Victim Cache and Skewed Associative Cache

02.10.2019

Group Members

Ayush Mangal
Harshit Verma
Hemant
Hemil Panchiwala
Kunal Mohan
Nalin Prabhath
Piyush Gupta

Abstract

Increase in processing speed in computers has always been a thirst in history. Different types of ISA's came through, each having its own characteristics. For eg : RISC concept, which uses a set of simple and limited instructions and is quite fast. Further, higher clock frequencies in computer and parallel/pipelining instructions has also contributed tremendously in increasing the processing speed. Though, instruction time has reduced a lot by various technologies, the main memory access time haven't changed much and this creates a latency between instructions processing and memory access, resulting in slow overall processing.

Caches solves this latency problem by acting as a storage for chunks of most accessed data (by using concept of temporal and spatial locality). Two most commonly used caches today are Direct mapped cache and Associative cache. Both of these differ in hit ratio and access time. Direct mapped cache has an edge over Associative cache in terms processing time but lags behind in terms of hit rate. Used togetherly, they leads to Set-associative cache having a good hit rate as well as less access time.

In our project, we have compared two new types of cache - the Skewed Associative Cache and Victim Cache.

Skewed Associative Cache has a better behaviour than Set-associative caches. Typically a two-way skewed-associative cache has the hardware complexity same as that of two-way set-associative cache, yet simulations show that it exhibits approximately the same hit ratio as a four-way set associative cache of the same size. The skewed-associative cache uses a multi-bank association where a single address can be mapped to different cache banks using different hashing functions. This leads to association of a single address to different locations and reduces increases the hit ratio.

Victim Cache is basically a fully associative cache, comparatively smaller in size than L1 cache and placed behind L1 cache. It holds the evicted block because of conflict miss from L1 cache. So rather than evicting the memory block, we store it in another cache, thus increasing the hit ratio and increasing the speed.