Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers

Norm Jouppi

In Context

- At the time, CPU performance was really beginning to pull away from DRAM performance
  - Increased interest in memory system performance
- Mark Hill had just introduced the 4-Cs as way of categorizing cache misses
  - Conflict
  - Compulsory
  - Capacity
  - Coherence
- Single-chip processors were really coming into their own
  - Increased pressure on area, so direct-mapped caches were very desirable.
- It was also before the "Quantitive Approach"

Goals

- Increase effectiveness of direct-mapped caches without spending much area.

- In the retrospective, Norm says he was looking at each class of miss individually.

Motivation

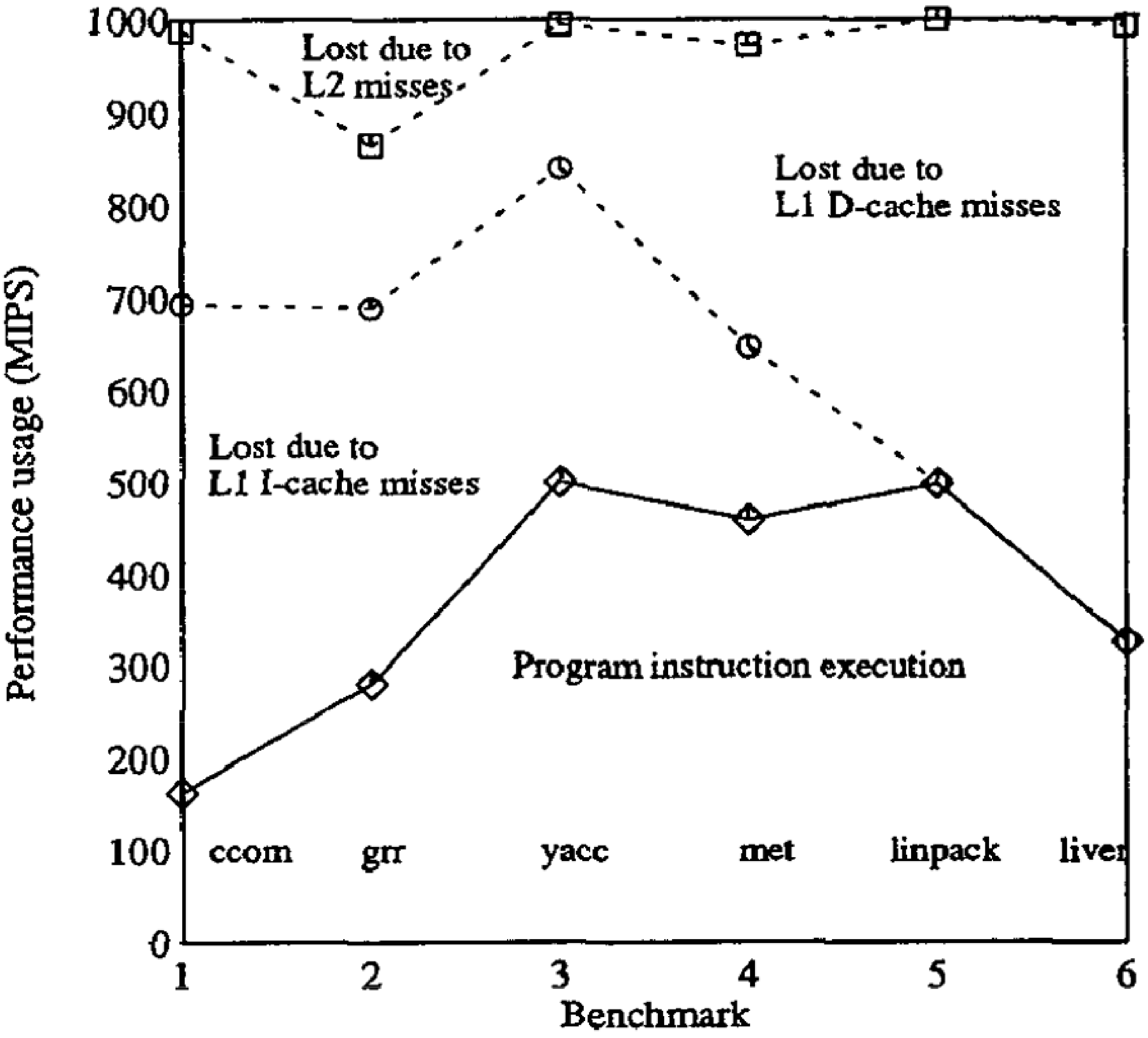

Figure 2-2: Baseline design performance

Idea 1: Miss Buffers

- When you fill a line, store a second copy in the miss buffer.
- If you need the data again, it'll be close at hand.

