

# Deconstructing the Inefficacy of Global Cache Replacement Policies

Rahul V. Garde Samantika Subramaniam Gabriel H. Loh

Georgia Institute of Technology  
College of Computing  
{garde,samantik,loh}@cc.gatech.edu

## Abstract

*In a conventional two-level cache hierarchy, L1 cache hits do not propagate to the L2 cache; as a result, the L2 cache only observes a “filtered” memory access stream. A frequently accessed address may hit in the L1, but since these accesses never make it to the L2, the corresponding copy in the L2 will “decay” with respect to its replacement policy state and may eventually get evicted. Previous studies have advocated the use of global replacement policies where the L1 access information propagates to the L2 to maintain a replacement policy state that is consistent with the overall global memory access stream. We first attempt to duplicate previously reported results on global cache replacement policies. Despite the intuitive explanation for why a global scheme should work, our experimental results show that the performance potential of global replacement is very limited. We deconstruct the problem with reuse-distance analysis and show that only under very specific reuse-distance profiles will a program be able to benefit from global replacement. Our experiments include the evaluation of multi-core shared caches, inclusive cache hierarchies, and a wide spectrum of cache sizes and associativities; we show that global replacement fails to provide significant performance benefits for any of these scenarios.*

## 1. Introduction

Cache replacement policies considerably impact the performance of many applications. Since these policies choose which lines to evict from the cache, they have a direct effect on miss rates which is usually directly related to performance. Over the decades, there has been a large body of work on basic replacement policies [16] and several recent proposals have shown that combinations of basic policies can improve performance by adapting to different program and phase behaviors [7, 15, 19]. All these research proposals, however, only make the replacement decisions *locally*; that is, the second level cache (L2) is oblivious to the state of the first level cache (L1), and vice-versa. Caches employing local replacement policies are easy to implement, but recent research suggests that limiting the replacement policy’s scope may have drawbacks [21].

The basic problem is that not all levels of the cache hierarchy see the same access patterns. Since the L2 is only accessed when we miss in the L1, the L2 cache only observes a “filtered” version of the global memory access pattern. As processors integrate L3 caches [20], the disconnect between the observed access patterns at different levels of the cache hierarchy could become even more dramatic. In the context of cache replacement policies, this means that even if a line is accessed repeatedly in the L1 cache, the line might still make its way to the least-recently-used or LRU position of the L2 cache and thus risk being evicted (assuming a LRU replacement policy, but this observation holds for most replacement policies).

An alternative to such local replacement decisions is a system that exposes the entire, unfiltered, global access stream to all caches in the hierarchy. Zahran described and evaluated several global replacement policies [21]. We believe that it intuitively makes sense that an L2 cache with a global view of the unfiltered L1 access stream can make better replacement decisions because it has more accurate and complete information about the true, program-level, memory access patterns. While Zahran’s preliminary results showed that a global replacement policy does not provide a substantial performance improvement, he described several characteristics of programs as well as potential processor configuration settings where a global policy would be beneficial. These include newer benchmark suites with larger memory working set sizes, different cache sizes and associativities, inclusive cache hierarchies with different line sizes at the different cache levels, and multi-core processors with shared L2 caches.

In this work, we explore these other possible scenarios, and disappointingly show that even in all of these other cases, global replacement policies still fail to show any promise. We then take a step back to try to explain the observed results. Using analysis based on memory access reuse distances, we describe the necessary conditions for a global replacement policy to provide benefit. We then quantify the actual reuse-distance histograms of our benchmarks and show that nearly *none* of the programs ever exhibit such behaviors, and due to the rather stringent set of conditions, we conclude that it is very unlikely that global replacement policies will ever be useful (unless there are some drastic















