引用格式：沈志，江博闻，江波，林涛．延迟命中场景下基于学习方法的缓存性能优化[J]．中国传媒大学学报（自然科学版），2023，30(03)：78-84.
文章编号：1673-4793(2023)03-0078-07

延迟命中场景下基于学习方法的缓存性能优化

沈志[1]，江博闻[1]，江波[1*]，林涛[2]
（1.上海交通大学，上海 200240；2.中国传媒大学，北京 100024）

摘要：传统的缓存算法大多基于简单的统计信息进行内容替换，在绝大部分场景下都和离线最优算法有着较大的性能差距。当前基于机器学习来设计高效的缓存策略基本都假设请求的大小相等，忽略了真实场景中请求的大小往往不等且大小变化的范围较大。由于传输速度已接近极限，但网络的吞吐量依旧在持续增长，延迟命中这一因素表现得更加突出。延迟命中下的命中率和传统算法中的命中率并不等价。本文同时考虑请求大小不等和延迟命中两个因素，提出ARC-learning+算法，研究内容主要包括：(1)对请求内容变大小场景，分析了缓存内容大小对于缓存性能的影响，并提出基于概率准入策略的ARC-learning+算法。在真实数据集上实验结果表明，在请求内容变大小场景下，修改后的缓存替换策略非常接近已有算法的最优内容命中率。(2)对延迟命中场景，ARC-learning+算法采用的排序函数综合考虑请求缺失所消耗的时延和请求的流行度等因素。实验结果表明，ARC-learning+算法在流行度变化较快情况下的时延提升明显优于已有的其他缓存算法。在其他流行度变化情况下也非常接近已有算法的最优时延性能。

关键词：基于学习；非平稳流行度；内容变大小；延迟命中
中图分类号：TP393.2　**文献标识码**：A

A learning-based method to optimize cache performance with delayed hit

SHEN Zhi[1], JIANG Bowen[1] , JIANG Bo[1*], LIN Tao[2]
(1.Shanghai Jiao Tong University, Shanghai 200240, China; 2.Communication University of China, Beijing 100024, China)

Abstract: Most traditional caching algorithms rely on basic statistical data for content replacement, creating a significant performance discrepancy in comparison to offline optimal caching algorithms. However, due to improvements in CPU performance, cache strategies based on machine learning have been developed in recent years to enhance cache performance. These studies typically assume that requests are uniform in size, without taking into account the wide range of sizes that occur in real-world scenarios. Moreover, with network throughput continuing to increase as transmission speed approaches its limit, the factor of delayed hit becomes more critical. Consequently, the hit rate under delayed hit is not equivalent to traditional algorithms. In this paper, we consider the cache problem with variable object size and delayed hit and propose the algorithm named ARC-learning+ . In summary, our research includes: (1) for the variable object size, this paper analyzes the impact of variable object size on cache performance. In the ARC-learning+ algorithm, a probabilistic admission policy is designed, which takes

基金项目：国家自然科学基金面上项目(62072302)；地区科学基金项目(62262018)；“媒体融合与传播国家重点实验室（中国传媒大学）”开放课题(SKLMCC2021KF011)
作者简介(*为通讯作者)：沈志(1998-)，男，硕士研究生，主要从事缓存技术研究。Email：shangjiao4313888@sjtu.edu.cn；江波(1983-)，男，博士，副教授，主要从事系统性能评测与优化研究。Email：bjiang@sjtu.edu.cn

into account the object size. Experimental results on real data sets show that the modified cache replacement strategy is very close to the existing optimal cache algorithm. (2) for the scenario with delayed hit, the ranking function adopted by the ARC-learning+ algorithm comprehensively considers the delay consumed by the missing request and the popularity of the request. Experimental results show that the performance of the modified sorting function algorithm is better than other existing caching algorithms when the popularity changes quickly. It is also very close to the optimal delay performance of the existing algorithm under other prevalence changes.