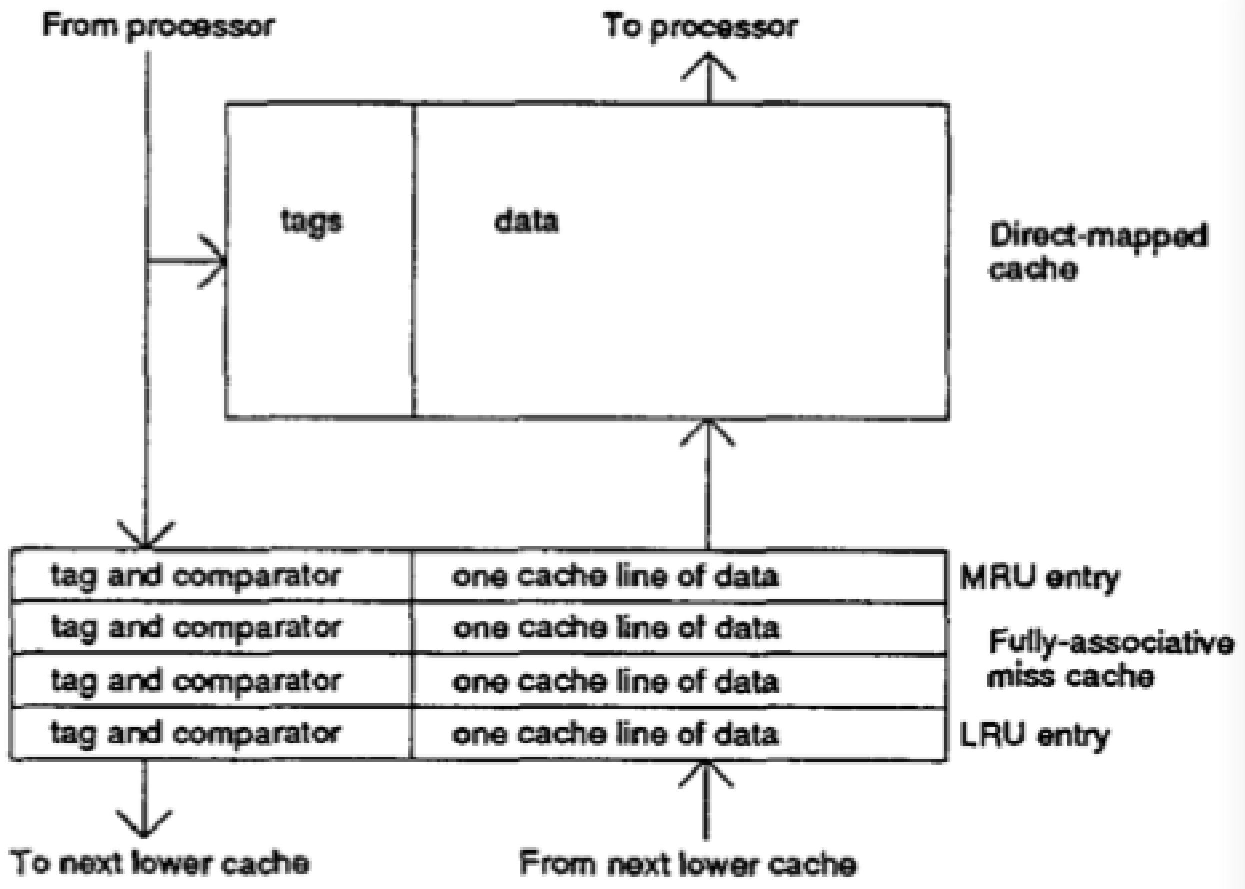

Figure 3-2: Miss cache organization

Miss Buffer Performance

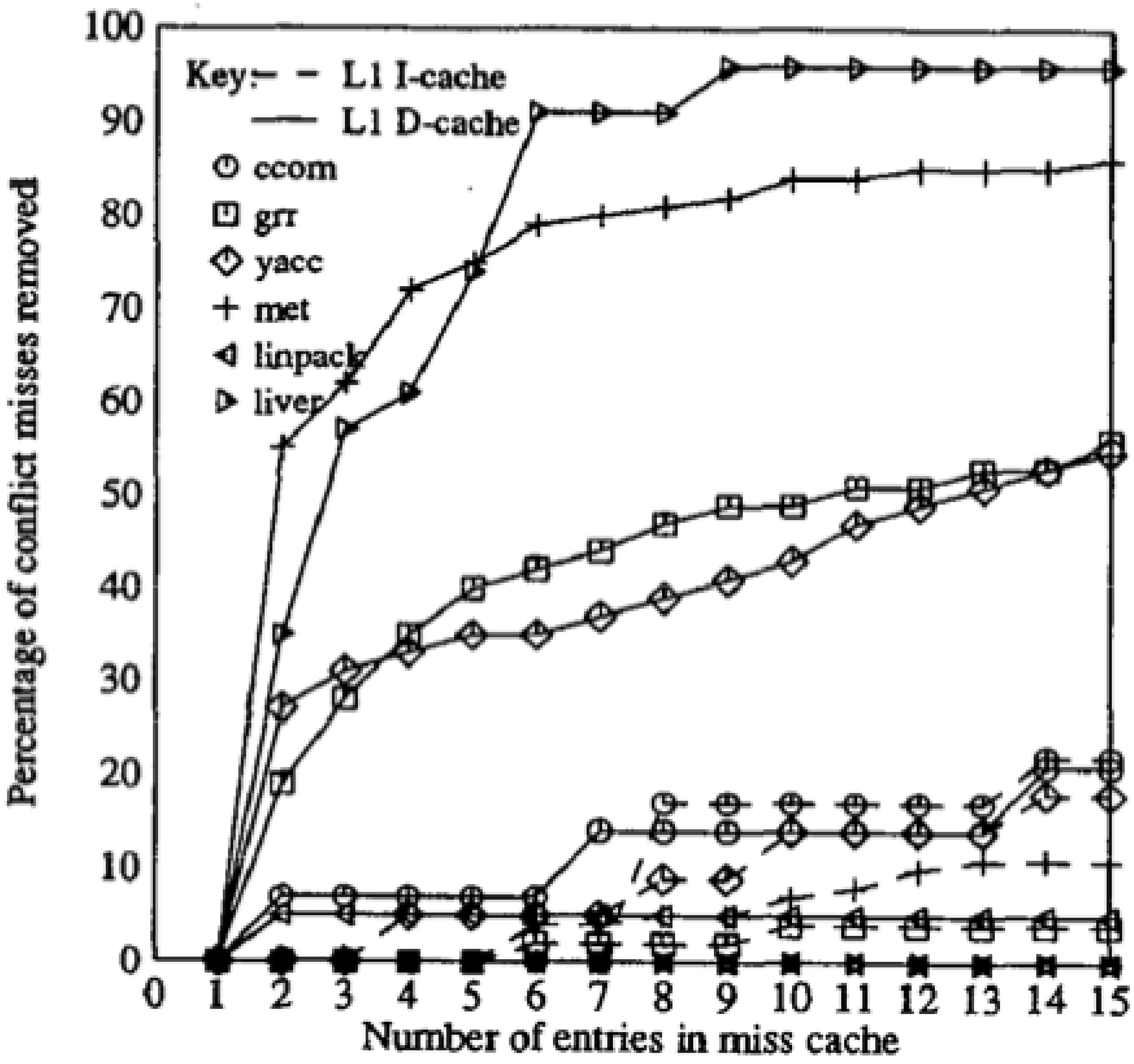

Figure 3-3: Conflict misses removed by miss caching

Miss Buffer Gains

- The miss buffer only address conflict misses
  - So it does better when there’s lots of them

Problems w/ the Miss Buffer

- It wastes space
  - It's contents are always replicated in the cache
  - It needs to be at least two entries to have any benefit.
- If the conflicting area is larger than the miss buffer, the miss buffer is of no use.
  - we should be able to get some benefit from it, since it is extra space
- The miss buffer is sort of pessimistic. It assumes that we are going to have a conflict on the data.
  - Let's be optimistic.

The Victim Cache

- Similar to the miss cache, but only put data in the victim cache when there's actually a miss.

