，L 都会被重新赋值为缓存中最小的缓存价值；$F(i)$为缓存对象被访问的次数；$Size(i)$为缓存对象的大小；$Value(i)$为将缓存对象引入缓存的代价(延迟、带宽等)。

GDSF 算法实现简单，缓存价值的计算开销较小，缺点是没有考虑缓存对象的最近访问时间和缓存对象的历史访问信息，对多种请求模式适应性差，当任务的请求模式频繁变动时，缓存效果不够好，而 PRCA 系统中执行各个应用时，请求模式通常是不同的。

因此，在 GDSF 算法的基础上，结合 LIRS 算法对象访问间隔的思想，考虑对象的最近访问时间和最近 K 次平均访问间隔。由于用时间作为参数因子，无法定性地分析用户请求的规律，参考性不强，因此记录最后一次访问对象的时间到当前时间之前用户请求的次数作为最近访问时间，设为 $Ln(i)$，初始值为 0；记录最近 K 次访问间隔内平均请求次数作为最近 K 次平均访问间隔，设为 $Gn(i)$，计算方法如式(4)所示，$g(i)$表示对象第 i 次访问和第 $i+1$ 访问的间隔，当对象只被访问过一次时，$Gn(i)$为 0。

$$Gn(i)=\frac{\sum_{1}^{k}g(i)}{K} \tag{4}$$

根据数理统计中均值-方差模型的简化形式，以对象的历史访问间隔(其均值为对象的历史访问间隔的平均值 $Gn(i)$)作为随机变量，以对象的最近访问间隔 $Ln(i)$为样本变量，来评估对象访问间隔波动率，即 $Ln(i)$与 $Gn(i)$之差的绝对值为 $Ln(i)$的标准差。因此，标准差越小，样本变量波动越小，对象被访问的概率越大；反之越小。可以得出缓存对象被访问的概率为 $P(i)$，其计算方法见式(5)。

$$P(i)=1/(1+|Ln(i)-Gn(i)|) \tag{5}$$

其中，$Ln(i)$代表对象 i 在最近一次访问到当前访问之间的请求次数；$Gn(i)$为对象 i 的历史平均访问间隔。

缓存对象被访问的概率 P 考虑了缓存对象的最近访问时间和最近 K 次平均访问间隔两个因素，利用数理统计的一般规律，使对象的历史访问情况和当前请求情况相结合，综合估计缓存对象的价值，增加了对循环访问模式的适应性。改进后的算法 GDSF-G 的缓存价值的计算方法如式(6)所示。

$$G(i)=L+F(i)*\left(\frac{Value(i)}{\log Size(i)}\right)*P(i) \tag{6}$$

GDSF-G 算法在原有算法的基础上增加了缓存对象的替换因子，更加全面地考虑了缓存对象被替换的参考因素。在多类型任务请求中，对于顺序访问模式，通过 $Ln(i)$ 可以尽快替换最近最久未访问的对象；对于循环访问模式，通过式(5)可以计算出对象被访问的可能性；再加上 GDSF 算法对传统访问模式的适应，可以看出本文提出的算法 GDSF-G 对任务的多请求访问模式具有较强的适应性。

3.2 三级结构的缓存模型及其优势

缓存结构模型直接决定了任务请求的效率和最终的缓存效果，缓存模型的数据结构采用如图 3 所示的索引结构模型，是由平衡二叉树 T 和链表队列 L 作为索引结构，对缓存对象进行两个方面的索引，即通过文件名和存储对象的存储价值进行索引，从而减少平均请求等待时间。