Keywords: learning-based; non-stationary popularity; variable object size; delayed hit

1 引言

随着视频音频等各类应用的发展,传统的源服务器直接向用户提供服务的部署模式已经不能满足当前应用对于快速获取数据的需要。在上世纪90年代人们引入了网络缓存的思想,即将一部分经常被访问到的数据存储在距离用户较近的缓存单元中,而用户对数据的访问会被直接路由到这些缓存单元上。考虑到大部分用户往往偏好于请求相同的数据,因此即使在存储容量较小的缓存中存储少部分内容也能够很大地提升用户体验。内容分发网络(Content Delivery Network, CDN)通过将内容缓存到靠近用户的服务器上并快速地将这些内容交付给用户来提高网络性能。一个大型的CDN系统,比如Akamai运营的CDN,每天从位于全球100多个国家的1500多个网络的17万多个服务器上服务数万亿用户请求[1]。

CDN服务器位于用户和广域网(WAN)之间。每当用户请求当前未缓存的内容时,CDN必须通过网络向更上一级的服务器获取该内容。当CDN服务器向上一级服务器获取内容时候,向更上一级服务器获取内容会造成用户端感受到更长的传输时延,传输过程中也会带来额外的带宽和内存开销。已有的研究试图去寻求一些方法以最小化这些开销,这些开销在研究中通常用内容命中率(Object Hit Ratio, OHR)[1][2][3][4],时延(Latency)[3][5]来衡量,即可以直接从边缘服务器获取的内容占比,内容字节占比,和累计时延。较高的内容命中率意味着更多的请求能在更短的时间内得到请求数据,较高的字节命中率则意味着向边缘服务器向上一级请求数据量较少,骨干网络传输压力较低。累计时延则是指从用户请求到达到服务器将内容发送给用户过程中所产生的时间开销。

很多音视频应用,往往要求较少的卡顿以保证用户体验。设置边缘缓存尽可能高地提高内容命中率,减少累计时延,可以有效提高此类应用的用户端体验。另一方面提高缓存命中率也可以减少边缘服务器向上一级服务器请求时产生的流量开销,从而降低数据传输对骨干网络的影响。相比于数据存储中心,边缘节点存储容量较小,无法存储大量的请求。因此如何设计高效的缓存替换算法来提高缓存命中率已经成为研究的重点。现如今许多研究专注于将基于学习的方法应用到缓存中。这些算法通过对内容的流行度进行预测,通过存储流行度最高的内容以获得较高的内容命中率。对于大多数边缘服务器而言,其目标都是最大化内容命中率以此为更多的用户请求提供服务。在请求内容变大小场景下,缓存体积较大的内容会使得缓存能换成的内容个数减少,综合考虑请求内容的大小和流行度可以为更好地优化请求内容大小变化场景下的内容命中率。此外对一些处于高吞吐量环境下的服务器,命中率和用户端累计时延的优化并不等价。如何在服务端存在一定延迟的场景下,最小化用户端累计时延对于提高用户体验有着重要意义。

本文综合考虑请求的大小可变和服务端存在延迟命中的场景,提出基于学习的缓存替换算法ARC-learning+。总体来说,本文的主要贡献如下:

①针对请求内容大小变化的场景,本文基于马尔科夫链对概率准入策略的最优阈值进行求解,并在ARC-learning+的LRU删除模块中添加准入策略。为了更好地优化请求内容变大小场景下的缓存性能,本文同样优化了基于学习的删除模块所使用的删除策略。实验测试发现,ARC-learning+算法的命中率在各种不同的数据集上都能接近最优算法,具有很好的鲁棒性。

②针对延迟命中场景,本文综合考虑请求的流行度、写入时延、请求大小这些指标,并基于此提出排序函数来决策每一时刻缓存中需要存储的内容。本文在请求流行度变化速度不同的数据集上测试算法的累计时延。实验结果表明相比于已有的其他缓存算

法,本文所提出的算法在各种流行度变化情况下都能达到最低时延。

2 研究背景介绍

2.1 请求内容变大小

传统的内存缓存,其缓存对象是内存页,缓存内容的大小相等。对于等大小情况,已有研究表明,离线Belady算法可以达到最好的缓存替换效果[5]。但CDN服务器的边缘缓存其缓存对象往往是音视频或者网页,内容大小从几十KB到几百MB不等。这意味着对传统的无准入的缓存替换算法,准入一个大小较大的请求往往需要删除大量大小较小的请求。准入请求内容的流行度比删除的任何一个内容的流行度都要高,但却很可能并没有比所有被删除内容的流行度之和更高。因此频繁地存储大小较大的请求会使得服务器在未来一段时间内请求命中率下降。删除较多的大小较小的请求也会使得缓存中产生大量的空闲空间,造成缓存利用率下降。因此在请求内容大小变化的场景下设计缓存替换策略时,请求内容的大小和请求内容流行度这两个因素都应该考虑。

