Proposed Multi Level Cache Models Based on Inclusion & Their Evaluation

Megha Soni[#1], Rajendra Nath[*2]

[#1]*M.Tech., Department of Computer Science and Applications,*

Kurukshetra University, Kurukshetra, Haryana, India.

[*2]*Professor, Department of Computer Science and Applications,*

Kurukshetra University, Kurukshetra , Haryana, India.

***Abstract*— The performance of a multi-level cache hierarchy is decided by the number of cache levels following the inclusion or mutual exclusion property. Making all the cache levels in a hierarchy to behave either as inclusive or exclusive is not much beneficial. Significant performance improvement is possible if some of the cache levels are made to follow inclusion property while others not. Different cache design models are possible under this situation. This paper presents two proposed cache design models in a three level cache hierarchy. Our first proposed cache model involves making level1 and level2 as exclusive and level3 as inclusive with both of them. In second proposed cache model, level1 and level2 are taken as inclusive and level3 is taken as exclusive with level2. Both the proposed cache models have been experimentally evaluated and compared on six evaluation parameters using simulator which has been implemented in PHP-5.3. Proposed cache model 2 has been found to give better overall performance than proposed cache model 1.**

***Keywords*— cache memory, inclusive, exclusive, multi-level cache.**

I. INTRODUCTION

Scaling cache size with the advancement in on-chip technology is not possible. For this reason, computer architectures resort to multi-level cache hierarchies. Multi-level cache hierarchy is a promising approach for dealing with large cache sizes. But, still more sophisticated cache design techniques are needed with the increasing reliance on more on-chip caches. One such cache design technique that needs to be re-thinked includes the inclusion property of multi-level caches. The inclusion property states that in a multi-level cache hierarchy, the data held by upper cache levels must also be replicated to lower cache levels i.e. upper cache levels are subsets of those of lower cache levels. Therefore, multi-level cache inclusion represents the amount of duplicated data present in cache levels. This property is of great importance in multiprocessor environment as the introduction of cache hierarchy there could aggravate the well known cache coherence problem. Multi-level inclusion simplifies this problem by limiting the effect of coherence messages only in upper levels of the cache hierarchy and shields lower cache levels from their effect.

According to whether the multi-level cache follows the inclusion property or not, they are categorised into two main groups: inclusive cache and exclusive cache. Inclusive cache satisfies multi-level inclusion property and thus, data is present at different levels concurrently. In a three level cache hierarchy, this implies that L1⊂L2, and L2⊂L3. This data duplication decreases the effective cache capacity available for unique information but simplifies the cache coherence mechanism in multi-processors. Inclusive caches generally undergo the problem of frequent back-invalidations which in turn raises the number of inclusion victims and also the bandwidth requirement in cache hierarchies. Modern day Intel microprocessors like the Intel Core i7processors uses inclusive cache hierarchies.

Inclusive cache hierarchy reduces the overall system performance as there may be possibility that duplicated data will not be referenced again in future and thus, causes loss of useful data. To mitigate this problem, another multi-level cache design choice includes the use of exclusive caches. In exclusive cache hierarchy, data is not replicated from one level to another .This implies that L1⊄L2 and L2⊄L3. Exclusive cache hierarchy efficiently utilizes total cache area and also, modification of any data item at a cache level does not require modification at another level. Therefore, there is no back-invalidation and inclusion victim concept involved. Thus, exclusive cache hierarchy improves the system's performance with the limitation of difficult cache coherence mechanism. Processors of AMD Athlon and Operton, uses this cache hierarchy design choice.

Using both inclusive and exclusive cache choices, cache hierarchy could be designed in many ways. We have proposed two multi-level cache design models. In a three level cache hierarchy, our first proposed model involves level1 and level2 as exclusive and level3 as inclusive with both of them and in second proposed model, level1 and level2 are taken as

inclusive and level3 is taken as exclusive with level2. We have experimentally evaluated these two models using simulator which has been implemented in PHP-5.3 and compared both using various evaluation parameters like hit count, cache access time etc.

