Volume 3, Issue 4, April 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Framework to Compare Inclusion Property Based Caches

Megha Soni*
M.Tech., Department of Computer Science and Applications,
Kurukshetra University, Kurukshetra, Haryana, India.
Rajendra Nath
Professor, Department of Computer Science and Applications,
Kurukshetra University, Kurukshetra , Haryana, India.

Abstract— Demand for increasing functionality and performance in system designs leads to the development of multi-level cache hierarchy. Proper organisation of multi-level cache hierarchy is of prime importance. For this, many concepts of cache hierarchy such as inclusion property, needs to be re-thinked. According to the decision for whether to enforce inclusion or not, multi-level caches can be grouped into three categories-inclusive, exclusive and non-inclusive. This paper concentrates on these three cache types and a comparative study has been carried to find which cache is more effective by considering various parameters like data duplication, hit-rate, capacity, speed, performance, presence/absence of inclusion victims, bandwidth requirements and power consumption, coherency management etc.

Keywords— multi-level cache, inclusion, non-inclusion, exclusion, cache coherence.

I. Introduction

Since the time cache memory was introduced, a lot of improvements have been made. Hardware engineers are continuously trying to limit the growing performance gap between fast processor and slow main memory. Reduction in access time is the most desirable requirement in memory systems. This needs increment in the number of cache hits. Hit rate is a function of the size of the cache [1]. Increasing hit rate implies building large sized caches which in turn reduces the average access time. Thus cache size cannot be increased without limitation. There exists a fundamental tradeoff between size and speed of memory system [2].

To address the above mentioned tradeoff, caches are organized into hierarchies. Multi-level cache hierarchy as shown in Fig1, is the most effective way in dealing with large caches.

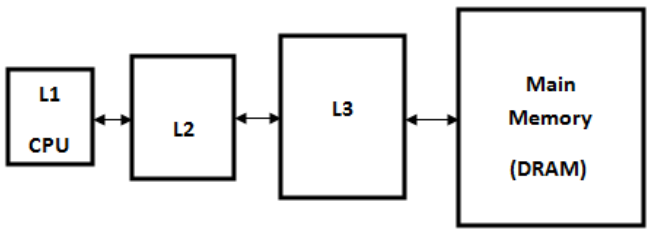

Fig1. Multi-level cache hierarchy

Primary cache attached to CPU is Level1 (L1) cache often present on-chip. For high speed benefit and reducing cache size, Level2 (L2) is provided, which may be on-chip or nearby on module. It provides a good backup for L1 cache misses. Further, more levels can be added to cache hierarchy to avoid the worst case main memory access. Each level must have a significantly larger capacity than the level above it in the hierarchy. Cache memory hierarchy takes the advantage of the principle of locality of reference [3]. The future memory accesses are near past accesses. Two types of locality, temporal and spatial are used. This can be exploited by caching data likely to be used again in upper cache levels i.e. place most needed data in L1. As we move deeper in the hierarchy, locality of reference seen by each level decreases. Processor architects should design an efficient and high performing cache hierarchy. One of the key design choices is whether or not to enforce inclusion. Inclusion and mutual exclusion are two important mathematical principles. These principles can also be applied to computer memory systems. The principle of inclusion and mutual exclusion defines a particular class of relationship that exists between any two fundamental units. An inclusive relationship between two units is one in which every item found in one of the units has a copy found in another leads to duplication at all the units: an inclusive relationship between A and B specifies that any given item is present in both A and B or in neither. An exclusive relationship between two units is one in which the expected intersection between those units is the null set: an exclusive relationship between A and B specifies that any given item is either in A or B or in neither. Based on the inclusion property of multi-level cache hierarchy, caches are grouped into the following categories-inclusive, exclusive and hybrid of these. The multi-level cache organisation, which satisfies the inclusion property, is named as Inclusive Cache (IC) organisation. The multi-level cache organisation, which satisfies mutual exclusion property, is named as

Exclusive Cache (EC) organisation. Non-Inclusive Cache (NIC) organisation is the hybrid of IC and EC. IC causes wastage of storage area, but simplifies coherence. Some disadvantages of IC can be overcome by NIC and remaining by EC. The rest of the paper is organized as follows: Section II. describes related work. Section III. introduces IC, EC and NIC organisation schemes. Section IV. gives a comparison among the three schemes using various parameters like hit rate, capacity, performance, speed, execution time etc. Section V. concludes the paper and Section VI. presents future scope of the work.

