ECE7995
Caching and Prefetching Techniques in Computer Systems

In-class questions for Presentation:

Improving direct-mapped cache performance

by the addition of a small fully-associative cache and prefetch buffers
Victim caching uses a very small fully-associative caches to buffer victims of (conflict) misses of a first-level (direct-mapped) caches. It dynamically increases the associativity of a cache of small associativity when temporal locality is present.

In victim caching, when and what data are loaded into the small fully-associative cache? Compared with loading missed data (miss cache), what is the benefit of this method?

It is mentioned in the paper that “In all cases the first-level caches are assumed to be direct-mapped, since this results in the faster effective access time.”. Could you elaborate on this? By dynamically increasing associativity, what is the benefit relative to the approach of statically increasing cache associativity?

It’s mentioned in the paper that “The relatively large access time for main memory in turn requires that second-level cache line sizes of (larger size) 128B or 256B are needed.” Could you explain this?
• For following piece of code, assume address distance of X[0] and y[0] is a multiple of the cache size, explain why x and y conflicts on a direct mapped cache, and how victim cache can remove them.

```c
int x[4], y[4];
int sum = 0;
for (i = 0; i < 4; i ++)
    sum += x[i] + y[i];
```

• What is compulsory miss? Can this type of miss be avoided? If yes, how?

• Why are stream buffers better than tagged prefetch and prefetched on miss in taking advantage of the memory bandwidth available in pipelined memory systems for sequential references?

• With the increase of cache line size, the reduction in misses provided by stream buffer (either single data stream or multi-way stream buffers) falls. Why?