VFAST Transactions on Software Engineering
http://vfast.org/journals/index.php/VTSE@ 2017 ISSN(e): 2309-3978;ISSN(p): 2411-6246
Volume 5, Number 1, January-December, 2017

pp. 50-57

ENHANCE THE PERFORMANCE OF ASSOCIATIVE MEMORY BY USING NEW METHODS

HAFSA MEHBOOB [1], HAMMAD AHMAD NIAZ [1]
[1]Department of Computer Sciences, University of Mangement and Technology, Lahore, Pakistan
Email: 15007108006@umt.edu.pk

ABSTRACT. *Data or instructions that are regularly used are saved in cache so that it is very easy to retrieve for the purpose of increase the cache performance. Evaluating the execution of multi-core systems the part of the cache memory is very important. A multicore processor is shared circuit in which two or more processors are joined to enhance the performance and perform multiple tasks. This paper describes the performance of cache memory based on cache access time, miss rate and miss penalty. Cache mapping methods are defined to increase the performance of cache but it face many difficulties. Some methods and algorithms are used to decrease these difficulties. In this paper describes the study of recent competing processors to evaluate the cache memory performance.*
Keywords:Cache memory, CPU memory.

1. **Introduction.** Associative memory is also called cache memory. Cache memory is a fast memory that is located in CPU and main memory. It enhance the execution time of program or instruction. Instructions that are regularly used are located in cache memory so it may be very easy to retrieve them for the purpose to increase the whole performance of the computer. There are various levels of cache.
Cache performance enhanced by decrease the miss rate, miss penalty and time to hit in the cache. That segment of memory which is not found in the cache is called cache miss and that section which is found is called hit rate. The time delayed by the CPU for a memory access through the time to exchange a block in the cache is called miss penalty. For this purpose, some techniques are used that are described in this paper.
The cache at first level is fastest and smallest, cache at last level is largest but slowest. In a processor first level cache is in the processor but second level cache and third level cache are on separate chip. In multi-core processors, every processor has its own first level cache but last level cache is shared by all the cores. Cache strategy design includes the CPU. It has three basic components these are instruction unit, execution unit, and storage unit. There are some components of storage unit these modules are a translator, Translation look-aside Buffer (TLB), Address Space Identifier Table (ASIT), Buffer Invalidation Address Stack (BIAS) and write Buffers.

In Section 2 discussed previous related work for the improvement in cache performance and review related work. There are different cache mapping techniques through which data from main memory is mapped on the cache and then used by the processor. Section 3 works on recent advances in the cache memory that solves the problems that occur in existing techniques. We work on some replacement algorithms and techniques these are Hybrid Cache Memory, Non-Uniform Cache Access, Energy Efficient Replacement Algorithm, Replacement, and Algorithm for Flash Memory and Adaptive Multilevel Cache Hierarchy.

2. Literature Review. To decrease the gap among processor speed and memory access latency many efforts are done. These efforts are established on hardware, compiler, and operating system. To decrease this gap many algorithms are applied.
Replacement Algorithms. To decrease miss rates and improve cache performance there are many

replacement algorithms. These are Least Recent Used (LFU), Least Frequently Used (LFU). When join LRU and LFU it becomes Least Recently plus Five Least Frequently it decrease the miss rate cache as compared to LRU and LFU. In this algorithm, certain values are assigned to every block these blocks are synchronized by both LRU and LFU algorithms. After that these values are joint with an algorithm for the purpose to assign a complete value to the block [11].

Direct mapping is fast access time as compared to other mapping techniques. But there is a problem that is conflict misses. This problem is solved by using some approaches. One of these approaches is column associative cache. In this approach increase a repeat bit for every cache set. Multiple addresses can be changed if they share similar memory place [12]. To improve the performance of direct cache mapping must retain the retrieving cache sets stability. For this purpose decrees the decoder size. Replacement algorithms and decoder increase the use of less cache retrieved. For every access as compared to direct mapping cache, stable cache reduces high power [10]. In target cache valid blocks are exchanged conflict misses may be minimized by these blocks. When a miss occurs in the main cache instead of going to main memory first data is searched in victim cache.