II. Related Work

A multi-level cache hierarchy had remained an important concept since cache memories were introduced. Simplification of cache coherence protocol using multi-level inclusive cache hierarchies was firstly studied by Baer et al. in [4]. Coherence problem was simplified, but low capacity and inclusion victims still remained a problem in inclusive caches. Various techniques were used to reduce the number of inclusion victims. For single threaded workloads, to improve the performance of inclusion, a Global Replacement was proposed by Zahran in ([5],[6]). The number of inclusion victims was reduced but desirable performance benefits were negligible. Zahran's work was then followed by Garde et al. [7], he deconstructed global replacement for single and multi-core processors. They analysed the miss stream of a non-inclusive last level cache for global replacement in inclusive caches. Inclusion victims in direct mapped network caches were observed by Fletcher et al. [8]. Three solutions were proposed to address the problem i.e increasing the cache associativity, a victim cache [9], or making the LLC non-inclusive and using a snoop filter to ease cache coherence. Thus additional hardware requirement was the main problem. To further reduce the number of inclusion victims and minimise the hardware requirement, Jaleel et al. proposed Temporal Locality Aware(TLA) cache management policies to allow an inclusive LLC to be aware of the temporal locality of blocks present in upper level caches[10]. These three TLA policies were Temporal Locality Hints (TLH), Early Core Invalidation (ECI) and Query Based Selection (QBS). In these three schemes, the identification of hot lines were either conveyed to, derived by or inferred by the LLC. No external victim cache was used; instead an in-LLC victim cache was used in ECI. To relax inclusion property, non-inclusive cache was used. Non-inclusive cache hierarchies have been proposed in the context of aggressive prefetching ([6],[11]). Various non-inclusion cache design methods and benefits has been studied by Zahran et al. in [5].He has found non-inclusive cache to be effective in reducing level2 conflict misses. To further relax the inclusion property, most extensive work has been done within the context of prefetching. Zheng et al. has showed an algorithm for exclusive cache hierarchies [12], to avoid inclusion property at the expense of hardware complexity. Theodore et.al. examined the usage of exclusive caching in networking storage devices[13]. Dynamic exclusion scheme in multi-level cache design has been described by McFarling[14]. The level in which the instruction cache block will be placed was denoted by tags.

III. Inclusive, Exclusive and Non-Inclusive Cache Organisation

The multi-level cache organisation, which satisfies the inclusion property, is named as Inclusive Cache(IC) organisation. Data or content present in upper levels of multilevel cache hierarchy also resides in lower levels [4]. This implies that $L1 \subset L2$, and $L2 \subset L3$. In inclusive last level cache (I-LLC), the last level of the hierarchy contains contents of all the levels above it. Subset property is fully attained. When data miss occurs in L1 but hits in L2, data is copied from L2 to L1 by cache controller. If data miss occurs in both L1 and L2, data is retrieved from main memory to L2, which then provides data to both L1 and L2 caches, thus leads to duplication. Due to its simplicity, inclusive caching has been the standard cache behaviour. Due to redundancy, there is wastage of precious silicon estate. This wastage of space is more when IC is not much bigger than the previous levels of the hierarchy. Thus, it decreases the effective cache capacity available for unique information. The capacity of an IC hierarchy is equal to the size of the last level. There occurs a problem if the block is rarely referenced or not referenced again in future, thus its presence at all the levels causes' loss of useful information, reducing the overall system performance. To mitigate the problem, inclusion property should be relaxed. One possible solution can be prefetching [15], which implies violation of inclusion property in many cases. Another solution gives rise to use of Exclusive and Non-Inclusive Caches.

The multi-level cache organisation, which satisfies mutual exclusion property, is named as Exclusive Cache (EC) organisation. A rule is imposed that an element of information can only be held by a single cache. Data is not replicated from one level to another in cache hierarchy. This implies that $L1 \not\subset L2$ and $L2 \not\subset L3$ and so-on. Because there is no duplication of data, the effective cache capacity is equal to the sum of sizes of all the levels in the cache hierarchy. This reduces the dependence on main memory and hit rate is also increased. This scheme first fills the block at upper levels in hierarchy and upon eviction, fills them at lower levels. Suppose there are two levels in cache hierarchy. Data is present either in L1 or L2 cache. When there is a miss in L1 and hit in L2 upon access, then cache lines between the two are exchanged. For this, a victim buffer is needed [12], as shown in Fig2. Thus, data is exchanged not copied.

