Implementation and Improvisation of Cache Optimization Techniques

Ashwin S Sivan
Ayush Bhandari
Bharat Singh

Introduction

- A processor can process data as fast as it can be fed data
- Memory hierarchy (and Why we need it?)
- Cache - A temporary storage space or memory that allows fast access to data

Cache

- Improves the rate at which processor can process request
- Principle of locality – Temporal and Spatial
- Hit time, miss rate, miss penalty and access time
- Improves performance by – reducing hit time, miss rate and miss penalty

Important Terms

- Mapping – rule by which a block is placed in memory
- Direct Mapped Cache (and its advantages & disadvantages)
- Associative Cache – several places at which a block can be placed in Cache
- Set Associative Cache
- Fully Associative – block can be placed at any position in Cache
- Read
- Write

Progression of Our Proposed Cache Model

What we wanted from our cache?

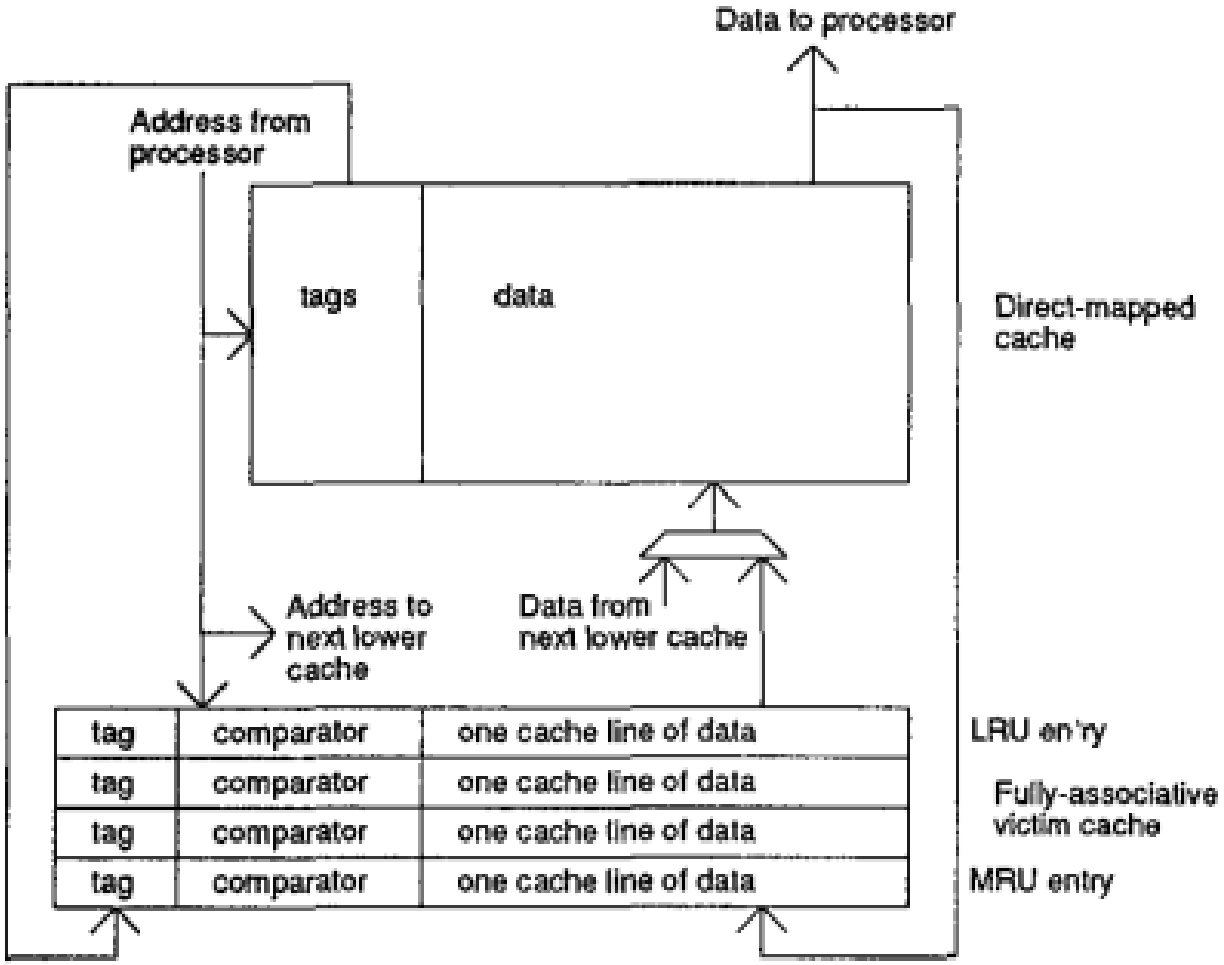
- It can predict and bring memory blocks (in advance) which are even far away from each other in the memory (based on previous experiences)
- It can provide best performance possible for an application
- Essentially we wanted to provide Intelligence to cache

