International Journal For Technological Research In Engineering
Volume 5, Issue 10, June-2018 *ISSN (Online): 2347 - 4718*

A STUDY ON CACHE REPLACEMENT POLICIES IN COHERENT CHIP MULTIPROCESSOR SYSTEMS (CMPs)

Deeksha Satish[1], Dr. Manimala S[2]
[2]Assistant Professor, [1,2]Department of Computer Science and Engineering,
JSS Science and Technological University, Mysuru, India

ABSTRACT: Computers usually store data in terms of hierarchy of memoriesranging from the most expensive fast memories to low-cost andslower memories. It is very much prevailing to store data in fastmemories to try to avert requests to the slowerones and the same is referred to as caching. But whenthe fastest memory becomes adequate enough and a new data must be interpolated, some other data has to be recovered. There are various methods to decide which data to remove and these methods are oftencalledcache replacement algorithms. A miss in the last level caches causes hundreds of stall cycles due to the need for a memory access. Therefore, last level caches are designed to reduce the possibility for cache misses. Since last level caches absence of temporal locality, the Least Recently Used policy leads to bad performance for them. Hence, many replacement policies were suggested to improve the miss rate for last level caches while maintaining low hardware aerial and minimum design changes. Various algorithms have been studied andtheir performance often depends on the workload.This paper provides an overview of some much studied cache algorithms – Least recently used (LRU), Most recently used (MRU) and Quad-age replacement policy in a coherent chip multiprocessor system.

Keywords: LRU, MRU, CMP, Quad-age, Request to response latency, Hit rate.

I. INTRODUCTION

Caches play a critical role in the reduction of delay of memory accesses by giving a temporary fast-access storage unit for the data that is being accessed by the CPU. On a cache miss, data is fetched from the memory and placed in its analogous locale in the cache. A productive cache replacement policy can naturally reduce the cache miss rate and thus reducing the risk of a penalty of hundreds of cycles expenditure due to memory accesses.The concept of cache is to store data from a slow memory in a faster memory. This isdone in order to minimize the number of requeststo the slower memory and hence resulting in the reduction of memory accesslatency. Cache is used in various applications such as, the harddisks, web servers, databases and CPUs to name afew.Computers contain different memories which form a hierarchy (as shown in Figure 1) with respect to speed, cost andcapacity. A classic CPU today contains at least 3cache blocks, which are called L1, L2 and L3. Theseare the fastest memories. The Random Access Memory is slower than the CPU cache but faster than anyHard Disk Drive. An overview of access times in computer hardware is shown in Table 1. As can be seenfrom this overview, it is more than a million timesfaster to retrieve data from the L1 cache than froma hard disk. It is therefore useful to cache frequentlyused data.

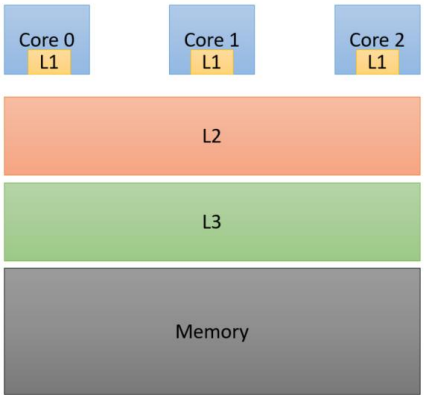

Figure 1: Representation of Multilevel cache hierarchy

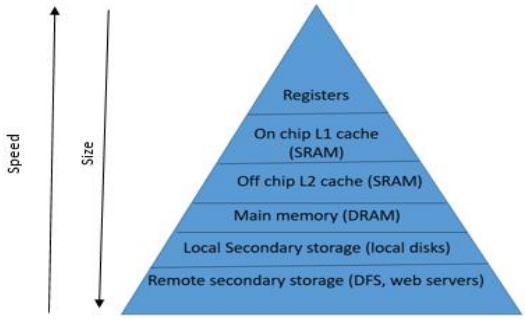

Figure 2: Representation of memory hierarchy

L1 cache reference	0.5ns
L2 cache reference	7ns
Main memory reference	100ns
Read 1MB from memory	250,000ns
Read 1 MB from SSD	1,000,000ns
Hard disk drive seek	10,000,000ns
Read 1 MB from a Hard Disk Drive	30,000,000ns

