Dynamic Performance Profiling of Cloud Caches

Hjortur Bjornsson
University of Iceland

Gregory Chockler
Royal Holloway, University of London

Trausti Saemundsson
Reykjavik University

Ymir Vigfusson
Reykjavik University

In-memory object caches, such as `memcached`, are critical to the success of popular web sites, such as Facebook [3], by reducing database load and improving scalability [2]. The prominence of caches implies that configuring their ideal memory size has the potential for significant savings on computation resources and energy costs, but unfortunately cache configuration is poorly understood. The modern practice of manually tweaking live caching systems takes significant effort and may both increase the variance for client request latencies and impose high load on the database backend.

Contributions. We provide an efficient online algorithm to estimate how an LRU cache would perform using a *different* memory allocation, continually exposing a *hit rate curve* as a function of space (Figure 1). Our method is lock-free and compatible with modern multithreaded cache servers, such as `memcached`.

Approach. For a cache of size n, the challenge is to generate a hit rate curve for cache sizes ranging from 0 to $2n$. First, to predict how a cache would perform beyond the current allocation of n, we track metadata for n additional dataless elements, so-called "ghosts" [1]. While technically a cache miss, a hit on a ghost provides information about how the larger cache allocation would fare under the same workload [4].

To track statistics, our method splits the LRU stack into a list of variably sized *buckets*. The first bucket represents the top of the LRU stack and the last bucket represents the tail. Whenever a cache hit occurs, the element e causing the hit is moved to the first bucket. The stack distance of e can then be estimated by summing up the number of elements in buckets in front of where e was in the list. Next, the hit rate curve is updated (Figure 2). The elements are then aged to maintain about the same number of elements per bucket. For aging, we can trade off performance (constant time vs. linear in number of buckets) for estimation accuracy. If accuracy is favored, a global average stack distance is maintained and only certain elements are aged.

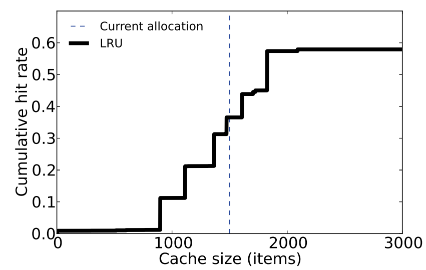

Figure 1: ***Hit rate curve.*** *The normalized cumulative cache hit rate achieved for cache sizes different from the current allocation.*

Results. The faster algorithm, ROUNDER, achieves over 96% accuracy measured by the mean average error of the hit rate curve on a wide variety of cache workloads, compared to over 99% for the slower one. We implemented and evaluated ROUNDER in `memcached` and found negligible throughput degradation on standard benchmarks. We conclude that online generation of hit rate curves are both useful for provisioning and monitoring, and can be made practical for large cache systems.

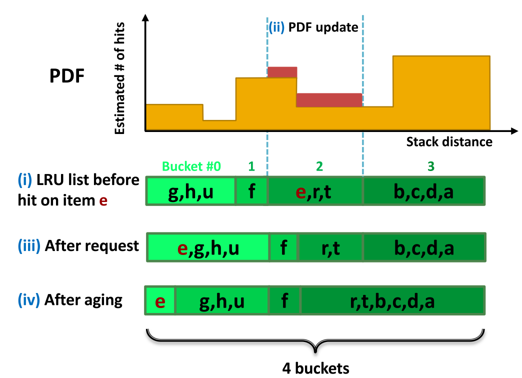

Figure 2: ***The* ROUNDER *algorithm.*** *Steps to update the hit rate curve and bucket lists of the* LRU *stack when item e is hit in the cache.*

SoCC'13, 1–3 Oct. 2013, Santa Clara, California, USA.
ACM 978-1-4503-2428-1. http://dx.doi.org/**10.1145/2523616.2527081**

References

[1] M. R. Ebling, L. B. Mummert, and D. C. Steere. Overcoming the network bottleneck in mobile computing. In *Proceedings of the 1st Workshop on Mobile Computing Systems and Applications*, WMCSA '94, pages 34–36, Washington, DC, USA, 1994. IEEE Computer Society.

[2] B. Fan, D. G. Andersen, and M. Kaminsky. MemC3: Compact and concurrent MemCache with dumber caching and smarter hashing. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, NSDI '13, pages 385–398. USENIX Association, 2013.

[3] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, et al. Scaling Memcache at Facebook. In *Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation*, NSDI '13, pages 385–398. USENIX Association, 2013.

[4] R. H. Patterson, G. A. Gibson, E. Ginting, D. Stodolsky, and J. Zelenka. Informed prefetching and caching. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, SOSP '95, pages 79–95, New York, NY, USA, 1995. ACM.