2.2 延迟命中

绝大多数缓存相关研究都采用命中率作为衡量缓存算法性能的主要指标。但Nirav Atre[6]指出在当今的网络环境下,由于高吞吐量服务器往往存在着较大的服务端写入时延,对时延的优化与命中率的优化并不等价。之前大多数研究计算时延时只考虑了命中或者未命中的情况。此外,由于请求命中所带来的时延要远小于请求缺失时所带来的时延,所以最小化时延可以等价为最大化命中率的问题。但当今的CDN环境中,请求的高并发性导致服务端的写入时延越来越不能被忽略。当用户向边缘服务器请求数据时,如果服务器存储了该数据,则发生命中,时延可以忽略不计。如果边缘服务器未存储该数据,边缘服务器便向更高一级的存储单元请求缺失的数据,该过程产生的时延较大。如果在该过程中同样的请求再次达到,那么用户端的感知时延要大于请求命中时的时延。一个简单的例子是假设T时刻用户向服务器请求一个未被缓存的内容A。此时服务器向上一级服务器请求内容,经过T_1时间后该内容到达服务器,再经过T_2的时间内容被成功写入。而在这段长为T_1+T_2的时间段内,内容A又发生了K次请求。此时的K次请求实际均为延迟命中,对于这些请求,用户的感知时延均超过命中时的时延。但在以往的研究中,这部分时延都被忽略掉了。考虑到CDN边缘服务器所存储的内容大小不同,网络吞吐量不同,所引起的时延也不同,这些额外的时延给最小化时延问题带来了挑战。

3 ARC-learning算法介绍和改进

3.1 ARC-learning算法介绍

ARC-learning[2]是基于LRB(Learning Relaxed Belady)[1],为适应各种流行度变化设计的算法,该算法有效降低了流行度变化较快场景下,LRB算法对窗长的依赖,使LRB算法在窗长较长时也能在流行度变化较快的场景下,达到较好的性能。算法由基于XGBoost(Extreme Gradient Boosting)的删除模块,LRU(Least Recently Used)删除模块,在线获得训练集的采样模块,和一个自调节模块组成。请求间隔时间(Inter-Request Times, IRTs)和指数衰减计数器(Exponentially Decayed Counters, EDCs),分别表示过去一段时间内容请求的时间间隔和过去特定时间段内内容的请求个数。ARC-learning算法框架如图1所示。

ARC-learning处理过程中,每当内容请求到达时候,算法对请求内容的特征进行更新,首先更新IRTs,再更新EDCs。每次请求到达时候,算法会随机对记忆窗内的一个内容打上采样标记,并记录下此时的采样时间。当该请求再次到达时候,算法会观察请求内容上是否存在采样标记。如果有,则遍历所有的采样标记时间。此时记录的该样本的特征即为记录的IRTs,EDCs,采样标记时间到上一次请求时间的时间间隔,请求内容大小。而其标签为对应时刻的下一次到达时间,即为当前请求时间和采样时间的差值。由此请求上每有一个采样标签,算法将相应获得一个训练样本。而对于那些在记忆窗内没有到达的请求,算法将其标签设置为记忆窗大小的两倍,以此表示这个特征的请求将来流行度非常低,缓存应该及时予以删除。在收集到足够训练集后,算法会对模型进行更新。相比于LRB,ARC-learning添加了一个自调节模块以适应流行度快速变化的情况。当服务器需要进行缓存替换时,模型会根据自调节策略给出来的结果采用LRU或模型对缓存中当前存储的内容进行删除。