The rest of the paper is organized as follows: Section II. gives an overview of the related work on inclusive and exclusive cache hierarchies. Two proposed cache design models are presented in Section III. Both the proposed models is experimentally evaluated and compared in Section IV. Section V. concludes and summarizes the paper.

II. RELATED WORK

A multi-level cache hierarchy has remained an important concept since cache memories have been introduced. Baer et al. in [1] firstly studied the simplification of cache coherence protocol using multi-level inclusive cache hierarchies. Coherence problem was simplified, but low capacity and inclusion victims still remained a problem in inclusive caches. Different techniques were used to reduce the number of inclusion victims. Inclusion victims in direct mapped network caches were observed by Fletcher et al. [2]. Three solutions were proposed to minimise the problem. These were increasing the cache associativity, using a victim cache [3], or making the LLC non-inclusive and using a snoop filter to ease cache coherence. Additional hardware requirement still remained main problem. To further reduce the number of inclusion victims and minimise the hardware requirement, Jaleel et al. [4] proposed three 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. Junlin Lu et al. [5] proposed a two-level eviction priority policy. It appends an additional high level of eviction priority to LLC blocks besides the eviction priority provided by the baseline replacement policy. Theodore et al. [6] introduced a DEMOTE operation to achieve exclusive caching. To minimize the overheads inherited in DEMOTE operation, Gill et al. [7] proposed a better alternative called as PROMOTE which provides exclusive caching without demotions. Gaur [8] explored selective insertion and bypass algorithms for exclusive LLCs. As cache bypassing inherently breaks the inclusion property, S. Gupta [9] presented a solution to enabling cache bypassing for inclusive caches by introducing a bypass buffer to an LLC.

III. PROPOSED MULTI-LEVEL CACHE MODELS

In [10], we have seen a comparative study on two inclusion property based multi-level caches, where either we can apply inclusion property to all the cache hierarchy levels (inclusive cache hierarchy) or to none of them (exclusive cache hierarchy). To take benefits of both design choices, it is possible that we can apply inclusion property to some cache levels and mutual exclusion property to others, so that some cache levels behaves as inclusive cache while others behave as exclusive cache. Using this fact, a cache hierarchy could be designed in many ways. We have proposed two multi-level cache design models. In a three level cache hierarchy, our first model involves cache levels L1 and L2 as exclusive and L3 as inclusive with both of them (as shown in Figure 1) and second model involves cache levels L1 and L2 as inclusive and L3 as exclusive with L2 (as shown in Figure 2). To find out which proposed cache model gives best performance, we have compared them using various performance parameters like hit count, miss count, hit rate, cache access time, local and global miss rates.

Proposed Cache model 1: L1 and L2 exclusive and L3 inclusive.

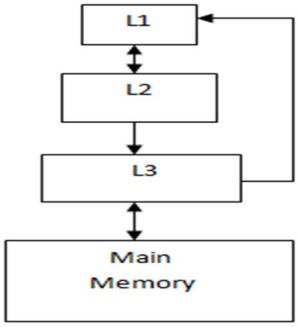

Figure 1. L1 and L2 exclusive & L3 inclusive

Proposed Cache model 2: L1 and L2 inclusive and L3 exclusive.

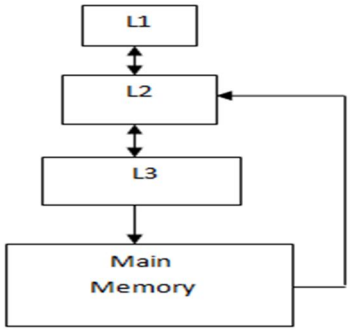

Figure 2 L1 and L2 inclusive & L3 exclusive with L2

A. *Evaluation Parameters for Proposed Cache Memories*

The two proposed cache models have been compared using following evaluation parameters:

1) *Hit count (h):* A hit is access to data already resident in cache memory. Number of hits can be calculated for different cache levels. Higher the hit count value better is the performance.

