文献综述

——第2组 M201773173 谌美玲

# 前言

在Web服务开发中，服务端缓存是服务实现中所常常采用的一种提高服务性能的方法。DRAM键值缓存对于减少应用程序延迟和吸收Web应用程序中的大量数据库请求负载至关重要。例如，Facebook有几十个应用程序访问存储在memcached [1]内存缓存中的数百TB数据[2]； 较小的公司使用外包多租户内存缓存这种经济高效的方式，来提升SQL数据库性能。高访问速率和缓慢的后端数据库性能意味着，降低高速缓存未命中率将直接转化为端到端应用程序性能的大幅提升。所以如何提高DRAM键值缓存的命中率则是我们为了提升web应用程序性能所必须考虑的问题，也是该论文研究的问题。

专题涉及的学科范围涵盖计算机系统结构，计算机操作系统，数据中心技术等，综述引用文献的时间范围是2013年至2016年。

内存容量是一定的，所以缓存命中率受限于内存容量，这是硬性限制。同时，由于不同web应用会有不同性能要求的工作负载，而缓存服务器上通常是多用户多租户同时发出请求的，所以如何进行缓存分配也是提升整体缓存命中率的重要一环；当缓存被占满之后，新的请求出现，选择什么样的键值对被替换才能尽量保证被替换出的键值对不会短时间内重新请求，导致命中失败，这也是为了提升整体缓存命中率必须考虑的问题。本综述即针对上述主题和具体方面，分别按时间列举当前研究现状，分析利弊，整理出本领域清晰的发展脉络。

# 研究现状

首先关于缓存分配的问题。最优情况下，我们希望缓存分配时能给每个应用其刚好需要的“量”。然而，当工作负载发生变化时，通过缓存系统提供的有限信息，并不能知道当前工作负载需要被分配多少内存才能获得原本的性能，即刚好需要的“量”。Trausti Saemundsson等人[3]提出了一个轻量级的在线分析器——Mimir存储系统，该分析器动态地在分布式内存缓存中添加或删除内存资源，从而实现对成本和性能的影响，用一种不需要脱机调优的方式回答缓存性能的“刚好的量”的问题，找到最优内存配置的值。

常用的内存缓存系统如memcached使用通用的先来先服务设计，简单地固定大小内存分配，忽略了应用程序不同的需求和实际的工作负载，这些简单的默认策略会导致显著的性能损失，未命中率高。Harshad Kasture等人[4]考虑到不同性能要求的工作负载，提出了Ubik动态分区技术，同时满足低延迟要求的应用程度和计算密集型批处理应用程序的高性能。Asaf Cidon等人[5]主要针对低延迟性要求的web应用，提出了高速缓存控制器Dynacache，同样也是动态调整原则，通过分析应用程序和动态裁剪内存资源，以及调整驱逐替换政策，大幅提高web应用的命中率。一年后，针对同样的问题，Asaf Cidon等人[6]又提出一种新的算法——Cliffhanger，它是在内存缓存服务器（常用的如memcached）上运行的轻量级迭代算法，它基于动态变化的工作负载逐步优化应用内部的资源分配，该算法尤其对性能临界的增加有显著优势。

在应用程序之间动态重新分区内存时，也有不少研究人员提出采用日志结构设计，可以提供比传统缓存更好的命中率。Abutalib Aghayev等人[7]针对移动设备中最大化缓存命中率要求的浏览器应用，提出了一个日志结构的浏览器缓存，并在Android浏览器上实现，提高浏览器应用的缓存命中率，提升性能。Stephen M. Rumble等人[8]把日志结构的内存管理方法应用到基于DRAM的存储系统中，抛弃了使用内存效率低下的传统内存分配机制，使用日志结构达到80-90%的内存利用率。同时日志结构化内存的RAMCloud实现使用两级清理策略，节省磁盘带宽，并在高内存利用率时将性能提高6倍。清洁剂与正常操作同时运行，并采用多个线程来隐藏大部分清洁成本。该文作者也是小组交流论文的第二作者，论文方法也深深受该文算法的影响。