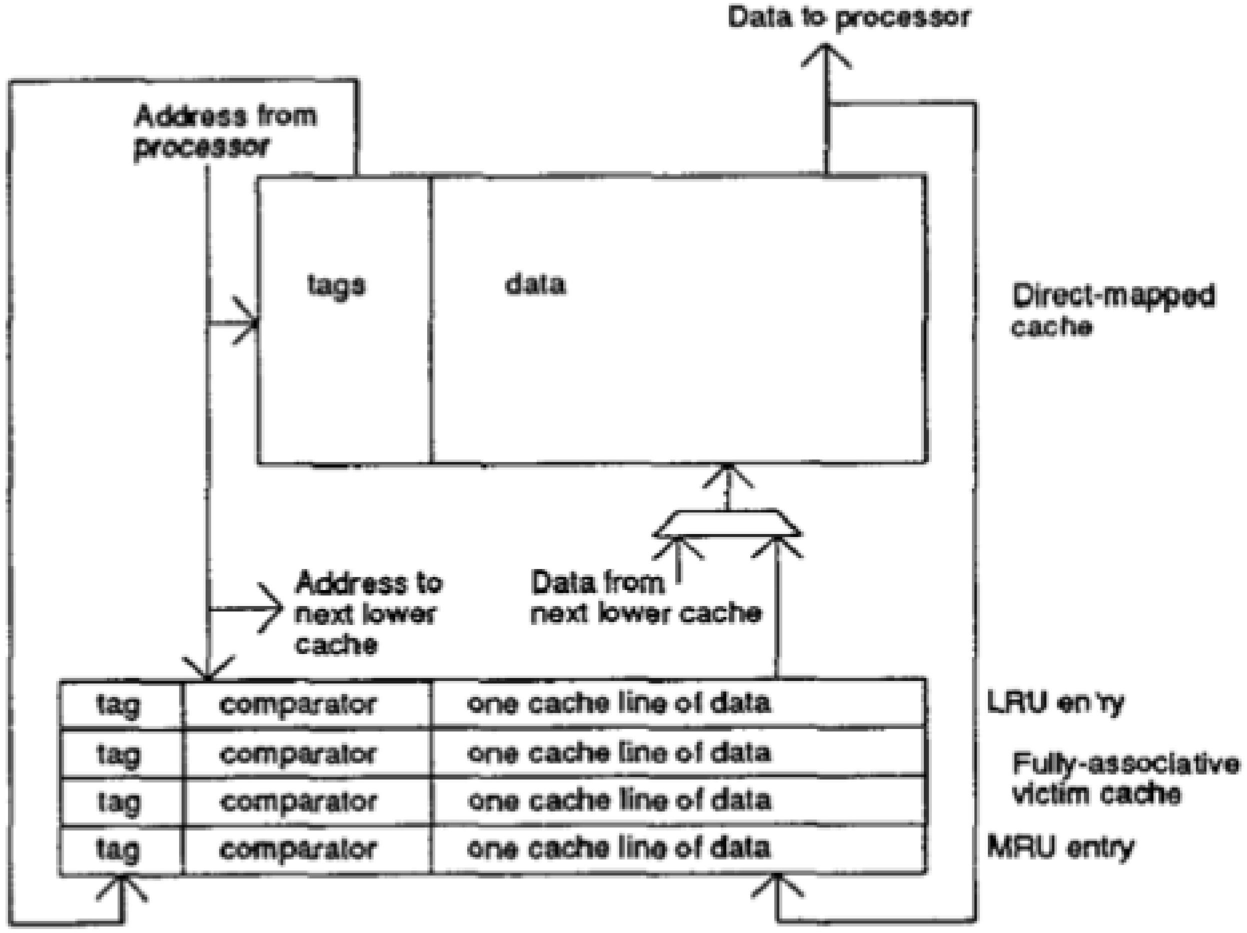

Figure 3-4: Victim cache organization

Victim Buffer Gains

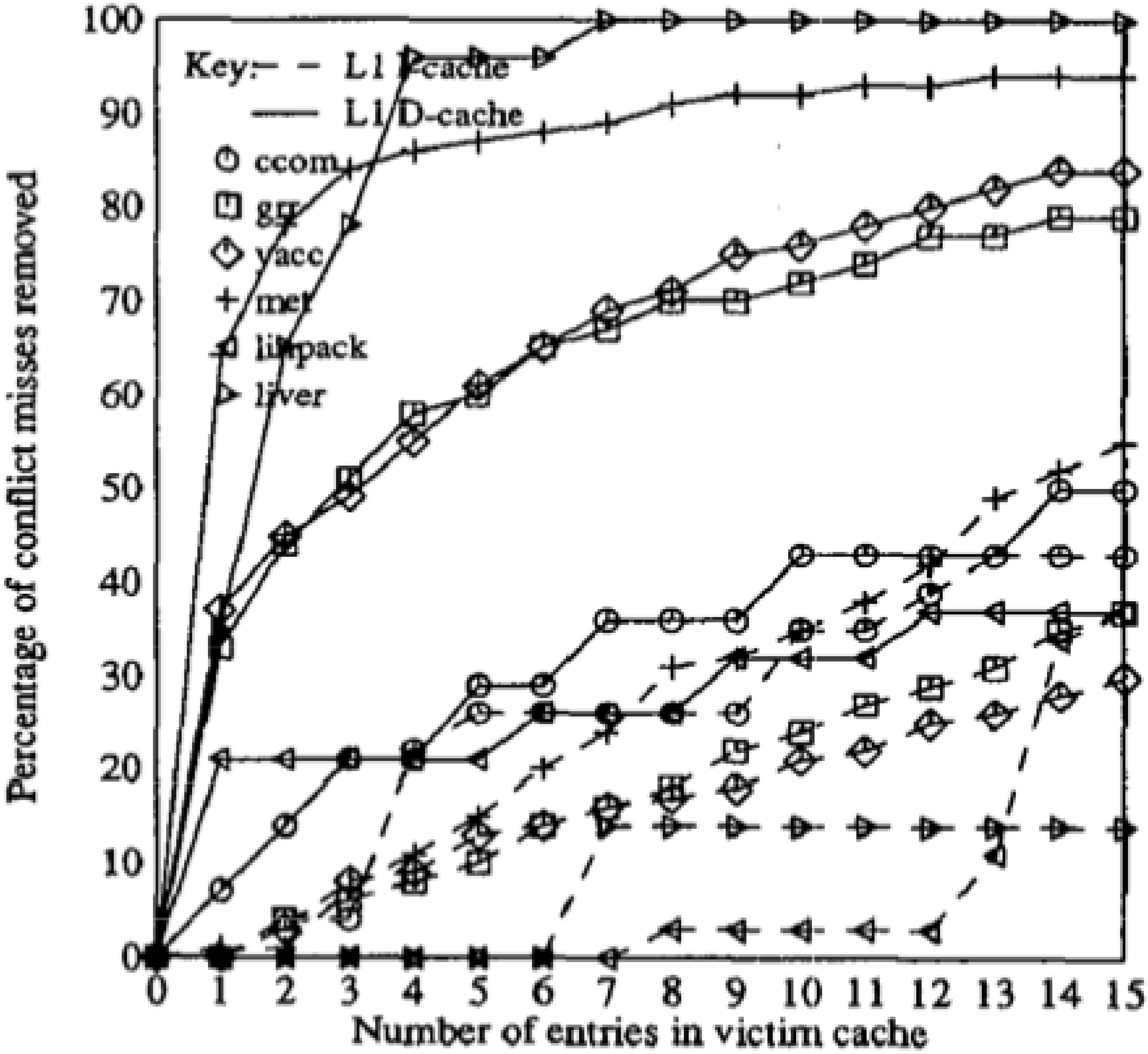

Figure 3-5: Conflict misses removed by victim caching

Interesting Metrics

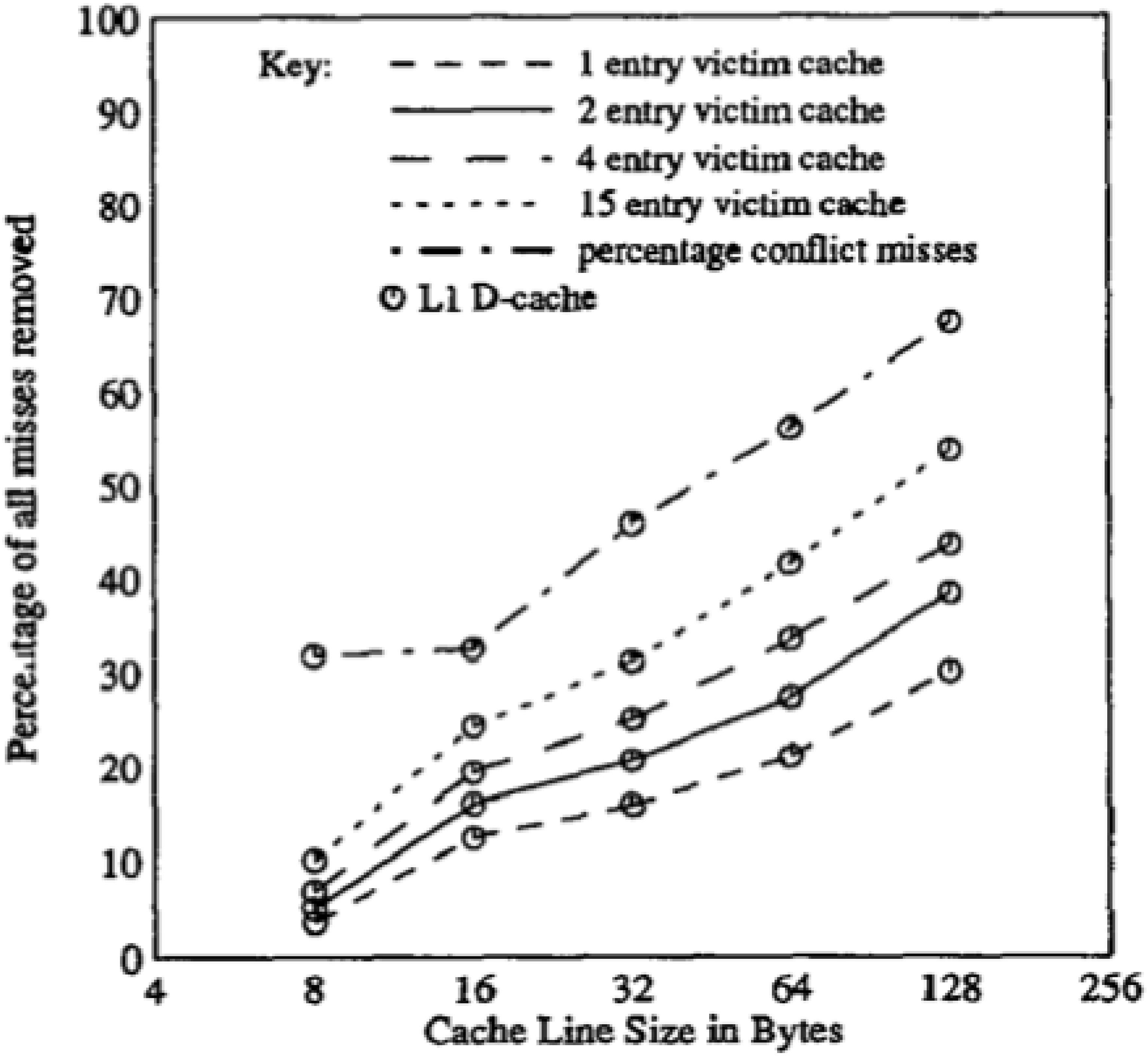

Figure 3-7: Victim cache: vary data cache line size

Fractional Associativity

- Norm mentions the notion of fractional associativity.
- You can think of a victim buffer as adding additional associativity to just the lines in the cache that need it.
  - Why pay for associativity everywhere, when it’s just a few problematic cache lines?