2) *Miss count (m):* If any cache access does not find resident data, miss is said to occur. This forces access

to next slower cache level and finally to main memory. Performance degrades with increase in miss count.

3) *Hit-rate (hr):* It is the fraction or percentage of time data is found in cache memory. Higher the hit-rate value better is the performance. For example, for L1 cache,

$$hr_{L1} = h_{L1} / a_{L1}$$

where hr_{L1} is the hit rate at level 1 and h_{L1} and a_{L1} is number of hits at cache level1 and number of accesses to level 1 respectively.

4) *Local Miss-rate (lmr):* Number of misses in the cache divided by number of accesses to this cache. Lower the local miss-rate value, higher is the performance. For L1 cache,

$$lmr_{L1} = m_{L1} / a_{L1}$$

where lmr_{L1} is the local miss rate at cache level 1 and m_{L1} and a_{L1} is the miss count for L1 and number of accesses to L1 respectively.

5) *Global miss-rate (gmr):* Number of misses in the cache divided by total number of CPU generated accesses. Lower the global miss-rate value, higher is the performance. For L1 cache,

$$gmr_{L1} = m_{L1} / a_{CPU}$$

Where gmr_{L1} is global miss rate at L1 cache and m_{L1} and a_{CPU} is the miss count for L1 and total CPU generated accesses respectively.

6) *Access time (t_a):* It is the total time required to access all blocks of reference string. Time can either be in terms of cycles or in nano-seconds.

$$t_a = (a_{L1} * c_{L1}) + (a_{L2} * c_{L2}) + (a_{L3} * c_{L3})$$

where t_a is the total cache access time, a_{L1}, a_{L2} and a_{L3} is the number of times L1, L2 and L3 cache is accessed & c_{L1}, c_{L2} and c_{L3} is the number of cycles required to access L1,L2 and L3 cache.

IV. EXPERIMENTAL EVALUATION

In this section we present a detailed quantitative evaluation of the two proposed cache models. Such an evaluation is important to compare the two proposed cache models and to find out which cache model gives better overall performance gain.

A. *Experimental Methodology and Setup*

To evaluate the performance of two proposed multi-level cache models, we have used a simulator which has been implemented in PHP version-5.3. PHP stands for: Hypertext Pre-Processor, is widely-used, open source server side scripting language, and a powerful tool for making dynamic and interactive web pages. Work has done using Apache HTTP Server version 2.2. PHP has support for a wide range of databases. Here, we have used open-source My SQL 5.6. Various assumptions have been made. Cache memory hierarchy has been supposed to consist of three cache levels - 8-entry fully associative L1, 16-entry fully associative L2 and 32-entry fully associative L3 cache. Again, we have assumed main memory as consisting of 1000 blocks and our reference string consisting of 771 blocks with some of them repeating after certain time interval. Common line size has been used for movement of blocks between cache levels. Normally, access to L1 cache takes about 3-4 CPU cycles, access to L2 cache takes 6-14 cycles and access to L3 cache takes 20-38 cycles. We have taken access time for L1 cache as 3 cycles, for L2 it is 11 and for L3, it is taken as 25 cycles.

In model 1, data in L1 cache and L2 cache are different and L3 cache holds data of L1 cache & L2 cache and some extra data. LRU block replacement policy has been used. Most recently accessed block is placed at top in L1 cache. Least recently used blocks are placed at bottom in L1 cache and evicted blocks are placed in L2 cache. If any block is referenced again, then it is moved at top in both L1 and L3 cache. Table I. shows simulation parameters used for our simulator.