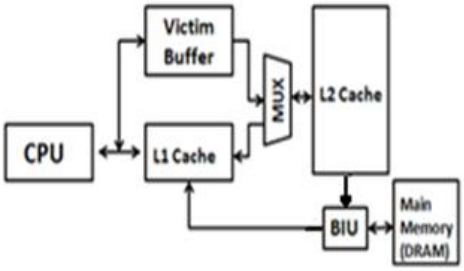

Fig2. Exclusive Cache Architecture

The multi-level cache organisation, which lies between IC and EC is named as Non-Inclusive Cache (NIC) organisation. It attempts to achieve best of both IC and EC. It neither ensures mutual exclusion nor maintains inclusion property. In this, the contents of upper level caches need not necessarily be a subset of lower level cache. This implies that in a two level cache hierarchy may be L1⊂L2 or L1⊄L2. A hit/miss in L1 cache need not necessarily be a hit/miss in L2 cache. Effective cache capacity is increased, which lies between that of IC and EC (i.e. between size of LLC and sum of size of all the levels in the hierarchy). A new safe cache analysis method for multi-level Non-Inclusive set associative caches can be found in [16].

If L1/L2 block size differs, then IC needs to evict multiple blocks from L1 in response to a single block eviction from L2, whereas in NIC, L1 may retain some portions of L2 cache block. Three different NIC designs are presented in [5]. In EC, exchange of data between levels is possible only if block sizes are identical; this makes the design space less flexible.

IV. Comparative Study of IC, NIC and EC

A sufficient background to compare IC, NIC and EC has been created in the previous section. The following parameters are identified to compare them: Data duplication, hit-rate, capacity, speed, bandwidth requirement, power consumption, victim cache/ victim buffer requirement, execution time, possibility of filtered access history, silent clean evictions, selective cache bypassing, coherence management, snoop filter requirement, state, back-invalidation, inclusion victim's presence, performance. Table1. gives the comparative study of three cache types.

A. Discussion

The various parameters shown in Table1 are discussed in detail below:

1) *Data duplication:* Data duplication depends on attainment of subset property among cache levels. As subset property is fully attained in IC, huge amount of duplicated data is there. NIC has partial or no attainment of subset property, thus duplicated data may or may not be present. In EC, subset property is absent; it contains unique data among all the cache levels.
2) *Hit-rate:* Hit rate is inversely proportional to data duplication. Therefore, hit-rate is the highest in EC and lowest in IC. In NIC, hit rate depends on the amount of duplicated data present.
3) *Capacity:* Data capacity means different for different cache types. Data duplication results in wastage of die-area. Thus, EC has highest capacity (sum of block sizes at all the levels) [17], then comes NIC (between size of LLC and sum of block sizes at all the levels). IC has lowest capacity value(size of LLC). Also when the LLC size is larger than other level caches, E-LLC and NI-LLC performs similar to I-LLC. When LLC shrinks, both E-LLC and NI-LLC performs better than I-LLC.
4) *Speed:* Next important parameter is speed. Presence of unique information in EC drives the need to check all the levels in the hierarchy as opposed to IC, where only last level needs to be checked. Thus IC is faster than EC. Speed of NIC lies between than of IC and EC.
5) *Bandwidth requirement and power consumption:* While handling misses in IC, there is less traffic between levels as the data is just copied from one level to another or from main memory to different cache levels. EC involves more data movement which causes large wastage of bandwidth and increased power consumption over IC and NIC
6) *Victim cache/victim buffer:* Performance of EC can be increased by using Victim Buffer which differs from Victim Cache used by IC, in that it holds data for the shortest time interval possible whereas Victim Cache holds data in the hope that it can be needed soon. Thus, it is the main resource differential between IC and EC. NIC may or may not use victim cache or victim buffer.
7) *Execution time:* Execution time of EC is highest due to worst case victim buffer penalties. Worst case occurs in a two level cache hierarchy when victim buffer is full and there is a L1 miss and L2 hit. Execution time of IC is