Victim Buffers Today

- Victim buffers are very popular today, but not as Norm envisioned them.
  - Associativity is not prohibitively expensive.
- In CMPs, cache inclusion makes less sense:
  - 256KB L2
  - 8 cores = 16KB L1 D + I
  - L1 capacity is equal to L2 capacity
  - Inclusion is very wasteful -- everything is duplicated
  - Instead, use the L2 as shared victim buffer
    - Associative, but not full associative.

Address Compulsory and Capacity Misses

- Fixing compulsory misses is tough: You must predict the future.
- Previous techniques
  - Larger cache lines
  - Next line prefetcher

Simple Prefetching

- Prefetch always
  - Always bring in the next line on every reference
  - Seems wasteful.
  - He says it’s not tractable, but that only applies to this system (maybe)
- Prefetch on miss
  - Seems more reasonable.
  - Similar to doubling the cache line size
  - Can reduce misses by half.
- Prefetch tagged
  - When a prefetched block is actually used, the next line is fetched.
  - Could reduce misses to zero, but waiting for the use is actually too late.
  - We need to get farther ahead in the access stream. That would require more space.

Stream buffers

- The previous techniques waste cache space.
    - perhaps displacing other useful data
- A stream buffer provides dedicated space for the prefetched data.

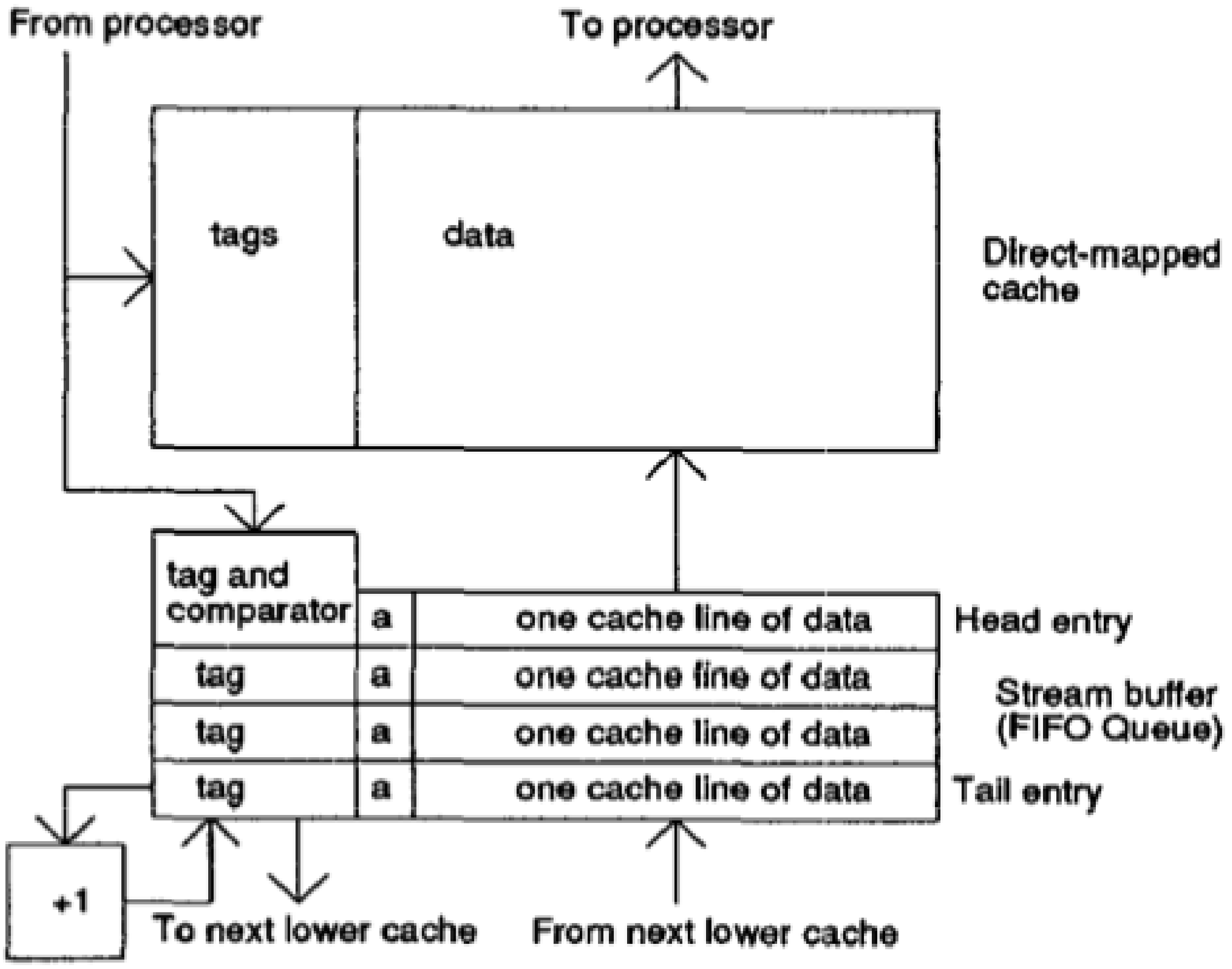

Figure 4-2: Sequential stream buffer design

Stream Buffers

- On a miss, start fetching successive lines
- When they return, but them in the stream buffer
- On future misses, check the head of the stream buffer, if it's a hit, great! Fetch another line.
- If it's a miss, clear the stream buffer and start over.

Effectiveness

- Great for instructions
- Ok for data.

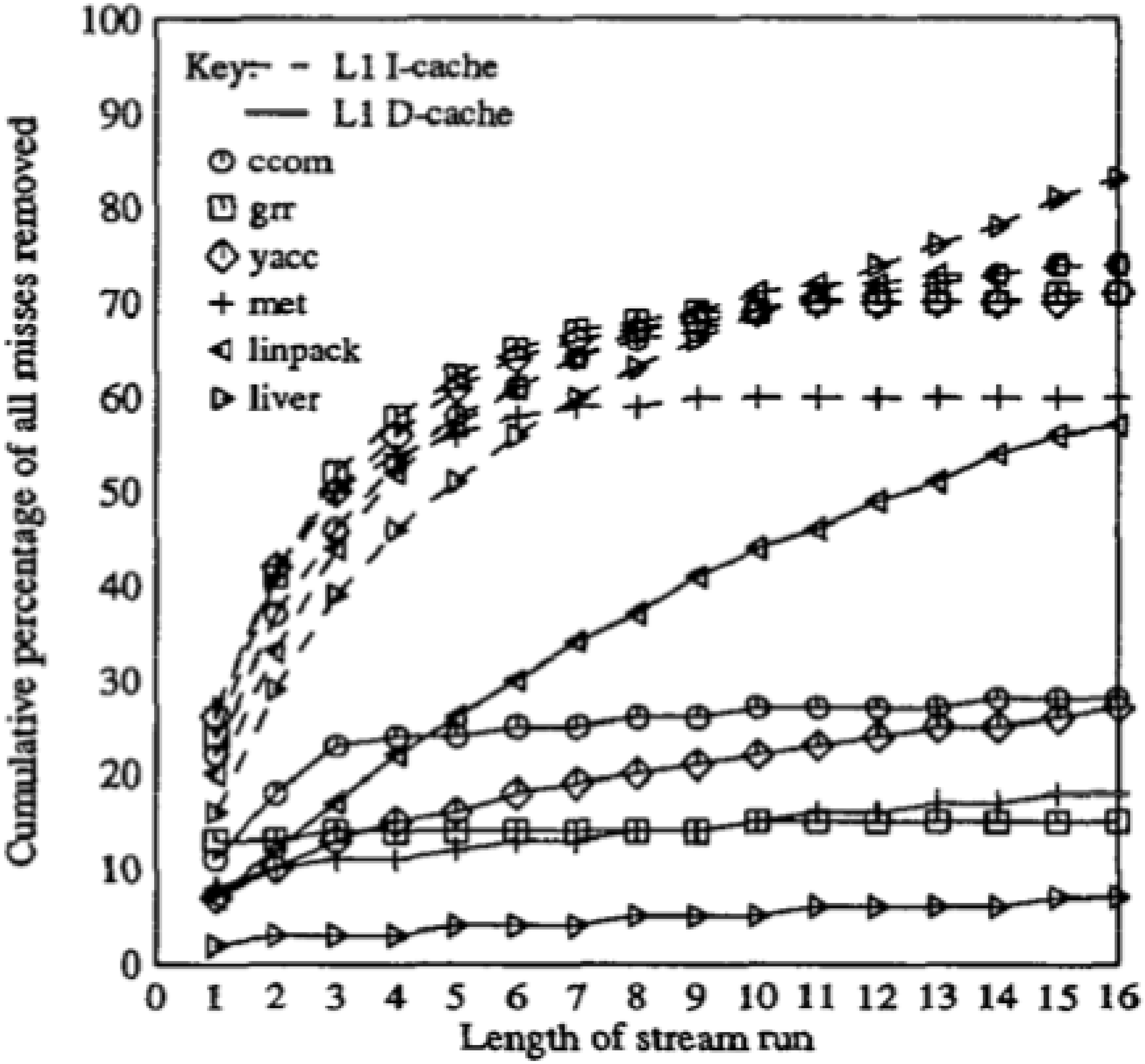

Figure 4-3: Sequential stream buffer performance

The problem with data

- Programs often make interleaved, sequential streams of accesses
- One stream buffer is not enough.
- There is only one instruction stream, however.

