Comparing Cache Models

Taylor Foxhall
CS 320

December 4, 2015

1 Purpose

In this assignment we look to evaluate the effectiveness of different methods of caching. We'll use direct caches, set-associative caches, fully-associative caches, and multiple write-through policies and next-line prefetching.

2 Direct Caching

Direct caching is consistently the worst estimator, although it seems to pull ahead for the largest of cache sizes (16 KB). This is what I expected, except in the last case.

3 Set-Associative Caching

This was the most general model, with all other models of the cache being some derivation or special case of the Set-Associative cache's functionality. For small caches, this model usually performed much better than the Direct Cache equivalent, though it was still not the best.

4 Fully-Associative Caching

While its simulated implementation was much less efficient than the set associative caching, fully associative caches performed better than the largetst way Set-Associative cache. However, the number of ways had a huge increase from the largest used in the Set-Associate model, and resulted in just a tiny return in hit counts. Taken into account for the loss in efficiency associated with scanning the entire cache for a block, this model isn't necessarily the best option.

4.1 Hot-Cold Replacement

Since in reality it is hard to simulate the LRU policy on the hardware efficiently, Hot-Cold approximation can be used, and as it can be seen, the replacement policy is less accurate than true LRU replacement, but not enough that it wouldn't be worth the gain in efficiency.

5 Write-through Policy

This model was plainly worse than the general Set-Associative model, except in one case. This isn't surprising, since we don't update the cache on a miss and we generally write to memory in a local area, if we miss once, we'll probably miss again soon. Although, we don't really lose a lot of our accuracy, write misses must not be a big contributor. And I will note, while it's not simulated in this program, a lot of write-misses will result in a lot of direct writes to memory, which will slow down a program in general.

6 Next-line Prefetching

This method seemed to be the best method we could get. It generally performed better than the general Set-Associative cache and frequently beat the accurate but inefficient Fully-Associative cache. The most optimal configuration of this type was the 16-way version which consistently performed better most other configurations, only being occasionally beaten out by the 16 KB direct cache oddly enough.

6.1 Prefetch on Miss

Preftching only on a miss gave good results, generally better than the Set-Associative cache, but like with the write-on-miss Set-Associative cache, this version of a prefetching cache was still slightly less accurate.