Table 1: Read access latency of computer hardware

II. LITERATURE SURVEY

Belady in [1] proposed the optimal replacement policy (OPT) which provided an upper limit on the hit rate that can be ever achieved. OPT states that "the optimal candidate for replacement is the one that is accessed farthest in the future". This policy cannot be implemented since it requires knowing which lines will be accessed in the future and when they are going to be accessed. The traditional replacement policies well known in the literature includes: LRU, LFU, random replacement and FIFO, with the LRU policy being the

standard for today's on-chip caches. Improvements on replacement policies have been static for a long time for two main reasons: the suggested solutions required significant additional hardware, and the assumption that LRU works sufficiently, which is the case for L1 caches only. [9] The fact that the LRU policy has bad performance when used with last level caches, led to the emergence of a lot of studies that attempt other optimized solutions that may achieve a performance as close as possible to OPT while maintaining low hardware overhead. Qureshi et al. [5] [6] proposed anti-thrashing policies that performs well for memory intensive workloads and proposed a dynamic policy that adaptively chooses either LRU or the proposed anti-thrashing policy depending on the workload.

Other policies that combine both LRU and LFU were proposed by Subramanian et al. in [9] and Megiddo and Modha in [3]. In [7] Qureshi et al. proposed a policy that takes its replacement decision in order to exploit Memory Level Parallelism (MLP), such that lines that may cause misses with high MLP cost are least probably evicted. Rajan and Govindarajan [8] proposed a policy that mimics Belady's optimal policy [1] by producing the shepherd cache. Zebchuk et al. [11] modified the shepherd cache so that it has lower hardware overhead.

Recent studies also include using replacement policies to manage shared caches in CMPs. Most of the proposed policies are optimizations to the proposed policies for uniprocessor. In Qureshi et al. modified their dynamic policy so that it chooses the replacement policy for each core individually. Kron et al. [2] proposed the notion of promotion policies and combines it with the dynamic policy proposed in [6] to form a policy that combines both dynamic replacement and dynamic promotion. In [10] Xie and Loh proposed a policy that implicitly partition the shared cache among cores using the insertion and promotion policies.

III. CACHE REPLACEMENT POLICIES:

The below section gives the brief idea about LRU and MRU policies usage and their consequences of using it in chip multiprocessor systems. Quad age replacement policy is one such less used algorithm in CMPs. The performance of CMPs with different replacement policies is also depicted with the graph.

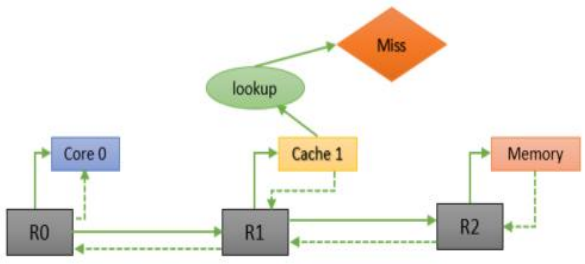

Figure 3: Representation of cache miss in the CMP system

3.1 MLP-Aware replacement policies

Memory Level Parallelism (MLP) refers to the process of serving multiple memory operations due to cache misses in parallel. MLP reduces the number of stall cycles due to memory accesses. Multiple cache misses may occur concurrently because modern processors are speculative superscalar processors with pre-fetching capabilities. [7]

In [7] Qureshi et al. proposed the idea of optimizing replacement policies so that they exploit memory level parallelism, these are called MLP-aware replacement policies.

Instead of reducing the number of misses, MLP-aware policies aim to reduce the number of memory stalls by avoiding misses that occur in isolation. Such misses do not exploit MLP since they will be served individually. When a replacement decision is to be made, lines that may cause isolated misses will be replaced with the least probability.