Table I. Simulation Parameters

Parameters	Value
Main Memory Size (number of blocks)	1000
Reference String Size (number of blocks)	771 with some repeating
L1 Cache	8-entry fully associative
L2 Cache	16-entry fully associative
L3 Cache	32-entry fully associative
L1 Cache Access Time (cycles), c_{L1}	3
L2 Cache Access Time (cycles), c_{L2}	11
L3 Cache Access Time (cycles), c_{L3}	25
Block Size	Common for all cache levels
Policy used (by all cache levels)	Least recently used (LRU)

In Model 2, L1 cache is subset of L2 cache and data in L2 cache and L3 cache is different. Here, also LRU replacement policy has been used. Most recently accessed data is placed at top in both L1 and L2 cache and L3 cache holds blocks evicted from L2 cache. Thus, least recently accessed blocks are placed in L3 cache. Environment for Proposed Cache Model 2 has been configured in same way as for Proposed Cache Model 1.

B. *Experimental Results*

In this section, we present and analyze our simulation results and based upon resulting parameter values we compare two proposed multi-level cache models. Graphical representations of simulation results have been prepared using Microsoft Excel 2013.

1) Hit Counts (h): The first set of experiment deals with hit counts at all cache levels. Hit counts in L1, L2 and L3 caches for both proposed models are shown in Figure 3. It has been observed that hit count remains same in L1 cache. Large difference in hit count values have been seen at L2 and L3 caches. Proposed cache model 1 has largest L2 cache hit count value than cache model 2. For cache model 2, L3 cache has largest hit count value than cache model 1. In cache model 1, hit count value at L2 is high because it contains no duplicated data of L1 cache, which is there in cache model 2 causing its hits to be shifted from L2 to L3 cache. Thus, it is clear that exclusive LLC gives better performance than inclusive LLC. Therefore, cache model 1 performs better at L2 and cache model 2 performs better at L3. To find out which proposed cache model gives overall best performance at hit count parameter, the total hit counts for both the cache models are shown in Figure 4. Higher the hit count value better is the performance. It has been found that proposed cache model 2 performs better than proposed cache model 1.

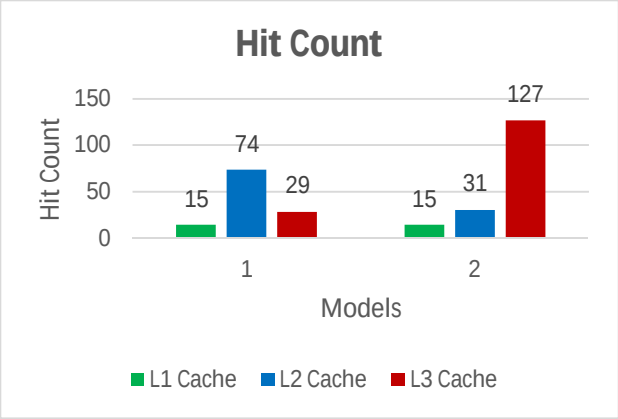

Figure 3 Individual Cache Hit Counts

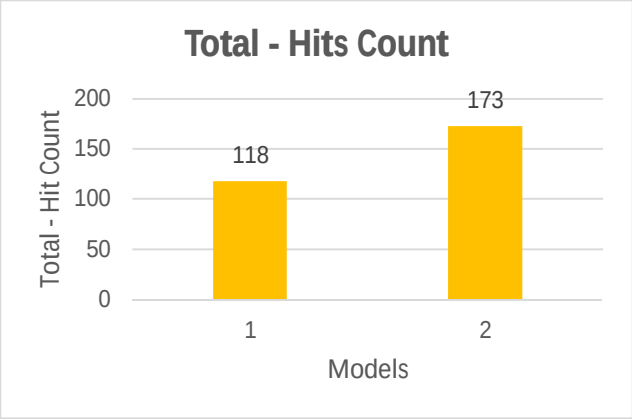

Figure 4 Total Hit Counts

2) Miss Counts (m): Miss counts in L1, L2 and L3 caches for both the cache models are shown in Figure 5.Total number of misses in L1 cache is same for both models. It has been found that model 1 performs better at L2 and model 2 performs better at L3. In order to find out which proposed cache model gives overall best performance at miss count parameter, the total number of miss counts for both the cache models is shown in Figure 6. Lower the miss count value better is the performance. Thus, model 2 gives overall better performance than model 1.