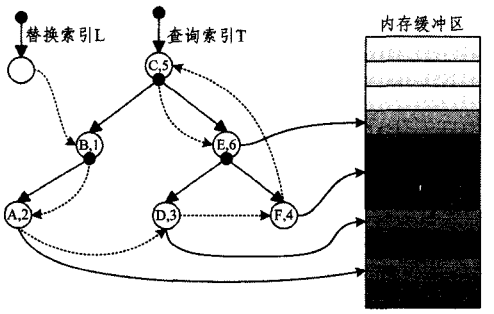


图 3 缓存模型的数据结构

在缓存结构模型中，查找索引是为了减少检查请求是否命中时消耗的索引代价，能够较明显地增加平均请求等待时间，它是以文件名为关键字建立平衡二叉树，通过平衡二叉树对文件进行索引，查询对象的时间复杂度为O(log2n)，在最坏的情况下，查找长度为平衡二叉树的高，即 log2N，N 为缓存区中对象的个数。初始化平衡二叉树的时间复杂度为O($n$log2n)，维护的时间复杂度为O(log2n)。替换索引是以缓存对象的存储价值为关键字构建链表队列，建队的时间复杂度为O(n)，需要替换出价值小的缓存对象时，就从队首元素进行替换，不需要进行相关的查询操作，因此替换复杂度为O(1)，替换后队列不需要维护，从而减少了维护缓存结构的开销。因此，通过建立缓存结构模型，针对不同需求建立索引机制，相比传统的顺序式或单索引式的缓存结构，能较大地提高索引效率，进而缩短平均请求等待时间。

三级结构的缓存模型如图 4 所示，是由 3 个集合组成的三级结构模型，分别为 F(Fast)集合，存储流行度比较高的对象；K(Keep)集合，存储新进入缓冲区或者被再次访问的对象；R(Replace)集合，存储即将被替换出缓存区的对象。R 集合的缓存结构模型如图 3 所示，而由于 K 和 F 集合中的对象不会被直接替换出缓存，因此它们的缓存结构模型是在图 3 的结构模型中减少了替换索引，只保留查询索引。

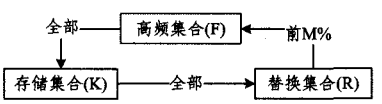


图 4 三级结构的缓存模型

传统的双队列缓存模型，大多数是为了弥补缓存替换算法的不足，避免“缓存污染”的现象。本文结合双队列机制和 CPU 三级缓存模型的三级结构缓存模型，除了具有双队列缓存模型的好处外，还间接考虑了缓存对象的流行度，三级结构中的 F 集合存储了所有缓存对象中缓存价值最高的对象，可以增加它们在缓存内的生存周期。三级结构缓存模型的动态运行机制描述如下：

```
Init(){ F=NULL,K=NULL,R=NULL;}
While(true){
  Input;  q;
  do{
    If( Buf. availible())
    {K. push(q);break;}
    else
      do{
        If(!R. isEmpty())
            deleteLowPriceNode();
        else
            {R=K;
             K=F;F=NULL;}
        }while(Buf. availible());
    }while(Buf. availible());
}
```

三级结构模型的建立主要从 3 个方面体现出它的优势：1)增加流行度比较高的对象在缓存中保留的时间，延长其生存周期。2)增加对多访问模式的适应性，保持对短暂集中式访问模式的适应性。3)由于平均请求间隔和最近请求间隔两个因素是动态变化的，在传统的缓存模型下，每次进行对象替换时，都要重新计算对象的缓存价值，并进行相关的比较，会产生较大的系统开销；而三级结构模型使用专用的替换集合 R，当缓存对象进入 R 集合时，计算其缓存价值，按照缓存价值的大小依次替换，从而降低了频繁计算和查询等操作带来系统开销。

3.3 缓存算法和预取机制的结合

预取技术是比较常见的提高缓存命中率的策略，如 Web 服务中对用户请求日志进行信息收集，通过数据挖掘得出用户兴趣模型，根据这个预测模型进行网页的预取，得到了很好的效果。然而 PRCA 系统中大多数应用并没有这么多的用户请求信息，因此，本文采取大文件后缀预取和缓存替换算法相结合的策略来实现对应用资源的缓存，建立 P_GDSF_G 算法。缓存算法和预取机制的结合，如图 5 所示，其中预取机制是通过并发机制实现的，当系统相关资源空闲时，就会启动预取机制。

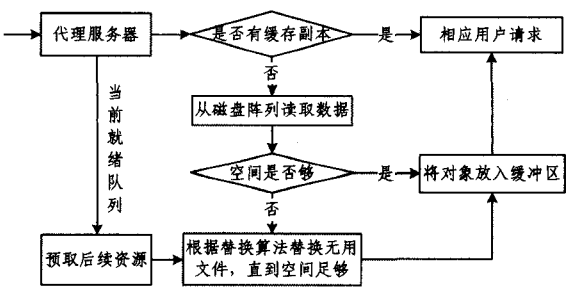


图 5 缓存算法和预取机制结合

缓存替换算法 P_GDSF_G 的整体流程具体描述如图 6 所示。其中，*File* 为请求文件 q 的大小；*Replace*(R)为替换 R 中的缓存块；$Q=First(M,q)$ 为读取文件 q 的前 M 块数据；