3.2 Pros and Cons of Least Recently Used(LRU) and Most Recently Used (MRU) Policies:

The Least Recently Used (LRU) Policy is one of the standard replacement policies used for multilevel caches (L1, L2 and L3 caches). The LRU policy works sufficiently with L1 caches since they benefit the most from temporal locality because of their direct interface with the processor. In addition the LRU policy is perfect for the simplicity required in the 3 design of L1 caches.

However, references to L2 caches (and last level caches in general) lack temporal locality, since lines that are referenced recently are kept in the L1 cache and they will not be requested from the L2 cache again. LRU will cause these lines to be kept in the cache until they travel all their way from the MRU (Most Recently Used) position to the LRU used position, and only then they will be evicted. In the worst case these lines may never be reused during their residence in the cache, these lines are called zero-reused lines. Moreover, for workloads that have working sets that are larger than the L2 cache, LRU causes thrashing.

These workloads are called memory intensive workloads. When the LRU policy is used for such workloads, lines that are inserted in the cache will be referenced in the future but due to the capacity misses, they will be replaced by new lines before being re-referenced.

Thus, LRU causes a 0 hit rate for these workloads since all of the cache lines will be zero-reused lines. In addition to what stated so far, the deign goal in L2 and last level caches is to minimize the possibility of long accesses to the memory that would be caused by a cache miss.

Besides, last level caches are of larger sizes and higher associativity. All these factors made recent studies attempt to find optimized replacement policies that can work better than the LRU poor performance with last level caches.

The fact that replacement policies can be optimized to gain more control over the cache, motivated the researchers to extend their work to consider the problem of shared cache management in chip multiprocessors (CMPs).

For all the proposed replacement policies, there are multiple design issues that must be taken in consideration. First, the additional required hardware should be as minimal as possible. Second, the policy should work efficiently with different types of workloads. It must not perform worse than LRU for a wide range of workloads. Finally, it should

produce minimum changes to the existing cache design.

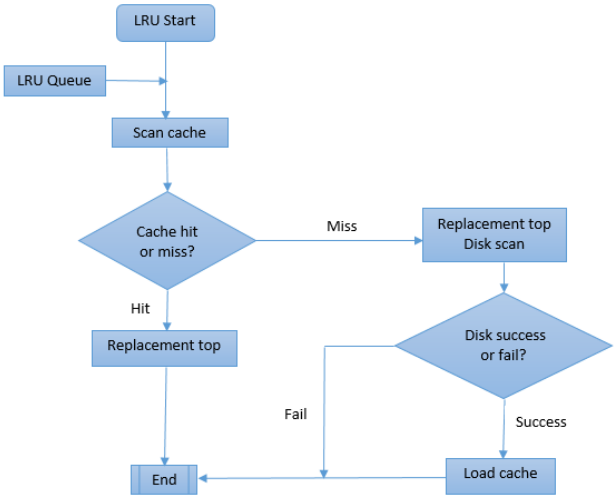

Figure 4: Flow diagram of LRU

3.3 Quad age Replacement Policy:
Quad age replacement works based on the ages assigned to a cache-line. Any cache-line can have age between 0 and 3, 0 being the least recently used and 3 being the most recently used. A cache-line is installed at an age of 1 to avoid thrashing and a hit promotes the cache-line age to 3. This is an independent piece of logic implemented in the cache. The performance of quad-age policy can be known by running the system with increasing the number of rows of core to 1 in every run. All these are depicted in the below study.

IV. EXPERIMENTAL RESULTS

This section gives the brief idea and performance of CMP system with three different cache replacement policies namely LRU, MRU and quad -age replacement.

Case 1:
Size of cache = 1 MB
Associativity of cache = 16 way
Number of sets = 4
Address stride = 64
Cache-line width = 256

Case 2:
Size of cache = 1 MB
Associativity of cache = 16 way
Number of sets = 4
Address stride = 128
Cache-line width = 256

Case 3:
Size of cache = 1 MB
Associativity of cache = 16 way
Number of sets = 4
Address stride = 64
Cache-line width = 512

Case 4:
Size of cache = 1 MB
Associativity of cache = 16 way
Number of sets = 4
Address stride = 128
Cache-line width = 512

All the above mentioned cases are run one at a time for 10000 Read requests.
Run 1: 10,000 Requests with two cores and a memory in one row
Run 2: 10,000 Requests with two cores and a memory in one row and two cores with a memory in second row
Run 3: 10,000 Requests with two cores and a memory in one row, two cores with a memory in second row and one core with memory in the third row.