Problems in this model

- AI element needs data in large numbers, to fairly predict the next access
- Such implementation would increase the size (increased hit time, miss time)
- It would take longer time to process a request (this defeats whole purpose of using a cache)
- Need a dynamic memory structure (as every new access must be used by AI element to predict the next access)
- Increased cost (due to addition of a new hardware component)
- Cost, latency and space are critical and important performance metric for Cache (hence this model was impractical)

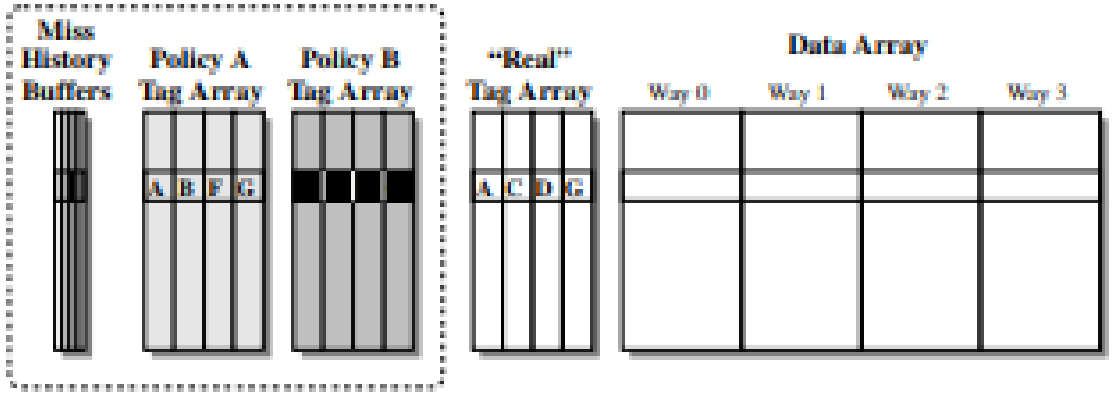
Victim Cache

- A small fully-associative cache present in b/w L1 cache and L2 cache
- It provides associativity to L1 cache
- It contains block that were recently evicted from L1 cache
- It prevents duplication of blocks in L1 cache and VC (hence provides space for an extra memory block to settle in)
- Advantages of using VC – decreased conflict miss, improved hit latency for Direct Mapped Cache (or caches with small associativity)
- Reference - *Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers,* Norman P. Jouppi, ISCA 1990.

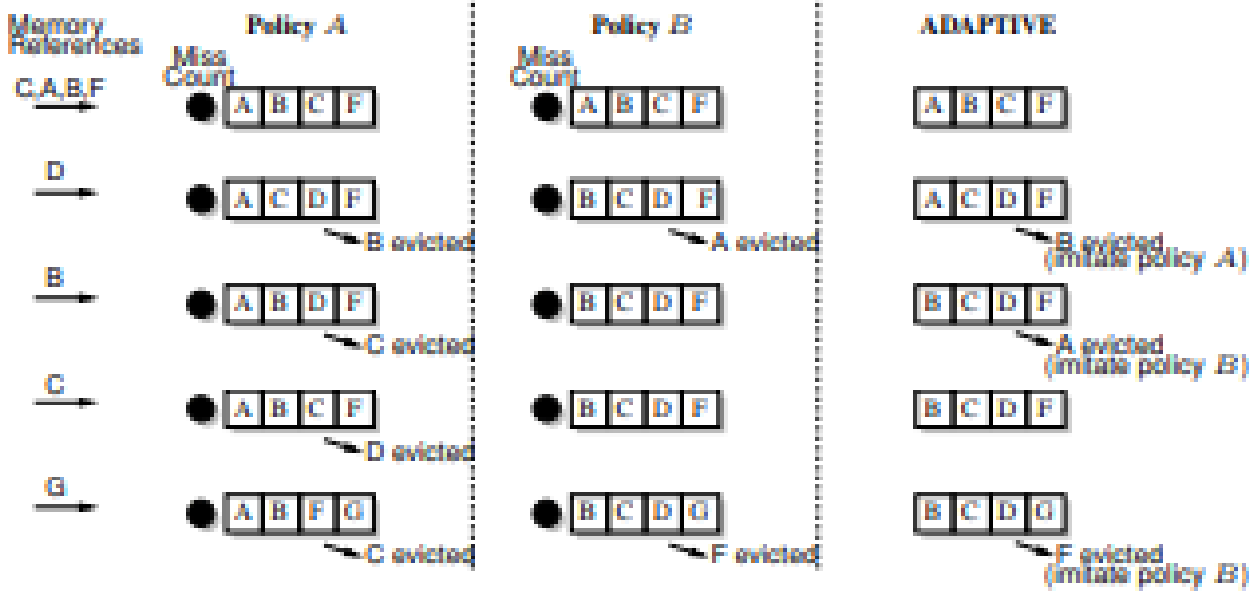

Data to processor
Address from processor
tags
data
Direct-mapped cache
Address to next lower cache
Data from next lower cache
tag
comparator
one cache line of data
LRU entry
Fully-associative victim cache
MRU entry