Cache Algorithms. CPU caches use the cache-oblivious algorithm in this information about cache parameters is not held. This algorithm improves performance on cache size because it does not hold information about cache parameters [2]. The strategy of replacement algorithms is improved like LRU. LRU choose new cache lines for replacement. When replace these cache lines checks that for LFU is out of suggested lines and then replaced by the low frequency. This algorithm shows good performance as compared to LFU [13]. Another method to decrease the failure rate is Block Passing. In last level caches, this algorithms is used. I n these algorithm feedback loops are used and supports to decrease miss conflict misses [5].

Design-Based Optimization. To decrease the gap among CPU sequence and memory latency by using a multi-level cache. When introduced the second level cache then failures of the first level cache may be decreased [8]. User Level Cache-Control (ULCC) is a user-level cache method is introduced in this method user have authority in the cache the distribution of space in their programs because of this later decrease cache trash. To assign cache space among virtual and physical pages a remapping section is introduced. The advantage to using this method is it provides fewer rates but very difficult in implementation [3] [6]. To improve the performance of cache many techniques are used. One of these techniques is creating multiple sets of cache for the purpose to retrieve multiple latencies. For this purpose Non-Uniform cache access (NUCA) strategy is suggested. In this strategy, the cache is organized in a way in which most data is served by rapid sets. Rapid sets are moved in stages with the help of switched network. The basic component of NUCA strategy is low latency access [7]. The performance of cache memory is affected by cache establishments. Jigsaw solved many problems like interference and scalability. In what way data is planned to segments is described in it also the area of virtual cache and shares is described in jigsaw strategy. Each segment has its different id. The performance of jigsaw is much better than NUCA [15].

Compiler and Prediction Based Optimization. Earlier cache optimization methods are used in which improve the loops over compiler. Retriever data may be adjusted in the cache because loops are decreased to the lesser area. Functions are allocated to one processor but only those functions which use same data, compiler examine all the process. All the functions which use same data are sequentially performed [5]. The compilers are used for the purpose to take results of cache with making suggestions for coming retrieved data. For the purpose of estimating which data is again used in the future compiler, algorithms are established. To increase the hit rates these estimations are very helpful. Evict-me is the estimation algorithm.

This algorithm uses cache line identifier of one bit. The cache line is exchanged if the evict-me identifier is fixed for some cache line [14]. All above methods are complicated in implementation and utilize more energy. For the implementation of set associative cache multiple techniques are used. One technique is predictive associative cache. When setting associative cache is implemented via predictive associative cache it takes access time that is equivalent to mapped cache.

To decrease access time and searching time many estimations are used but this technique takes low hit time [15]. To increase the cache performance generating the following data that is used with the cache. In the start, it is better to improve the data which is remaining used [13]. The use of cache is not reserved in attention for each processor when in a multi-processor any block is removed. To reduce this problem a technique is presented in which multiple processors have the right to place some constraints on every set of cache [11].

Web Based Optimization. The World Wide Web allows accessing the huge collective data items. The main issue in the web is server overloading. The items that are frequently used are reserved to positions secure the

clients. Some techniques are used for web cache optimization. These are Perfecting, cache replacement, and placement, cache contents, user access patterns, cache coherence, dynamic data caching, load balancing and proxy placement [4]. To increase the performance of cache heterogeneously assigned the cache area in Content Centric Network (CNN) [2].

Cache Design Strategy. The CPU has three basic components these are instruction, execution, and storage module. In instruction module, instructions are gain and translate. Execution module executing an instruction and done logical and arithmetic operations. The storage module creates an interface among other two modules. The main components of storage module are cache memory, translator, and Translation Look-aside Buffer (TLB). Address Space Identifier Table (ASIT), Buffer Invalidation Address Stack (BIAS) and write through buffers might be presented in the storage unit. Due to the technology it can be happen to put together lots of transistors that contains few part is needed to form a controlling CPU. On-chip, memory is located in the processor to decrease inter-chip data. Table 1 and 2 displays the different design policies of processors launched by Intel and AMD.