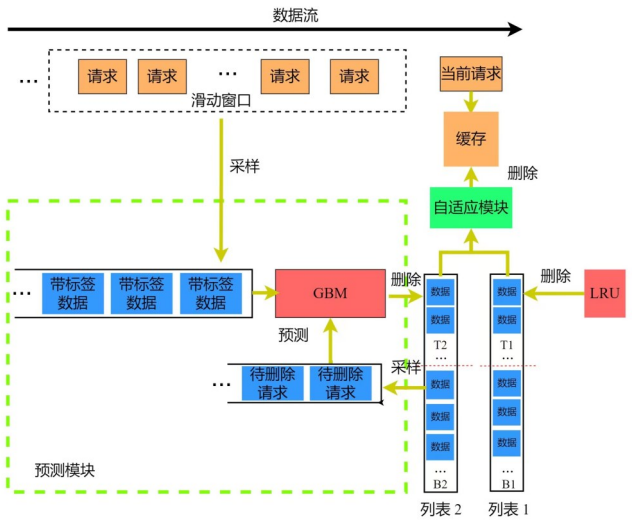

图 1 ARC-learning 缓存替换框架

3.2 基于变大小场景的算法性能改进

在请求内容大小变化的场景下，引入准入策略可有效提高缓存性能[4]。除了设置合适的准入策略，在进行缓存替换时候也应当综合考虑内容大小和流行度影响。因此，我们提出 ARC-learning+算法，通过添加基于请求内容大小的准入策略和在删除时考虑请求内容大小进行删除两个方向优化变大小场景下的缓存性能。ARC-learning+进行删除时主要操作如下：

（1）在采样时候首先将当前请求内容放到待删除集合当中。

（2）在模型预测得到所有待删除内容的下一次到达时间后，综合考虑预测下一次到达时间和请求内容大小进行删除。

相比于之前的策略，改进后的策略在进行缓存删除时，会优先删除缓存中体积较大的内容。改进后的算法在需要缓存删除的时候，会对新引入内容也进行预测。如果请求内容即为删除内容，则此时服务器拒绝缓存此次请求内容，不进行任何替换操作。

而针对 LRU 的改进，本文对变大小场景修改了 LRU 算法的删除策略。等大小场景下 LRU 会将上一次到达时间最远的内容删除，而在变大小场景下，本文将其修改为删除上一次到达时间和请求内容大小乘积的数值最小的内容。

本文还采用了文献[4]对于 LRU 链表的改进方法，该方法为 LRU 链表添加了一个概率准入策略，通过马尔科夫链对概率准入策略的最佳准入阈值进行求解。

通过将请求到达近似为泊松过程，结合请求 i 的到达速度、请求大小、准入概率等信息，我们可以求解请求 i 在缓存中的概率。在已知过去一段时间所有请求信息的情况下，我们可以离线得到 OHR 和准入阈值 c 之间的关系。通过求解 OHR 最大时 c 的取值，我们就可以找到最优准入阈值 c。

3.3 基于延迟命中场景下的算法性能改进

由于服务端请求排队和传输的影响，真实场景下用户请求在边缘服务器除了命中和未命中外，还存在延迟命中的问题[6]。在忽略了这部分延迟命中影响的情况下，由于请求未命中时候的时延远大于命中时候的时延，最小化时延的实质就是最大化缓存的命中率。近年一些研究慢慢意识到在实际场景中，请求未命中时发生的时延往往不能忽略，因此导致最大化命中率和最小化时延并不等价[6]。本节使用了文献[5]在优化时延时所使用的排序函数对 ARC-learning 进行改进。文献[5]对请求未命中时候由完全未命中和延迟命中产生的累计时延 $\overline{D_i}$ 和不进行缓存时候请求到达时产生的期望时延 $\overline{D_i}$ 进行了分析并指出为了最小化时延，缓存应当缓存那些在请求未命中期间产生累计时延较高和每个请求平均产生时延较低的内容。累计时延较高意味着缓存未命中时候会产生更多时延，而平均每个请求产生的时延较低意味着请求在未命中时候，向上一级服务器取内容期间产生的延迟命中请求较多。文献[5]基于这两点设计了排序函数。该排序函数可以有效提高延迟命中场景下缓存算法在时延上的性能。对于内容 i，流行度记为 λ_i，内容 i 在未命中时，从上一级服务器取内容所产生的时延记为 L_i，其排序函数定义如下：

$$\tilde{f}(i)=\frac{Aggregated\ latency}{Mean\ latency\ for\ each\ request}=\frac{D_i}{\overline{D_i}}=\frac{\lambda_i L_i(1+\lambda_i L_i)}{2+\lambda_i L_i} \tag{1}$$

考虑到变大小场景下应当尽可能缓存体积较小的内容以为更多的用户提供服务，上述公式可进一步优化为：

$$f(i)=\frac{\tilde{f}(i)}{s_i}=\frac{\lambda_i L_i(1+\lambda_i L_i)}{(2+\lambda_i L_i)s_i} \tag{2}$$

考虑到 XGBoost 模型预测结果是请求的下一次到达时间，而在预测准确的情况下请求下一次到达时间的倒数即为流行度的无偏估计。对内容 i，模型的预测结果为 p_i，因此得到的排序函数为：

$$f(i)=\frac{\lambda_i L_i(1+\lambda_i L_i)}{(2+\lambda_i L_i)s_i}=\frac{L_i(p_i+L_i)}{(2p_i+L_i)s_i p_i} \tag{3}$$