Table I. Comparison of IC, NIC and EC

Specifications	**IC**	**NIC**	**EC**
Data Duplication	Yes	May or May Not	No
Hit- Rate	Low	Depends	High
Capacity	Low	Depends	High
Speed	Fast	Medium	Slow
Bandwidth Requirement	Low	Medium	High
Power Consumption	Low	Medium	High

Victim Cache or victim Buffer	Victim Cache	Depends	Victim Buffer
Execution Time	Low	Medium	High
Possibility of filtered access history in LLC	Yes	May or May Not	No
Propagation of Silent Clean Evictions to LLC	No	No	Yes
Possibility of Selective Bypassing	No	Depends	Yes
Coherence Management	Simple	Difficult	Difficult
Snoop Filter Requirement	No	Yes	Yes
State	Simple	Depends	Complex
Back-invalidation	Yes	May or May Not	No
Inclusion Victim's presence	Yes	No	No
Performance	Low	Medium	High
Example	Modern day Intel microprocessors (like the Intel Core i7processors	Some Processors of the Intel Pentium family	Processors of AMD Athlon and Operton.

low because victim cache used by it performs faster than victim buffer. Execution time of NIC depends on its use of victim cache or victim buffer.

8) *Filtered access history:* Maintaining filtered access history is not possible in EC hierarchy, as the block is evicted from the last level upon a hit. A block can gather filtered access history in IC, which improves with the provision of hints. Achieving filtered access history is difficult or not possible in NIC.
9) *Silent clean evictions:* In EC hierarchy, silent clean evictions need to be written in the last level which is not required in NIC and IC hierarchy according to the inclusion property. This further increases bandwidth requirement in EC.
10) *Selective bypassing:* In EC, as there is not any requirement that every block evicted from higher level cache must be filled in LLC, selective bypassing can be exercised, which is not possible in IC, as it violates inclusion property. There are many selective bypassing algorithms ([18],[19]) for E-LLCs. Good bypassing algorithms helps in reducing bandwidth demand and improves LLC capacity. Selective bypassing may or may not be possible in NIC depending on presence or absence of duplicated data.
11) *Coherence simplification and snoop filter requirement:* The data stored at all the levels in the hierarchy should be consistent. IC simplifies cache coherence problem because tag look-ups need only to be performed at the larger IC, to decide if a cache block is not present in the inner levels of the cache hierarchy. Thus coherence simplification ([4],[20]) is a major advantage of I-LLCs. Also IC acts as natural snoop filter. When an LLC look-up results in a miss, then no snoop needs to be send to upper levels, because it is guaranteed to not to be present there. Coherence and natural snoop filter benefits of IC are lost in EC and NIC organisation. To avoid coherence traffic, they maintain copies of higher level tag sets down with the LLC. Coherence probe needs to check all the cache levels in the hierarchy. Snoop filters ([21],[22]) can be used with LLC, but it increases the hardware overhead and verification complexity.
12) *State:* State of different cache types is in terms of extra hardware or software requirements. Need of separate coherence directory and snoop filters in EC and NIC makes their state more complex over IC.
13) *Back invalidation and inclusion victims:* In IC, whenever a line is evicted from any level, then inclusion is enforced by back-invalidating the same line in the levels above it in the hierarchy. These evicted lines are called as inclusion victims. In case of I-LLC, these inclusion victims are not chosen correctly because last level is generally unaware of the temporal locality of the lines in upper cache levels. When multiple applications compete for LLC, then number of inclusion victims increases. It also increases when size of LLC is not much larger than upper level's size. There are three Temporal Locality Aware (TLA) cache management policies [10] to make I-LLC to be aware of temporal locality of lines in upper cache levels. In EC, as there is not any redundancy concept involved, no back-invalidation of data is needed. This implies that there are no inclusion victims, thus improves the performance over I-LLCs. Also there are no inclusion victims in NIC, which performs similar to the QBS of TLA cache management policy.

14) *Performance:* Last parameter is to compare overall system performance in all three cache types. Limited performance of IC is due to presence of inclusion victims, not due to reduced cache capacity. Performance of NIC is high as compared to IC due to absence of inclusion victims. Increased capacity of EC leads to increased performance over both IC and NIC.

V. Conclusions

Multi-level caching is emerging as an important field in memory system architecture. A key factor to obtain good application performance in today's computer systems is to have a proper understanding of various properties of hierarchical memory system. One such property is Inclusion property. Based on this property, multi-level caches have been grouped into three categories- inclusive, non-inclusive and exclusive. In this paper, these three cache types have been critically examined and analyzed. Also, these have been compared on 14 parameters. It has been found that EC is best in terms of capacity, hit-rate and systems performance over both IC and NIC. IC performs best in speed and coherence management.