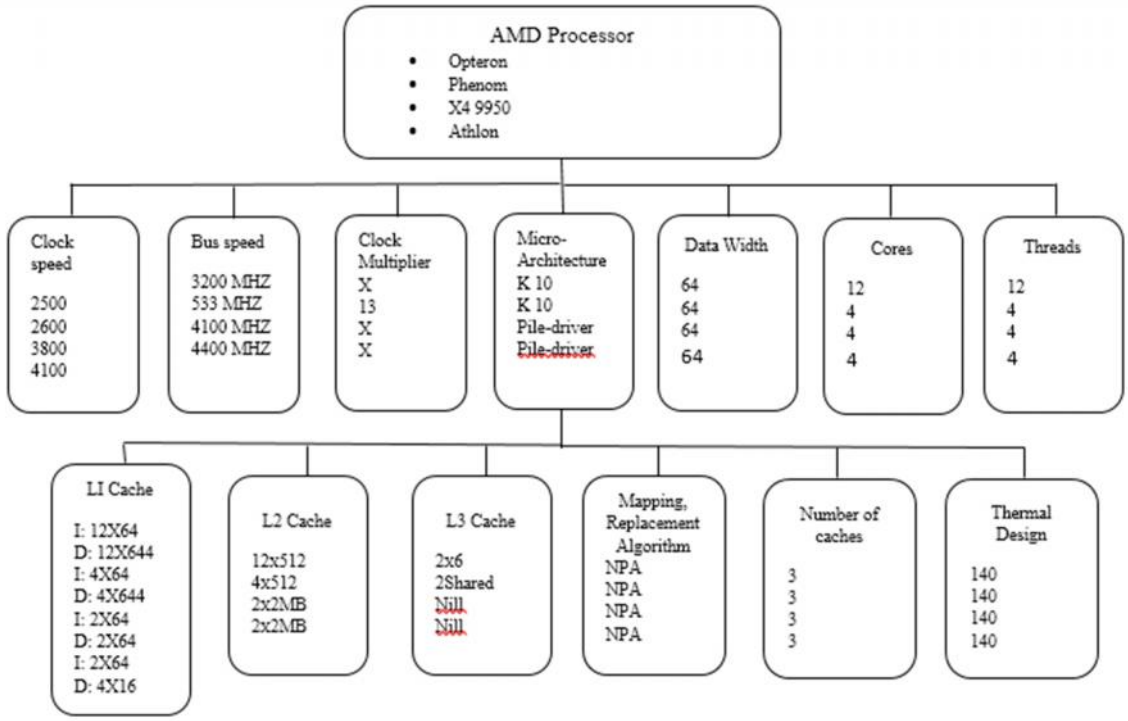

Figure 1. Specification of AMD Processor

For the purpose to define which block of main memory is now available in a specific cache line a mapping method is used. There are three mapping methods these are direct mapping, set mapping and set associative mapping. Due to the improved hit rate and fewer access time set associative cache considered best. It is described that after a definite boundary the effect of the increase in cache size is greater than an increase in associativity.

Replacement algorithm makes a decision in the cache memory design to choose a line of cache is exchanged by the wanted main memory block. The algorithms which are used for creating decisions are First in First out (FIFO) and Least Frequently Used (LFU). Because of easy execution and in which frequently recent words are used are expected referred again Least Frequently Used algorithm is a more efficient algorithm. In what way reliability is sustained among cache lines and interrelated blocks in main memory is decided by some policies. These policies are Write Caching, Write Back and Write Through. In write-back policy write operations are done to cache memory only and when the related cache line is removed from cache memory then the main memory is updated. The result of write back policy may be an inconsistency if two caches grip similar line and the line is updated in one cache but the other cache will grip an invalid value. In write-through policy, processes are done to both main and cache memory. Inconsistency can also arise in

the write through policy. In both write-back policy and write-through policy huge traffic is produced, a single bit fault cannot be accepted without Error Correcting Code (ECC) is delivered. Write caching is an arrangement of both caches in which associative cache is stored in a write-through memory.

There are three levels of cache. Level 1 cache is fastest and smallest. It contains Level 1 data cache and Level 1 instruction cache. The unified cache shows a greater hit rate but split cache does not due to the adjustment of load between fetched instructions and data. The difficulty of contention among fetch/decode and execution is resolved by split cache design.

The performance may be decreased by interfering with instruction pipeline. Mostly processors execution attributes for example replacement algorithms, mapping function and write policies may be not easily accessible. Intel x86 processors and AMD processors utilize a direct mapping level 1 cache and level 2 cache is among 2 to 4 way set associative. Level 3 cache and higher level caches might be among 16-way to 64-way set associative. Mostly caches need the least recently replacement and a write-back cache policy without complete change.