Adaptive Cache Replacement

- Scheme under which we can combine any two cache management scheme (LRU, FIFO, LFU, Random etc) and adaptively switch between them
- Simple and adaptivity actions are all performed off the critical path of memory reference handling
- According to this algorithm, policy producing lesser misses will be implemented
- Reference - *Adaptive Caches: Effective Shaping of Cache Behavior to Workloads, Georgia Institute of Technology (Intel).*

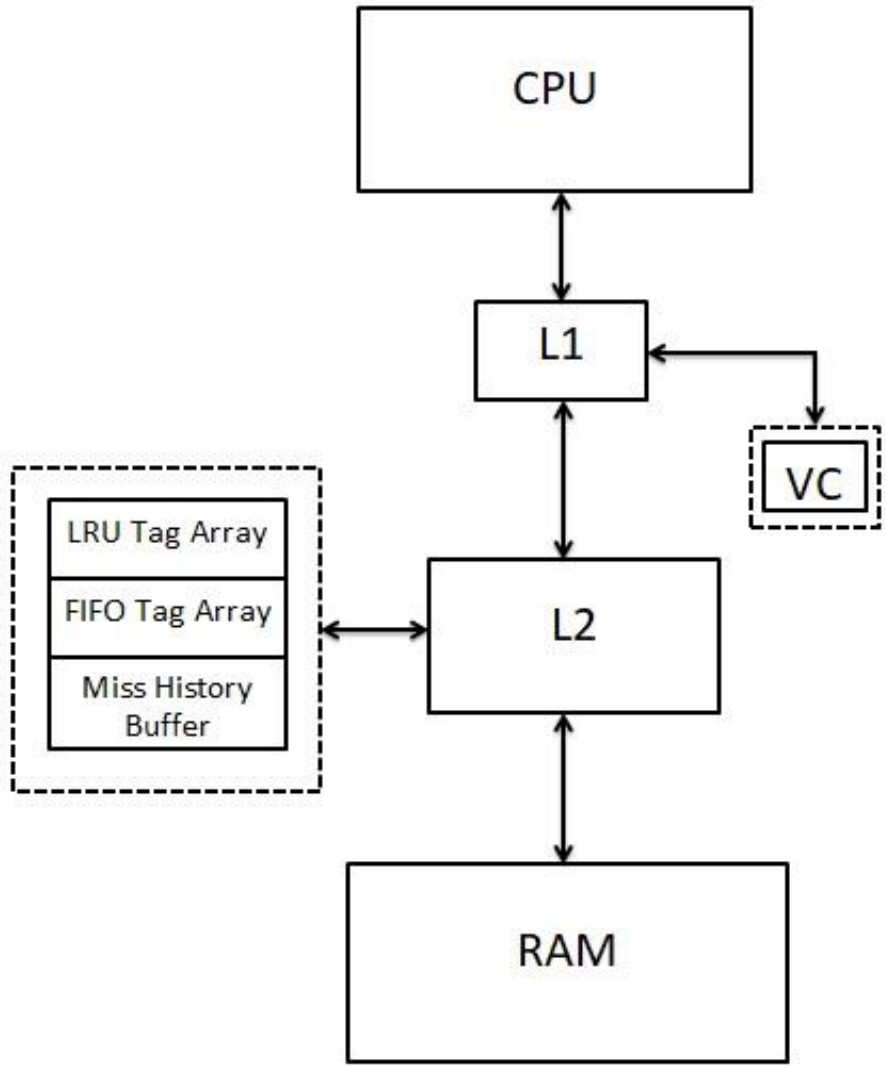


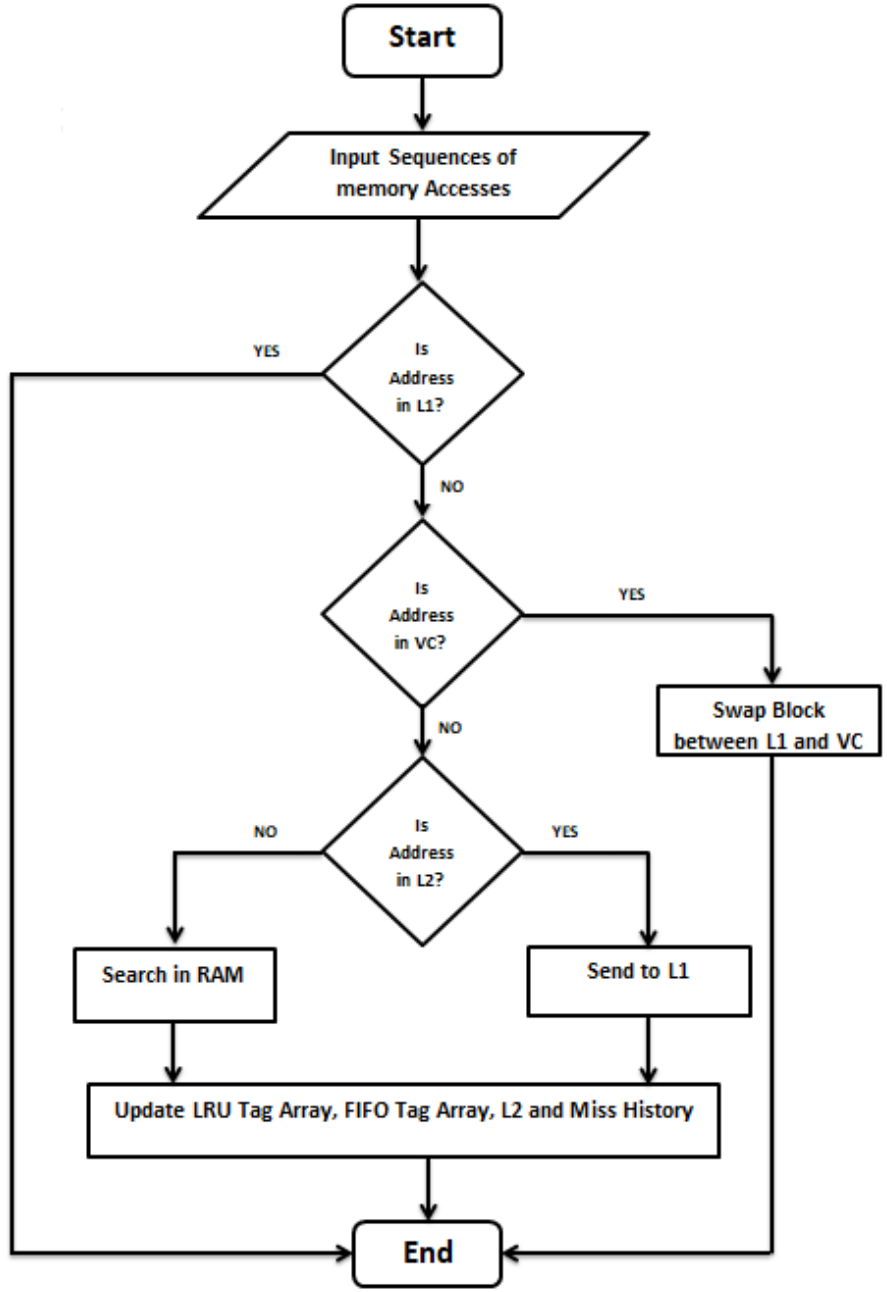
Additional structures for adaptive caching



Our New Proposed Cache Model

- We proposed a model which was combination of these two approaches
- Place VC in between L1 and L2 cache
- Use ACR on L2 cache
- Using a combination of these two methods – solved problems with our previous model


CPU
L1
VC
LRU Tag Array
FIFO Tag Array
Miss History Buffer
L2
RAM


Start
Input Sequences of memory Accesses
Is Address in L1?
YES
NO
Is Address in VC?