$K.push(q)$为把文件插入到 K 集合；$setParam(q)$为设置文件 q 的相关索引信息；$readyQ.push(q)$为把文件 q 索引到就绪队列，等待进程使用，如果 q 为大文件，开启线程，预取 q 的后续文件块。

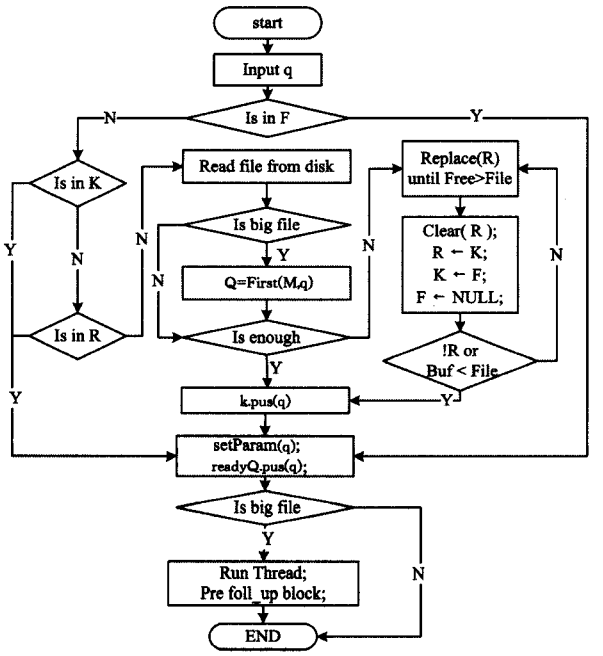


图 6　缓存替换算法 P_GDSF_G 的流程图

4　仿真实验

4.1　实验环境

为了模拟课题中 PRCA 系统的任务请求，特建立仿 PRCA 系统的物理环境，具体结构如图 7 所示。

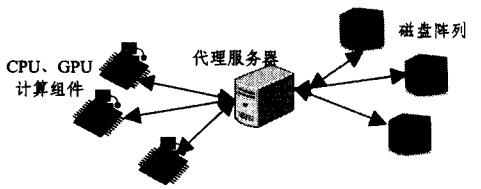


图 7　实验环境结构图

其中，磁盘阵列是由参数为希捷 1TB 7200 转的普通硬盘、东芝 500G 5400 转的移动硬盘和西部数据 2TB 7200 转的硬盘模拟的，用于存储实验数据。代理服务器是由参数为 Intel i7 处理器，内存扩充到 32G，64 位 Windows 7 操作系统的 lenovo 台式机模拟的，在此硬件环境下建立内存缓冲区。计算组件采用 3 台普通台式机模拟，分别根据各应用的访问模式并行地向代理服务器发起数据请求。

4.2　实验方案

基于本课题的研究需求和应用范围，实验数据取自时下比较典型的应用和高性能计算，分别为大文件（流媒体、视频点播等）、图像识别、密码破解 3 个应用，其资源请求特点为：

1）图像识别是通过目标图像或图片与现有视频图片进行比较，找出与目标图像相似或相近的资源信息，所涉及的资源大小不一，会涉及较多的访问模式，特别是循环访问模式；

2）密码破解应用依据相应数据字典相关信息，快速寻找正确的密码，所用资源比较少，但会有比较高的瞬间访问量；

3）大文件应用与传统的流媒体应用、视频点播应用基本相似，涉及到的资源比较大，而流行度比较高的资源具有比较高的访问量。

基于上述 3 种应用，设置多种资源请求模式，实现 LRU、LFU、GDS、GDSF 算法，在对象命中率、字节命中率和平均请求等待时间上，与本文算法 P_GDSF_G 进行对比。每种资源访问模式的资源量要超过实验所用的服务器内存大小。具体的实验数据集及来源如下。