Hybrid Cache Memory Based on SRAM-MRAM. Magnetic Random Access Memory (MRAM) and phase-change Random Access Memory (PRAM) are non-volatile so creating them appropriate for upcoming computing. They have limited strength and long write invisibility. Hybrid cache memory is strongly investigated to reduce these problems it containing volatile and non-volatile memory.

Features of SRAM are presented in Table 3. Many cache technologies are best in one field on the other hand drop in other fields. Cache is designed to using different technologies. Hybrid memory system design and procedure process give good performance and energy efficiency when the size of cache becomes larger and experienced.

Some abilities are evaluated by Wu al [1] created on Hybrid cache design provides accommodations on-chip orders. In his study, two architectures are included these are Level Hybrid Cache Architecture (LHCA) and Region based (RHCA). LHCA is inter-cache level. This is used to determine memory technology for the creation of levels within the cache and over standard cache it provides 7% regular mean IPC (instruction per cycle) enhancement.

Non-Uniform Cache Access. When the size of the cache is larger the processor quickly approach nearly data. Problem of data creates due to the data that is not near to the processor. NUCA (Non-uniform cache access) address this data. An interchanged network is used by NUCA that allows to shift multiple cache areas that are created on the access rate.

In Fig 1 Kim et al [4] describes distinct Level 2 cache designs. In figure 1 part (a) describes constant cache design or traditional design. When the size is over 4MB traditional design gives poor performance due to the limited ports and internal wire delays. Part (b) describes ML-UCA is a multi-level cache based on Level 2 and Level 3. Both levels of Cache are saved to support various related access. Inclusion is required so extra space is a waste in this design. In figure 1 (c) to reduce this problem shows a cache supportive non-uniform access. In which placement of data in many banks is hide through statically determining banks? This cache is known as S-NUCA 1.

A wide study first of all to relate the performance of each caches by a set of 16 benchmarks were approved out by Kim et al [5]. In Fig 1 displays comparative estimation of 16 MB/50nm IPC gained after UCA, S-NUCA-1, ML-UCA, S-NUCA-2, DN-best (best D-NUCA policy), and an ideal D-NUCA upper bound. The difficulty of important area above existing in the S-NUCA-1 larger range of banks can decrease partitioning of banks. In figure 1 part (d) describes this problem by S-NUCA-2 architecture. Cache access techniques are changed too quickly work at a lower level that has smaller size.

DN is the best on all benchmarks but grid, GCC. DN-best provides extra constant performance. The IPC result of DN-best was found only 16% bad. DN-best performance is near to best. When cache size cross some limits then NUCA memory systems provide more rapidly cache access but UCA does not provide rapid access. Flexibility and scalability of NUCA structures is important for chip multiprocessor architecture (CMP).

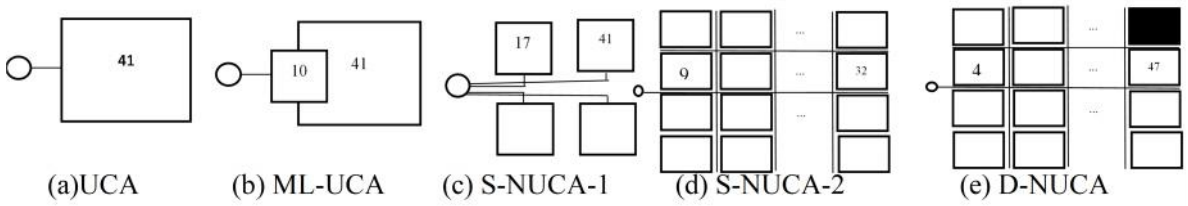

Figure 2. Architecture of Level-2 Cache

Energy Efficient Replacement Algorithm. In data centers storage systems utilize important energy floating energy consumption fears and for disks, it utilizes important energy situation for power controlling arrangements [6]. PA-LRU is a power conscious replacement algorithm [7]. It was suggested for a long period of time in the storage cache far away from active blocks from inactive disks. Because of their capability to stay for a longer time in low power methods and then active but the inactive only short period of time. So that's why the energy utilizes by the disks is decreased. In PA-LRU for to each assignment, it is essential awkward change. PB-LRU (partition based LRU) it is suggested by Zhu et al [8] for the purpose to create modifications easy. Cache break into single partitions these partitions individually handled. This is done by replacement algorithm. Every single disk for cache memories new strategy of energy effective replacement algorithm require.