基于上述分析得到的排序函数，不同于之前缓存在需要删除时删除预测结果最大的内容，ARC-

learning+算法会在模型预测后计算出对应内容i的排序函数值$f(i)$,并删除$f(i)$值最小的内容。

4 实验分析和性能评价

本节所使用的数据集来自wiki_2019数据集[1],该数据集包含591K个请求内容和1000K个请求,也含有请求内容的大小信息。基于该数据集我们可以验证ARC-learning+算法在变大小请求上的性能。对于请求内容i,其时延定义为:

$$L_i = \frac{s_i}{trasmission\ speed} + Latency' \qquad (4)$$

在实际中,传输速度transmission speed和排队时延$Latency'$可以通过监控网络传输速度和请求处理时的平均排队时延得到。此外,我们在wiki_2007[7]数据集和Youtube[8]数据集上进行了额外的补充实验。wiki_2007数据集和wiki_2019数据集来源于相同的应用网站,但数据集收集的时间不同。Youtube数据集则来源于校园网代理服务器。这两个公开数据集都缺少请求内容的大小信息,因此我们为wiki_2007和Youtube数据集人工添加请求内容的大小信息,以便于测试算法在真实场景的不同流行度变化情况下的性能。

本章对比的缓存替换算法除了LA-Cache[5]和一些传统缓存替换算法如LRU,LFU,ARC外,还对比了基于LRB的变体算法,包括LRB-S,LRB-SC和LRB-C算法。LRB-S算法,也即LRB-Size,是变大小场景下最大化内容命中率的LRB算法,该算法删除下一次到达时间和请求内容大小乘积最大的内容。LRB的开源代码通过LRU优先删除超出记忆窗长度没有到达内容,而在本文所使用的数据集中,该方法会导致算法在缓存容量较大时,一直使用LRU算法进行删除。而在变大小实验下,LRU的性能往往很差。为了更好地反应单独使用预测模块时候算法的性能,我们删除了这部分代码,并将修改后的不考虑内容大小和考虑内容大小算法分别命名为LRB-C和LRB-SC。仿真时对于ARC-learning+和LRB变体算法,窗长均设置为128K,采样个数设置为64,训练集大小设置为64K。无延迟情况下,3.3节中排序函数值恒为0,因此LA-Cache并不适用于无时延场景下优化内容命中率。在该场景下,本文修改LA-Cache算法,使其删除对象为内容大小和流行度倒数乘积最大内容。

4.1 基于真实变大小数据集的缓存性能分析

首先测试了在wiki_2019数据集[1]下,缓存容量对不同缓存算法性能的影响,实验结果如图2所示。图2表明,在wiki_2019数据集上,当缓存容量小于1GB时,ARC-learning+,LRB-SC,LA-Cache的算法命中率基本相当。当缓存容量超过1GB时,LRB-SC表现出了最好的缓存性能,略高于ARC-learning+,远高于其他算法。

其次,测试了在不同的缓存策略下,系统时延对用户端累计时延的影响,实验结果如图3所示。图3表明,在wiki_2019数据集上,ARC-learning+的时延性能略差于LA-Cache,高于其他算法。随着系统写入时延的增长,传输时延相比于固定的写入时延几乎可以忽略不计,此时每个内容未命中时产生的时延几乎相同。该场景下,较高的内容命中率往往意味着较低的时延。可以发现时延较低的LA-Cache,ARC-learning+,LRB-SC算法都是该缓存容量(1G)下,内容命中率较高的算法。