然后再来关注一下缓存替换策略。常用的键值缓存系统，如Memcache和Redis，它们都是用最近最少使用（LRU）置换政策，或者一个近似的选择，来决定出被替换出缓存的键值对。但是如果重新计算缓存值显著变化的成本后，会发现这些替换政策都不是最好的选择。Conglong Li[9]提出了一种新的成本意识的替换政策——GD-Wheel，这是第一个固定的时间实施GreedyDual替换策略的。Nathan Beckmann等人[10]提出了一种廉价实用的EVA实现，并利用马尔可夫决策过程证明该替换策略在某些合理的简化下是最优的。小组交流论文的Need和Rank结合的替换算法也是一种很巧妙的设计[12]。

最后关于单租户和多租户的问题。在数据中心中，缓存既能提供低IO延迟，又能减少后端网络和存储的负载。但它们没有被设计成多租户。Ioan Stefanovici等人[11]设计的租客和工作量感知系统Moirai，允许数据中心供应商控制分布式缓存基础设施，Moirai可以帮助缓解缓存基础设施的管理和实现各种目标，如提高整体资源利用或提供租户隔离和QoS保证。小组交流的论文同时考虑了单租户和多租户的情况。

# 总结

近年来，关于数据中心的缓存服务器的研究状况十分火热，尤其是针对web应用这种低延迟要求的应用程序。但大部分论文都只是从单一着力点进行改进，或者是缓存分配策略，或者缓存置换策略。我们小组交流的论文从两部分总和改进，采用日志结构设计，在应用程序之间动态重新分区内存，并计算应用程序的Need和应用程序的Item的Rank等级，实现一种新的仲裁机制和清除机制。并从实验数据上验证了优于Asaf Cidon等人的Dynacache和Cliffhanger算法等等[12]。

# 参考文献

1. Fitzpatrick B. Distributed caching with memcached[J]. Linux Journal, 2004, 2004(124):págs. 72-76.
2. Ongaro D, Rumble S M, Stutsman R, et al. Fast crash recovery in RAMCloud[C]// ACM Symposium on Operating Systems Principles. ACM, 2011:29-41.
3. Bjornsson H, Chockler G, Saemundsson T, et al. Dynamic performance profiling of cloud caches[C]// Symposium on Cloud Computing. ACM, 2013:59.
4. Kasture H, Sanchez D. Ubik: efficient cache sharing with strict qos for latency-critical workloads[J]. Acm Sigplan Notices, 2014, 49(4):729-742.
5. Cidon A, Eisenman A, Alizadeh M, et al. Dynacache: dynamic cloud caching[C]// Usenix Conference on Hot Topics in Cloud Computing. USENIX Association, 2015:19-19.
6. A. Cidon, A. Eisenman, M. Alizadeh, and S. Katti. Cliffhanger: Scaling performance cliffs in web memory caches. In 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI16), pages 379–392, Santa Clara, CA, Mar. 2016.
7. Aghayev A, Desnoyers P. Log-structured cache: trading hit-rate for storage performance (and winning) in mobile devices[C]// The Workshop on Interactions of Nvm/flash with Operating Systems and Workloads. ACM, 2013:7.
8. Rumble S M, Kejriwal A, Ousterhout J. Log-structured memory for DRAM-based storage[C]// Usenix Conference on File and Storage Technologies. USENIX Association, 2014:1-16.
9. Li C, Cox A L. GD-Wheel: a cost-aware replacement policy for key-value stores[C]// Tenth European Conference on Computer Systems. ACM, 2015:5.
10. Beckmann N, Sanchez D. Bridging Theory and Practice in Cache Replacement[J]. 2015.
11. Stefanovici I, Thereska E, O'Shea G, et al. Software-defined caching: managing caches in multi-tenant data centers[C]// ACM Symposium on Cloud Computing. ACM, 2015:174-181.
12. Cidon A, Rushton D, Rumble S M, et al. Memshare: a Dynamic Multi-tenant Key-value Cache[C]//2017 {USENIX} Annual Technical Conference ({USENIX}{ATC} 17). USENIX} Association}, 2017: 321-334.