VI. Future Work

This study is limited in that it only examines the static nature of cache organisation scheme which is not useful when workload behaviour changes. Dynamic flexibility is needed to switch the inclusion property among inclusion, non-inclusion and exclusion with changing workload requirements. Future is directing towards bringing such run-time flexible cache design schemes.

References:

[1] A. J. Smith, "Cache Memories", *ACM Surveys*, vol. 14, No. 3, September 1982.

[2] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann Publishers, 2nd edition, 1996.

[3] P. J. Denning, "The locality principle", *Communications of the ACM-Designing for the mobile device*, vol. 48, No. 7, pp. 19-24, July 2005.

[4] J. L. Baer and W. H. Wang, "On the inclusion properties for multi-level cache hierarchies", *In ISCA-10*, pp. 73–80, 1988.

[5] M. M. Zahran, K. Albayraktaroglu, and M. Franklin, "Non-inclusion property in multi-level caches revisited", *International Journal of Computers and Their Applications*, vol. 14, pp 99–108, 2007.

[6] M. Zahran, "Cache Replacement Policy Revisited" , *WDDD*, 2007.

[7] R. V. Garde, S. Subramaniam, and G. H. Loh, "Deconstructing the Inefficacy of Global Cache Replacement Policies", *In WDDD*, 2008.

[8] K. Fletcher, W. E. Speight, and J. K. Bennett., "Techniques for Reducing the Impact of Inclusion in Shared Network Cache Multiprocessors", *Rice ELEC TR 9413*, 1995.

[9] N. P. Jouppi, "Improving direct-mapped cache performance by the addition of a fully associative cache and prefetch buffers", *In ISCA*, 1990.

[10] A. Jaleel, E. Borch, M. Bhandaru, S. C. Steely Jr., and J. Emer, "Achieving Non-Inclusive Cache Performance with Inclusive Caches: Temporal Locality Aware (TLA) Cache Management Policies", *In MICRO-43, MICRO '43*, 2010.

[11] M. J. Mayfield, T. H. Nguyen, R. J. Reese, and M. T. Vaden, "Modified L1/L2 cache inclusion for aggressive prefetch", U. S. Patent 5740399.

[12] Y. Zheng, B. T. Davis, and M. Jordan, "Performance Evaluation of Exclusive Cache Hierarchies", *In ISPASS*, 2004.

[13] T. M. Wong, Gregory R . G, J. Wilkes, "My cache or yours? Making storage more exclusive", November 2000.

[14] S. McFarling, "Cache Replacement with Dynamic Exclusion", *Proceedings of the 19th Annual International Symposium on Computer Architecture*, pp. 191-200, 1992.

[15] S. P. Vanderwiel and D. J. Lilja, "Data Prefetch Mechanisms", *ACM Computing Surveys*, vol. 32, No.2, pp. 174-199, 2000.

[16] D. Hardy, I. Puaut, "WCET analysis of multi-level non-inclusive set-associative instruction caches".

[17] N. P. Jouppi and Steven J. E. Wilton, "Tradeoffs in Two-Level On-chip Caching", *WRL Research Report 93/3*, October 1993.

[18] J. Gaur, M. Chaudhuri, and S. Subramoney, "Bypass and Insertion Algorithms for Exclusive Last-level Caches", *In Proceedings of the 38th International Symposium on Computer Architecture*, pp. 81–92, June 2011.

[19] M. Chaudhuri, J. Gaur, and N. Bashyam, "Introducing Hierarchy-awareness in Replacement and Bypass Algorithms for Last-level Caches", *PACT'12*, September 2012.

[20] X. Chen, Y. Yang, G. Gopalakrishnan, and C. Chou., "Reducing verification complexity of a multicore coherence protocol using assume/guarantee", *In FMCAD*, 2006.

[21] A. Agarwal, R. Simoni, J. Hennessy, and M. Horowitz, "An evaluation of directory schemes for cache coherence", *In ISCA*, 1988.

[22] R. Simoni , "Cache Coherence Directories for Scalable Multiprocessors", PhD thesis, Stanford University, Oct. 1992.

