Problem
- Contention for shared resources is a major bottleneck for multicores—especially for mixed workloads.
- Modern processors implement instructions to manage caches, but they are typically unused.

Solution
- We propose a classification that allows us to predict application interference.
- We propose a method to automatically insert cache management instructions into misbehaving software.

Profile Run
We sample an application’s memory accesses using a lightweight sampler. Using StatStack, we create a stack distance profile from the sample.

Memory Access Analysis
Using the stack distance profile we can determine if the data referenced by an instruction will be reused after it is installed in the cache.

Compiler Feedback
We automatically disable caching for instructions that do not reuse data by injecting `prefetch` instructions into the compiler output.

Miss Ratio Curves
We calculate the new miss ratio and replacement ratio curves for the managed application using the stack distance profile and information from the analysis.

Classification
We define the cache sensitivity of an application, δ, to be the difference in miss ratio when running in isolation, \( r_p \), and when running only in the private cache, \( r_p \). Using δ and \( r_p \), we can classify an application’s cache behavior.

Better Mixed Workload Performance
The performance of mixed workloads containing applications classified as Gobblers & Victims or Cache Gobblers, is significantly improved by cache management. The reduced impact can be attributed to decreases in replacement ratio.

Classification of Managed Applications
Most applications classified as wasting cache resources get a nicer behavior when we apply cache management. Since it is replacements that cause most interference, we base the new classification on replacement ratios instead of miss ratios.