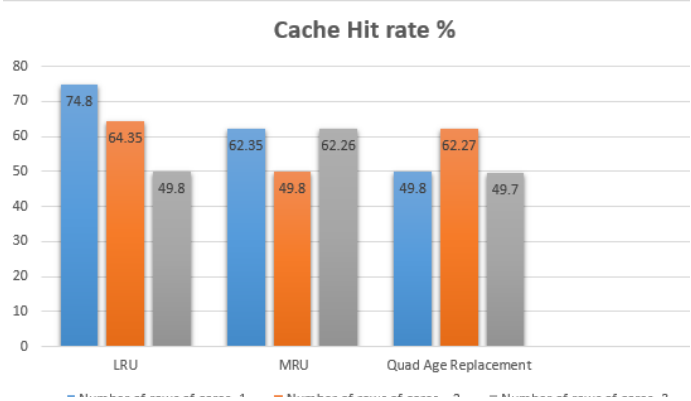

Figure 5: Performance graph with different replacement policies.

V. CONCLUSION

It's now concluded that LRU produces comparatively good result for first case. MRU for third and Quad age for the second. Overall, it's now clear that the cache replacement policies are to be chosen as per the requirements of number of cores.

REFERENCES

[1] Belady L. (1966). A Study of Replacement Algorithms for a Virtual Storage Computer. IBM Systems Journal, vol.5, pp. 78-101.

[2] Kron J., Prumo B. & Loh G. (2008). Double-DIP: Augmenting DIP with Adaptive Promotion Policies to Manage Shared L2 Caches. In the 2nd Workshop on CMP Memory Systems and Interconnects (CMP-MSI), Georgia Institute of Technology.

[3] Megiddo, N. & Modha, D.S. (2004). Outperforming LRU with an Adaptive Replacement Cache Algorithm. Proceedings of the USENIX Annual Technical Conference 2005 on USENIX Annual Technical Conference.

[4] Qureshi M., Jaleel A., Hasenplaugh W., Sebot J., Jr. S. &Emer J. (2008). Adaptive Insertion Policies for Managing Shared Caches. Proceedings of the 17th international conference on Parallel architectures and compilation techniques (PACt'08), pp. 208-219.

[5] Qureshi M., Jaleel A., Patt Y., Jr. S. &Emer J. (2008). Set-Dueling-Controlled Adaptive Insertion

for High-Performance Caching. IEEE Micro, pp. 91-98.

[6] Qureshi M., Jaleel A., Patt Y., Jr. S. &Emer J. (2007). Adaptive Insertion Policies for High Performance Caching. Proceedings of the 34th annual international symposium on Computer architecture (ISCA'07), pp. 381-391.

[7] Qureshi M., Lynch D., Mutlu O. &Patt Y. (2006). A Case for MLP-Aware Cache Replacement. Proceedings of the 33th annual international symposium on Computer architecture (ISCA'06). pp. 167-178.

[8] Rajan K. &Ramaswamy G. (2007). Emulating Optimal Replacement with a Shepherd Cache. In Proceedings of the 40th International Symposium on Microarchitecture (Micro'07), pp. 445-454.

[9] Subramanian R., Smaragdakis Y. & Loh G. (2006). Adaptive Caches: Effective Shaping of Cache Behavior to Workloads. Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (Micro'06), pp. 385-396. [10] Xie Y. & Loh G. (2009). PIPP: Promotion/Insertion Pseudo-Partitioning of Multi-Core Shared Caches. Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA'09), pp. 174-183.

[10] EJO Neil, PEO Neil, G Weikum. The LRU-K page replacement algorithm for database disk buffering. [C] ACM SIGMOD Int.lConf on Management of Data, Washington, DC, 1993