3) Cache Access Time (t_a): Figure 7. shows number of cycles used in L1, L2 and L3 caches for both our proposed cache models. It has been found that L1 and L2 caches are accessed the same number of times in both models. But the number of accesses at L3 cache level for model 2 is greater than model 1. This is because exclusive LLC (model 2) contains more unique entries than inclusive LLC (model 1) in cache hierarchy. Therefore, in model 1, miss at L2 cache level mostly leads to an access to costly main memory. In order to find out which proposed cache model gives overall best performance, the total number of CPU cycles used by both cache models are shown in Figure 8. Lower the number of CPU cycles used, better is the performance. Model 2 has been found to use 1075 extra CPU cycles than model 1. Thus, model 1 is better than model 2 for this evaluation parameter.

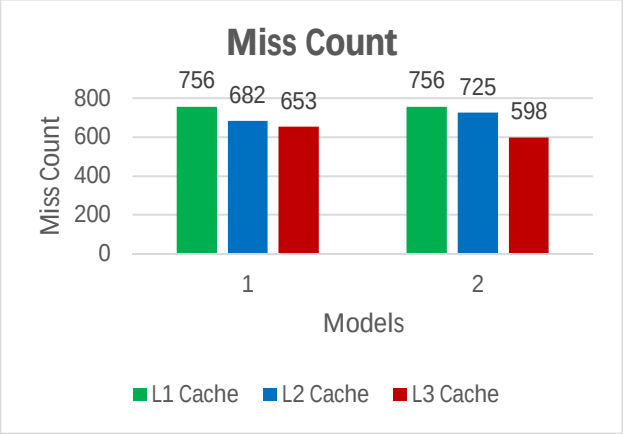

Figure 5 Individual Cache Miss Counts

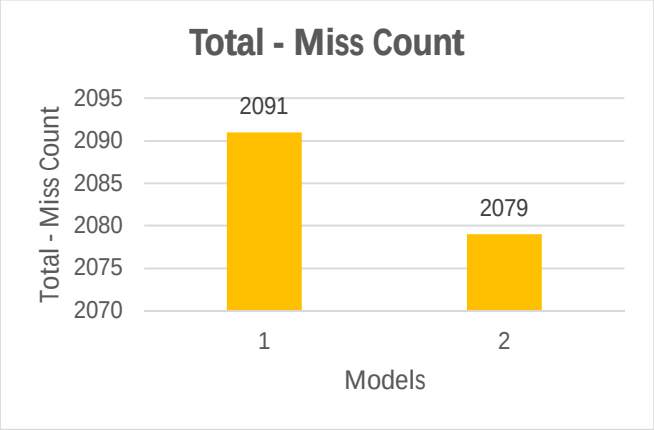

Figure 6 Total Miss Counts

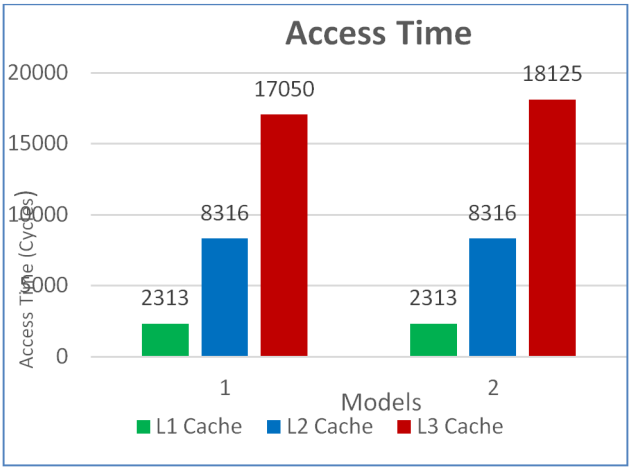

Figure 7 Number of Cycles Used in L1, L2 and L3 Cache

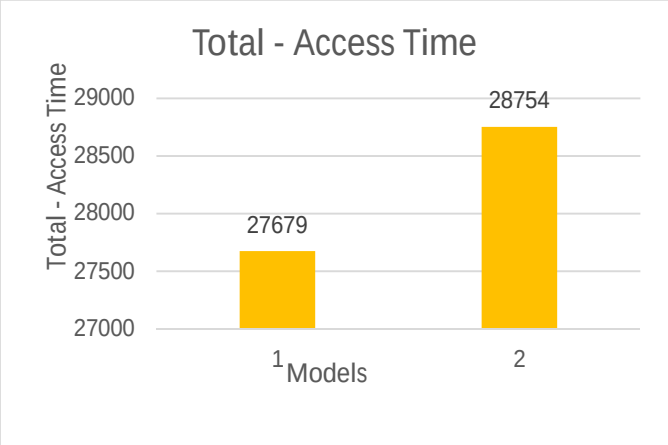

Figure 8 Total Number of Cycles Used

4) Hit – Rate (hr): Hit rates in individual cache levels for both cache models are shown in Figure 9. Again, hit rate is same for L1 cache in both the models. Hit rate at L2 and L3 cache levels is greater in model 1 and model 2 respectively for the obvious reasons. The total hit rates in both the proposed cache models for this performance parameter is shown in Figure 10. Higher the hit rate better is the performance. Thus, model 2 gives better overall performance than model 1.

5) Local Miss Rate (lmr): It depends on both the number of accesses to cache level and the number of misses occurring there. Lower the local miss rate values, better is the performance. For both proposed cache models, local miss rate in individual cache levels are shown in Figure 11. The number of accesses to slower LLC is more in model 2 but the possibility of occurrence of miss there is low causing local miss rate to go low at LLC (L3 here). Also, the number of accesses to L2 is same for both models, but in model 2, possibility of miss is high at L2 cache level causing local miss rate to go high there. Thus, model 1 performs better at L2 cache level and model 2 performs better at L3 cache level. Total local miss rates for two proposed cache models are shown in Figure 12. Cache model 2 has been found to perform better than cache model 1.