YES
Swap Block between L1 and VC
NO
Is Address in L2?
NO
YES
Search in RAM
Send to L1
Update LRU Tag Array, FIFO Tag Array, L2 and Miss History
End

```
In [51]: runfile('C:/Users/ashwinss3/Desktop/Project/Cache_without_vc.py', wdir='C:/
Users/ashwinss3/Desktop/Project')
Misses in L1 : 1413
Hits in L1 : 114
Misses in L2 : 105
Hits in L2 : 1308
Time taken: 103965

In [52]: runfile('C:/Users/ashwinss3/Desktop/Project/Cache_with_vc.py', wdir='C:/Users/
ashwinss3/Desktop/Project')
Misses in L1 : 1413
Hits in L1 : 114
Misses in L2 : 105
Hits in L2 : 1266
Misses in Victim Cache : 1371
Hits in Victim Cache : 42
Time taken: 102824.7

In [53]: runfile('C:/Users/ashwinss3/Desktop/Project/Adaptive_cache_with_vc.py',
wdir='C:/Users/ashwinss3/Desktop/Project')
Misses in L1 : 1413
Hits in L1 : 114
Misses in L2 : 175
Hits in L2 : 1190
Misses in Victim Cache : 1365
Hits in Victim Cache : 48
Misses in case of Lru Cache: 161
Misses in case of fifo cache: 343
Time Taken: 120020.5
```

Thank You