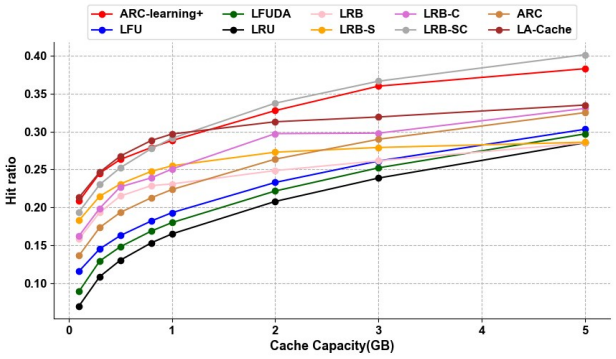

图2 请求内容变大小的wiki_2019数据集下不同缓存策略命中率随缓存大小变化情况

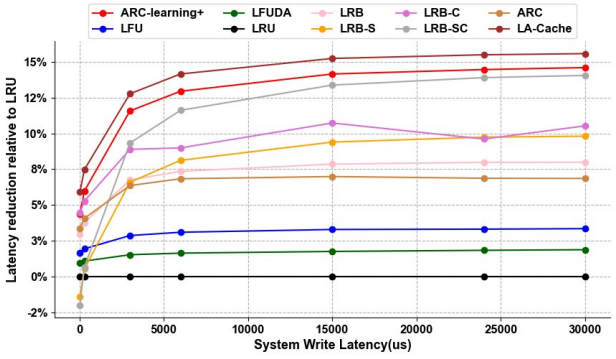

图3 请求内容变大小的wiki_2019数据集下缓存用户端时延随写入时延变化情况

4.2 基于人工数据集的缓存性能分析

本节将基于wiki_2007数据集和Youtube数据集来研究在请求大小不同的情况下,缓存容量和系统写入时延对缓存性能的影响。本文通过对wiki_2019数据集中的内容进行采样获取对应内容大小后,将其赋值给wiki_2009和Youtube中的随机内容,以此为这两

个数据集添加请求大小的信息。两个数据集中的请求的流行度变化快慢不同，基于这两个数据集的实验可以测试ARC-learning+算法在不同流行度变化场景下的鲁棒性。首先测试了不同数据集下缓存大小对ARC-learning+的性能影响，实验结果如图4、图5所示。在wiki_2007数据集上，当缓存容量小于1GB时，ARC-learning+算法的性能和LA-Cache接近，高于LRB-SC算法的内容命中率。在缓存容量大于2GB时，ARC-learning+算法下缓存命中率略低于LRB-SC算法，相比其他算法有着最高约6%的提高。

而在Youtube数据集上，当缓存容量小于2GB时，ARC-learning+命中率略低于LRB-S算法。当缓存容量大于2GB时，ARC-learning+的内容命中率和LRB-S，LRB-SC算法相当。另外，在Youtube数据集上，LA-Cache在各种缓存容量设置下的内容命中率都和其他算法有着较大差距。

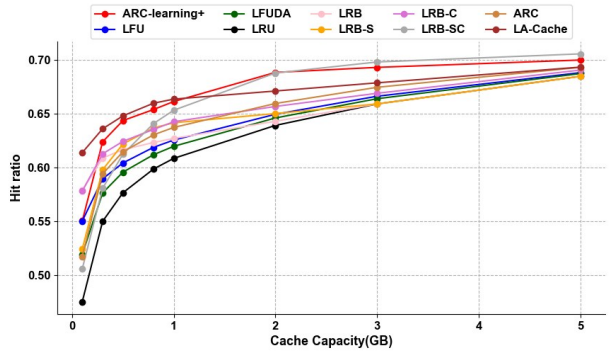

图4 请求内容变大小的wiki_2007数据集下不同策略命中率随缓存大小变化情况

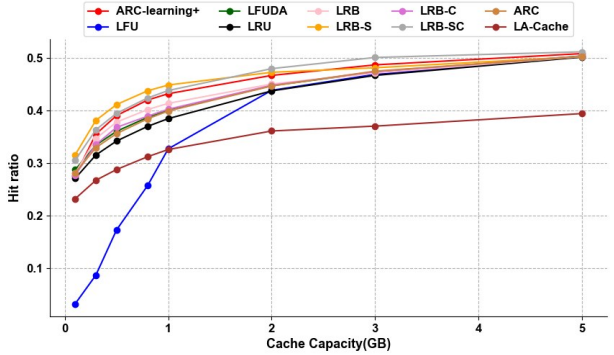

图5 请求内容变大小的Youtube数据集下不同缓存策略命中率算缓存大小变化情况

两个数据集上用户端累计时延随系统固定写入时延变化如图6、图7所示。在wiki_2007数据集上，ARC-learning+的时延性能和LA-Cache算法相当。而考虑请求内容大小进行删除的LRB-S和LRB-SC算法的时延性能都非常差，尤其在固定写入时延低于5ms时，时延性能比LRU最多下降了50%。而不考虑内容大小的LRB和LRB-C算法则取得了相对较好的时延性能，在各种系统时延设置下相比于LRB-S和LRB-SC算法至少降低了10%的时延。而在Youtube数据集上，当系统写入时延超过3ms时，ARC-learning+的时延性能和LRB-S相当，优于包括LA-Cache在内的其他所有算法。