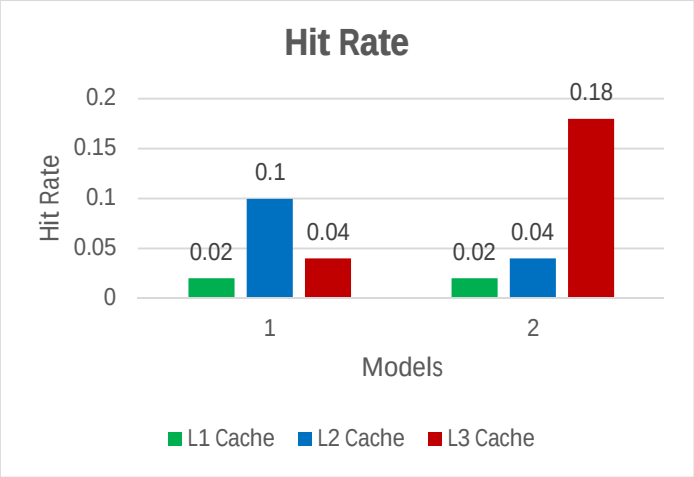

Figure 9 Individual Cache Hit-Rates

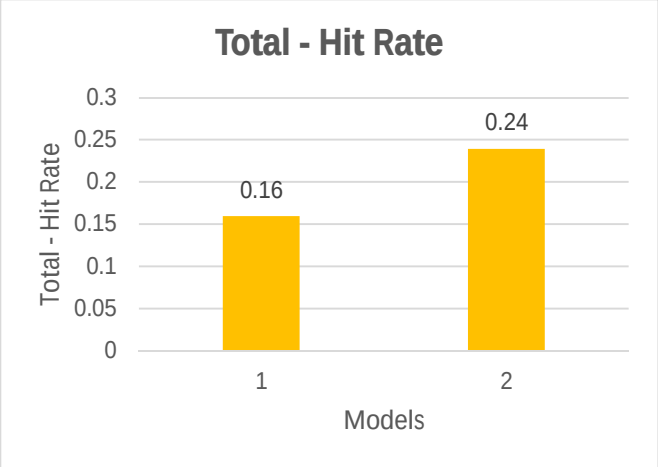

Figure 10 Total Cache Hit-Rates

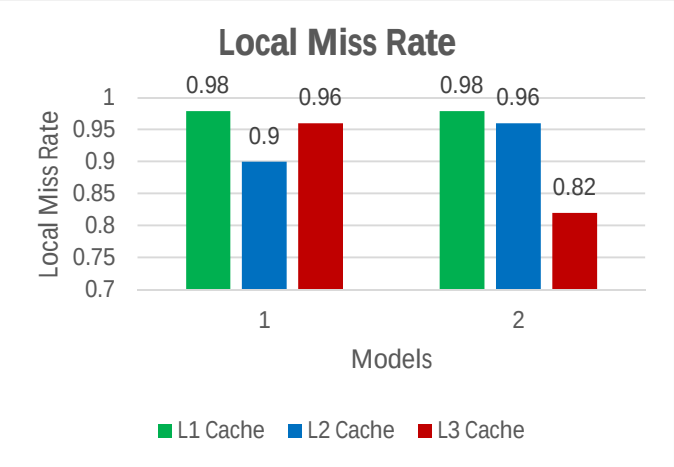

Figure 11 Local Miss Rates in L1, L2 & L3 Cache

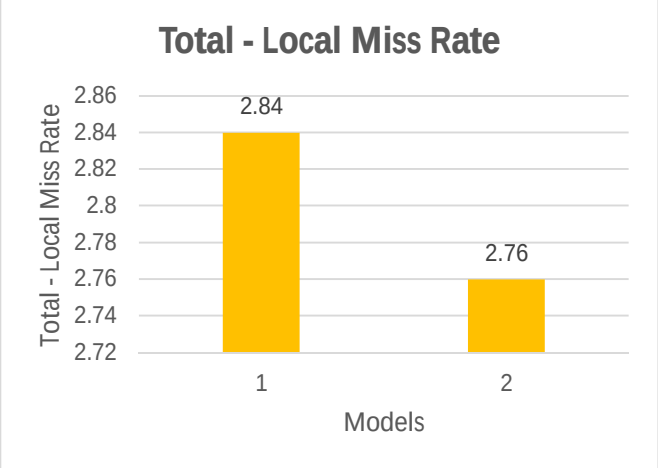

Figure 12 Total Local Miss Rates

6) Global Miss Rate (gmr): It is miss count at all cache levels divided by total number of CPU generated accesses. CPU generated accesses are fixed i.e. 771. Thus, global miss rate value depends only on number of misses occurring at each cache level. As the number of misses is more in inclusive L2 cache in Model 2 than exclusive L2 cache in Model 1, global miss rate is more at L2 in Model 2. Similarly, global miss rate is low at LLC in Model 2. Global Miss Rates for individual cache levels are shown in Figure 13. Therefore, model 1 performs best at L2 cache level and model 2 performs best at L3 cache level. Total Global Miss Rate for two cache models are shown in Figure 14. No large gap has been observed between the two model's global miss rates, but still model 2 gives better performance than model 1.

