Prefetching in a General-Purpose Caching Engine

DAIYAAN ARFEEN and SOPHIE SMITH

Modern web services rely on caching to improve performance, and many of them are client-facing making latency reduction an important goal. Prefetching is a common method for hiding latency; however, effective prefetching requires knowledge of future requests. While future requests are often unknown, they can be predicted with some accuracy. We utilize a cost-benefit analysis that relies on bounds of an object's request times to decide which objects to prefetch into the cache. We show that our method substantially improves hit ratios even when requests cannot be predicted perfectly. Our method can easily be adopted by any application using a key-value cache as it only relies on the application providing predictions for future access patterns (which do not need to be certain or precise). Furthermore, our method is agnostic to other parts of the caching system, such as eviction policy.

1 INTRODUCTION

1.1 Motivation

Web services rely on high hit-ratio caches to reduce client-observed latencies by orders of magnitude. With high enough hit-ratios, caching can also improve server throughput, allowing servers to fulfill many more requests-per-second than otherwise possible[6]. However, web services also often suffer from high degrees of churn and bursty traffic[4]. In this environment, having a larger window to fetch from the backend than the moment a demand miss occurs or using the knowledge of how popularity of objects is about to change can significantly reduce miss rates and latency. Prefetching addresses both of these issues, but in both cases requires knowledge of future requests. Previous work in prefetching assumes perfect knowledge of future requests through application provided hints[12], but this is not possible for a server that fulfills requests from thousands of users concurrently. The best that can be assumed in this setting is prediction of future accesses to some imperfect accuracy. Thus, prefetching must take into account both the predictions of future requests themselves as well as their uncertainty or likelihood of being incorrect. If done successfully, prefetching, even with imperfect knowledge, can improve hit ratios and mitigate slowdowns from churn and traffic bursts.

Throughout this work, we provide three main contributions. We designed a cost-benefit model applied to the generalized caching framework, CacheLib. We also extended the CacheLib implementation to support any arbitrary prefetching implementation which adheres to our framework presented below. We evaluate the performance of this simulated prefetching models (both perfect knowledge and imperfect knowledge) to estimate performance in practice.

1.2 CacheLib Caching Engine

CacheLib is a general-purpose caching framework employed at Facebook to provide generalized, efficient support for numerous applications, including the social-graph cache, CDNs, and the storage-backend cache. CacheLib is proposed as a unified caching framework intended to prevent unnecessary duplication and specialization of caching functionality for different applications at Facebook. Although developing an independent, specialized cache for each application enables workload-specific hardware optimizations and consistency support, the development overhead caused by distinct systems outweighs these performance benefits. In practice, the overlap in system functionality can be simplistically and efficiently abstracted into a common API. CacheLib manages this abstraction by exposing a straightforward API of necessary cache functions, as well as supporting advanced features, i.e., dynamic resource usage, hybrid caching. To generalize the implementation to various workloads and storage mediums, CacheLib relies on the application for