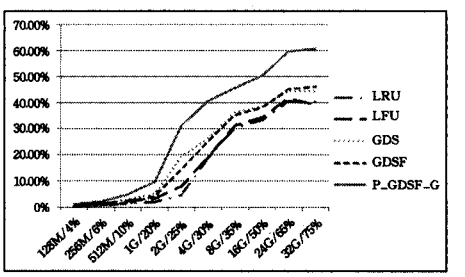
1）大文件以. mkv，. wmv 和. avi 等流媒体文件和 ISO 镜像文件为主，数据来自上海大学乐乎 pt 上的视频资源，平均文件大小为 1G，总体大小约为 1. 2T，涉及的请求模式主要为短暂集中式访问和顺序式访问。

2）模拟图像匹配中的图形、图像文件时使用爬虫工具——图片收割机，从淘宝山爬取得. jpg，. gif，. bmp 等文件，平均文件大小为 3M，总体大小约 2G，涉及请求模式主要为循环访问和全局概率性访问。

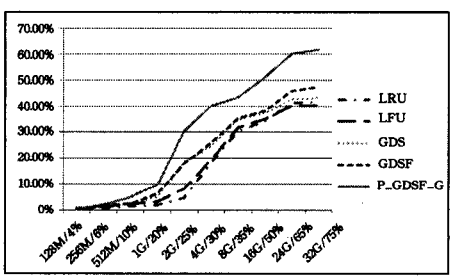
3）密码破解采用自动生成的文本文件，平均大小为 1M，总体大小约 1G，文件瞬间访问频率比较高，涉及的请求模式主要为典型的短暂集中式访问。

4.3　实验结论及分析

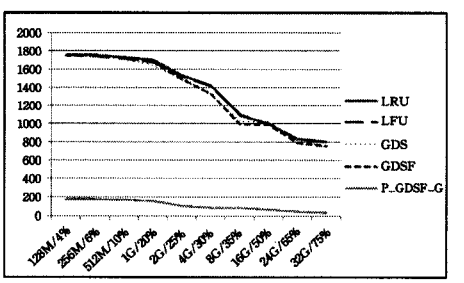
按照实验方案，采用多种请求模式交替进行，以缓存大小占总请求大小的比例为 x 轴，分别以对象命中率、字节命中率和平均请求等待时间为 y 轴，对 LRU、LFU、GDS、GDSF 和本文算法 P_GDSF_G 进行对比实验，得出的实验结果如图 8 所示。



(a)缓存算法的对象命中率对比



(b)缓存算法的字节命中率对比



(c)缓存算法的平均请求等待时间对比

图 8　P_GDSF_G 算法和几种常见算法的对比

虽然 P_GDSF_G 算法在单类型单请求模式的任务中的缓存效果没有经典算法理想，但通过图 8 的实验结果表明，P_GDSF_G算法在多类型多请求模式的任务中，有如下优势。

1)从图 8(c)中可以看出，P_GDSF_G 在平均请求时间上具有较大的优势。由于普通的缓存结构模型无法平衡缓存对象和待替换对象的效率，总体查询效率是 $O(n)$，而 P_GDSF_G算法使用链表和二叉平衡树建立多关键字索引，又结合三级结构缓存模型的动态调度，减少了资源缓存价值的频繁计算和排序的问题，虽然在每个节点上会有少量的空间消耗，但却使总体查询效率得到提高，时间复杂度为 $O(\log n)$，实验结果也证实了这一点。

2)对于流媒体等较大的请求对象，算法可以通过预取的策略维持较高字节命中率。

3)使用缓存对象的平均访问间隔和最近访问间隔两个特性来设计替换策略，使算法能适应更多的请求访问模式。虽然在特定请求模式下，P_GDSF_G 算法和其他专用算法的缓存效果相当，但对于本课题背景下的多类型多访问模式的应用需求，能够取得较好的缓存效果。

因此在请求多样化繁杂化的社会需求中，P_GDSF_G 算法具有较好的应用前景。

结束语 综上所述，缓存技术能够有效地提高计算机系统运算组件的使用率，从而提高系统的综合性能。本文在 GDSF 算法的基础上，引入缓存对象的平均请求间隔和最近访问间隔两个因素，增强了 GDSF 算法在不同任务请求模式下的适应性；建立缓存结构模型，用平衡二叉树和链表队列建立双索引机制，有效提高了查询效率，降低了替换代价；当任务的请求资源为大文件时，用并发的方式对后续文件块进行预取，使预取机制和缓存替换策略进行有机的结合。在 PRCA 系统的背景下，通过与 LRU、LFU、GDS 和 GDSF 算法进行对比实验，证明了 P_GDSF_G 算法能够适应多种访问模式，能够减少平均请求等待时间，提高缓存对象的命中率和字节命中率。