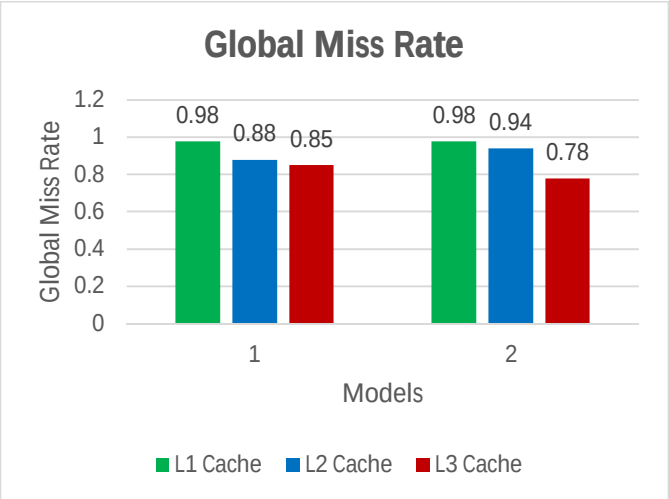

Figure 13 Global Miss Rates in L1, L2 & L3 Cache

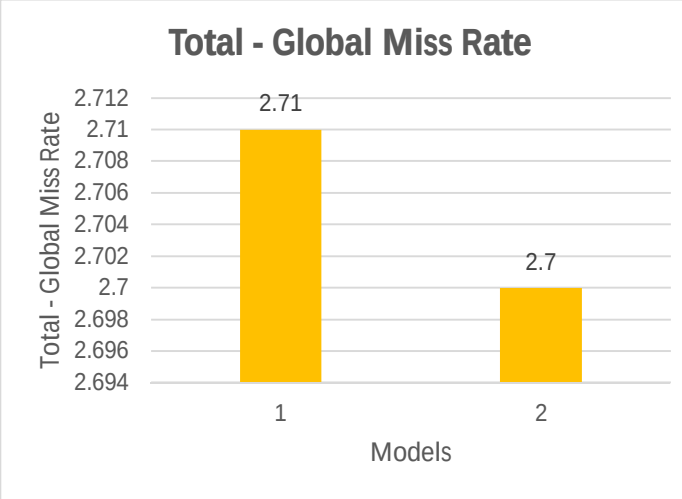

Figure 14 Total Global Miss Rates

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 concepts of hierarchical memory system. One such concept is multi-level cache inclusion. The inclusion property of cache hierarchies dictates that the contents of the upper level caches be a subset of those of lower level caches. According to whether this property has been enforced or not, multi-level caches have been grouped into mainly two categories- inclusive and exclusive. We've also proposed two cache models. The two proposed cache models have been experimentally evaluated using simulator which is implemented in PHP version-5.3. Our first proposed cache model involves making level1 and level2 as exclusive and level3 as inclusive to both level1 and level2 in a three level cache hierarchy and second proposed cache model involves level1 and level2 as inclusive and level3 as exclusive to level2. These two multi-level cache models have been compared using 6 parameters- hit count, miss count, hit rate, access time, local miss rate, global miss rate. It has been found that second proposed cache model gives best overall performance in terms of high hit count and hit rate and low local and global miss rates.

REFERENCES

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

[2] 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.

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

[4] 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.

[5] Junlin Lu, Xu Cheng, Zichao Xie, Lingda Li, Dong Tong, "Improving inclusive cache performance with two-level eviction priority," In IEEE 30th International Conference on Computer Design (ICCD), pp. 387-392, 2012.

[6] T. M. Wong and John Wilkes, "My cache or yours? Making storage more exclusive," USENIX Annual Technical Conference, pp. 161–175, 10–15 June 2002.

[7] B. S. Gill, "On Multi-level Exclusive Caching: Offline Optimality and Why promotions are better than demotions," FAST'08 Proceedings of the 6th USENIX Conference on File and Storage Technologies, No. 4, 2008.

[8] J. Gaur, M. Chaudhuri, S. Subramoney, "Bypass and insertion algorithms for exclusive last-level caches," ISCA-38, 2011.

[9] S. Gupta, H. Gao, H. Zhou, "Adaptive Cache Bypassing for Inclusive Last Level Caches," 2012.

[10] Zheng, Ying, Brian T. Davis, and Matthew Jordan, "Performance evaluation of exclusive cache hierarchies," In Performance Analysis of Systems and Software, 2004 IEEE International Symposium on-ISPASS, pp. 89-96, 2004.