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 multi-threaded 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.

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.
References