参 考 文 献

[1] 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)：20-23
Tian Xiao-bo，Chen Shu-yu. Proxy cache replacement algorithms for streaming media based on smallest cache utility[J]. Journal of Computer Applications，2007，27(3)：20-23

[2] 叶剑红，叶双. 基于混合模式的流媒体缓存调度算法[J]. 计算机科学，2013，40(2)：61-64，83
Ye Jian-hong，Ye Shuang. Dynamic Scheduling Algorithms for Streaming Media Based on Hybrid Content Delivery Network [J]. Computer Science，2013，40(2)：61-64，83

[3] 陈起，李松年. P2P 流媒体点播系统缓存策略的研究和实现[D]. 上海：复旦大学，2008
Chen Qi，Li Song-nian. The research and implementation of P2P streaming media on-demand system caching strategies [D]. Shanghai：Fudan University，2008

[4] 韩向春，田玉根. 基于预测的 Web 缓存替换算法[J]. 计算机工程与设计，2010，31(1)：110-113
Han Xiang-chun，Tian Yu-gen. Web cache replacement algorithm based on the prediction[J]. Computer Engineering and Design，2010，31(1)：110-113

[5] 王超宇，李静梅. 缓存替换策略研究[D]. 哈尔滨：哈尔滨工程大学，2012
Wang Chao-yu，Li Jing-mei. Research on Cache Replacement Strategies[D]. Harbin：Harbin Engineering University，2012

[6] 赵英杰，肖侬. 网络存储服务器缓存替换策略研究[D]. 长沙：国防科技大学，2010
Zhao Ying-jie，Xiao Nong. The Buffer Cache Replacement Policies in the Network Storage Servers[D]. Changsha：National University of Defense Technology，2010

[7] Dan A，Towsley D. An Approximate Analysis of The Lru And Fifo Buffer Replacement Schemes[J]. ACM Sigmetrics Performance Evaluation Review，1990，18：143-152

[8] Song Jiang，Feng Chen，Zhang Xiao-dong. CLOCK-Pro：an effective improvement of the CLOCK replacement[C]// Proceedings of USENIX Annual Technical Conference，2005：323-336

[9] Chierichetti F，Kumar R，Vassilvitskii S. Similarity caching [C]// Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. Providence，Rhode Island，USA，2009：127-136

[10] Johnson T，Shasha D. 2Q：A Low Overhead High Performance Buffer Management Replacement Algorithm[C]// Proceedings of the 20th VLDB Conference. Santiago，Chile，1994：439-450

[11] IBM. IBM enterprise storage server[OL]. http://www.storage.ibm.com/hardsoft/products/ess/ess.htm

[12] Megiddo N，Modha D. ARC：A Self-Tuning，Low Overhead Replacement Cache[C]// Proceedings of the 2003 Conference on File and Storage Technologies (FAST'03). San Franciso，CA，2003：115-130

[13] Patil J B，Pawar B V. GDSF#，A better caching algorithm that optimizes both hit rate and byte hit rate in web proxy servers [J]. International Journal of Computer Science & Applications，2008，5(4)：1-10

[14] 李静梅，王朝宇. 一种改进的自适应时钟算法[J]. 计算机工程，2012，38(20)：286-289
Li Jing-mei，Wang Chao-yu. An Improved Algorithm for Adaptive Clock[J]. Computer Engineering，2012，38(20)：286-289

[15] Janapsatya A，Ignjatovic A，Peddersen J，et al. Dueling CLOCK：Adaptive cache replacement policy based on the CLOCK algorithm[C]// Proceedings of the Conference on Design，Automation and Test in Europe. Dresden，Germany，2010：920 - 925

[16] 刘磊，熊小鹏. 最小驻留价值缓存替换算法[J]. 计算机应用，2013，33(4)：1018-1022
Liu Lei，Xiong Xiao-peng. Least cache value replacement algorithm[J]. Computer Applications，2013，33(4)：1018-1022

[17] Kedzierski K，Moreto M，Cazorla F J，et al. Adapting cache partitioning algorithms to pseudo-LRU replacement policies[C]// 2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS). Atlanta，GA，2010：1-12