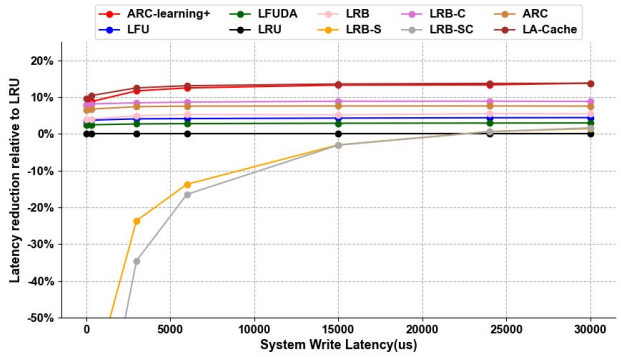

图6 请求内容变大小的wiki_2007数据集下用户端累计时延随系统写入时延变化

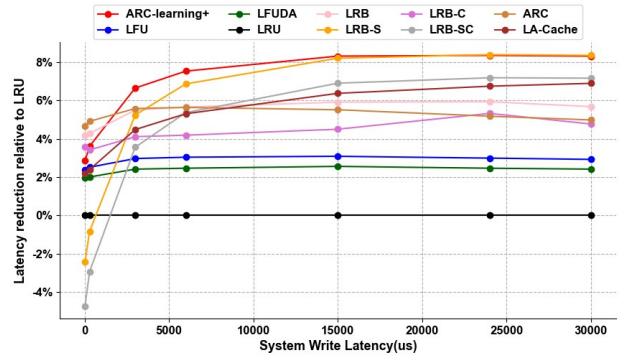

图7 请求内容变大小的Youtube数据集用户端累计时延随系统写入时延变化

5 结论

本文综合考虑了缓存系统中请求大小可变和延迟命中的场景。对请求大小可变的情况，本文引入了概率准入策略并优化了删除模块进行改进。对于延迟命中，本文优化了基于模型进行删除时的排序函数。基于真实数据集和人工数据集，本文测试了不同流行度变化下，ARC-learning+算法在内容命中率和累计时延两个指标上的性能。实验结果表明，LRB-SC算法在本文所使用的数据集上都能很好的对内容命中率进行优化，但当系统时延较小时，考虑内容大小进行缓存会使得时延性能明显下降。LA-Cache算法在流行度稳定的情况下能达

到最优的时延性能,但当内容流行度变化较快,不适合基于过去一段时间内容的请求间隔对流行度进行计算时,其性能明显差于其他算法。而本文所提出的ARC-learning+算法在大多数实验设置和数据集上和各种算法的最优内容命中率和时延相比有最多10%的提升,表现出了很好的算法鲁棒性。

参考文献(References):

[1] Song Z, Berger D S, Li K, et al. Learning relaxed belady for content distribution network caching [C]//17th USENIX Conference on Networked Systems Design and Implementation(NSDI '20),2020:529-544.

[2] Shen Z, Jiang B, Wang X, et al. ARC-learning: a self-tuning cache policy under dynamic popularity [C]// IEEE 8th International Conference on Computer and Communications (ICCC), 2022: 610-615.

[3] Paschos G, Iosifidis G, Caire G. Cache optimization models and algorithms [J]. Foundations and Trends® in Communications and Information Theory, 2020, 16(3-4): 156-345.

[4] Berger D S, Sitaraman R K, Harchol-Balter M. AdaptSize: orchestrating the hot object memory cache in a content delivery network [C]//14th USENIX Conference on Networked Systems Design and Implem-entation (NSDI '17),2017: 483-498.

[5] Yan G, Li J. Towards latency awareness for content delivery network caching[C]// USENIX Annual Technical Conference (USENIX ATC 22),2022: 789-804.

[6] Atre N, Sherry J, Wang W, et al. Caching with dela-yed hits [C]//Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication,2020: 495-513.

[7] Urdaneta G, Pierre G, Van Steen M. Wikipedia work-load analysis for decentralized hosting[J]. Computer Networks, 2009, 53(11): 1830-1845.

[8] Zink M, Suh K, Gu Y, et al. Watch global, cache l-ocal: YouTube network traffic at a campus network: measurements and implications [C]//Multimedia Computing and Networking, 2008, 6818: 35-47.

编辑:王谦