Build Multiple Buffers

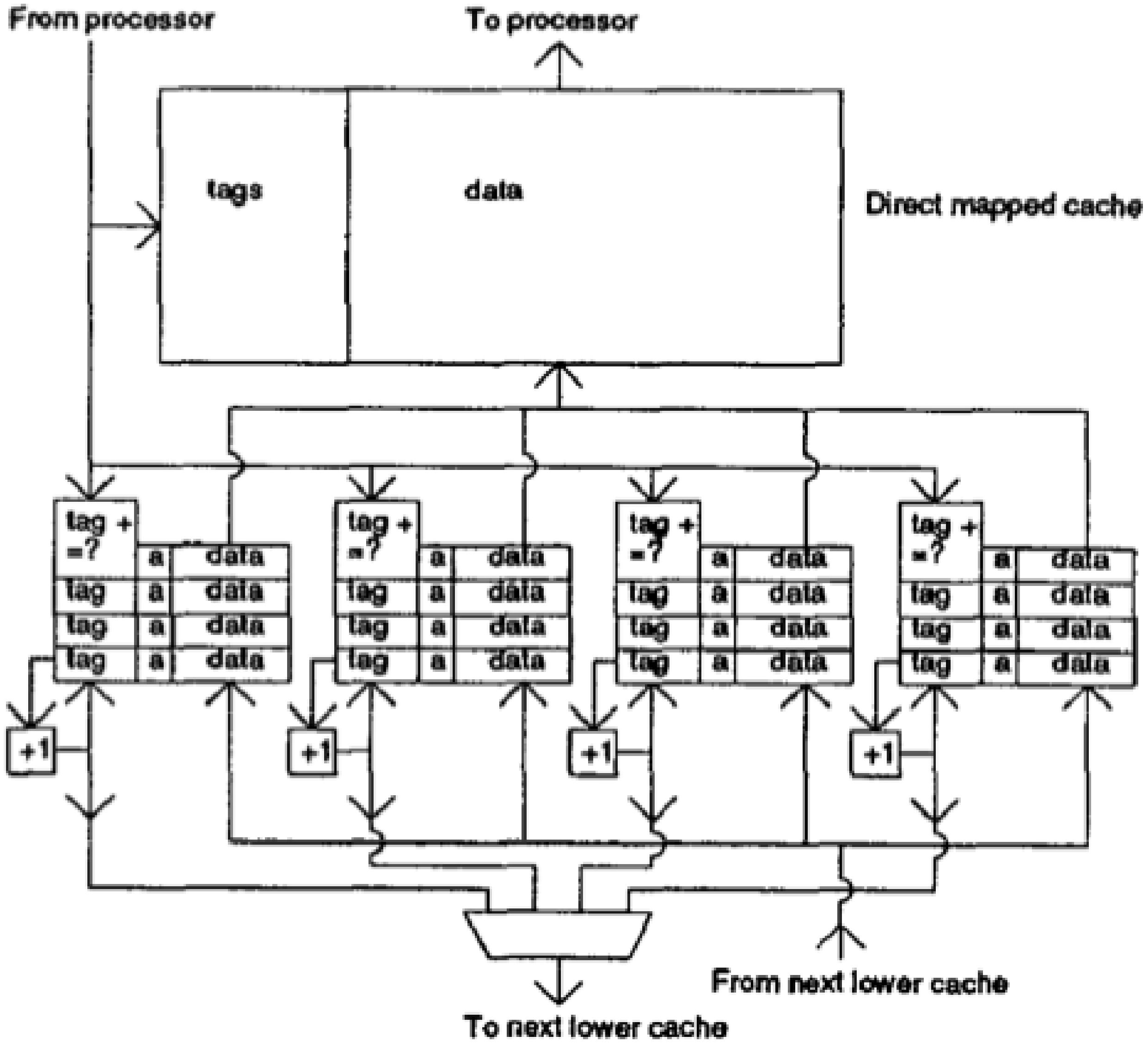

Figure 4-4: Four-way stream buffer design

Stream Buffers today

- Prefetching is very popular today
- Prefetchers are very sophisticated, and very hard to reverse engineer and/or out-smart.
  - You need to disable them if you want to measure much of anything about your memory hierarchy.
- You will design your own prefetcher later in the course.

Conclusions

- Victim buffers and stream buffers are worthwhile
- They can substantially reduce 3 of the 4 Cs
- The paper says very little about how they would perform on a particular machine or how they should be provisioned.
  - It is all about trends and the underlying characteristics of the access stream that they exploit.
  - The hardware trade-offs are also important.