Replacement Algorithm for Flash Memory. Use of flash memories grows into extra visible because of their tiny lightweight System and low power consumption ability. In most operating systems a number of
Memory hits are the only fear storage systems. Replacement is recognized by for flash memory replacement algorithm. This cost is produced via choosing dull victims as well hit count. Park et al [9] suggested Clean-First LRU (CFLRU) replacement algorithm. This algorithm is divided into two areas first is working area and second is the clean-first area. In the clean first area, this algorithm removes clean pages. When valued in buffer cache that is created on a file system and access block is used then for swap systems average replacement cost is 28.4% minimize and 26.2% minimization in the buffer cache. When banned closures happen there is a major issue occur. This issue arises when in the SDRAM cache both algorithms CFLRU and LRU recall dull pages. So it is proposed that CFLRU algorithm is used by the file system. In flash memory improved performance of replacement algorithm upcoming algorithms must be combined.

Adaptive Multi-Level Cache Hierarchy. Effective cache hierarchy is crucial to reduce the invisibility time and enhance the performance.16 core CMP cache topology in which L2 portion is joint by X cores and their L1 caches to each of Y number, the L3 portion is joint by Y number of the L2 portion to each of Z number. These are symbolized such as (X: Y: Z).The greatest alignment differs with time throughout the implementation of the workload when the maximum quantity of material [10]. When dealing with different kind of applications the idea one cache topology is enough for all is not right so to provide this facility Shekhar et al [10], suggested an adaptive multi-level cache order that is reconfigurable and called Morphe Cache. In Morphe cache exists multiple cache topologies. These topologies have permission to organize by the similar design with dynamical change in a CMP multi-level cache topology. When using a 16 core CMP and the multi-threaded and multi-programmed workloads normal quantity and harmonic mean is estimated then it is enhanced by Morphe cache. Reconfigurable Morphe cache (16 cores) is not working appropriately provides good results by processors that have smaller cores. Dynamic cache designs must be essential to execute when it is necessary to complete the requirements of that processors which have further 16 cores.

Memory Hierarchy Programming. Toward the growth in the amount of similar processing components and designed for effective use of memory bandwidth, it is essential to improve different software design concepts to manage the memory in multi-core processors to increase the attention. For the transmission of data among memories on and off the chip clear memory managing is necessary to enable program accuracy. Fatahalian et al [11] suggested a software design model named as Sequoia. In this model, a programmer has a clear authority of the movement and location of memory hierarchy on every stage of the data. Through taking an applied method that is moveable similar programming Sequoia model offers restricted set of concepts. This restricted set may be professionally applied. In programs, hierarchical grouping is needed because in this way

parallel divide-and-conquer plans and hierarchy are inspired. In a chain to admit opinions, every task is known as space limited method, therefore, conquering a smaller amount of storage. For the coming memory hierarchies, Sequoia has an unlimited scope but improvements are possible in it.

3. Comparative analysis. In this paper we discussed a few recent improvements in cache memory for the purpose to increase access time and energy consumption of CPU. To save the energy, decrease the spin-up of data drives for this purpose MAID replacement algorithm use the cache drives [6]. To control the cache by some nodes in light weight reconfigured the cache.

Several companies had suggested, use on-chip cache memories through chip microprocessors. They consist of CMP-SNUCA [18], CMP-NuRAPID [17], Victim Replection [19], and CMP-CC [18]. Apply the NUCA method in CMP-SNUCA arrangement. On-chip cache companies transfer blocks near to the receiver to decrease the wire delays. Hybrid caches described placement constraints for one division of data. It requires coherence tools and difficult lookups to enhance the latency [19].

Table 1. Comparative analysis of different techniques

	Advantages	Disadvantages
Hybrid Cache Memory Based on SRAM-MRAM	• It provides good performance when the size of cache increase • Architecture is used to determine memory technology for the creation of cache levels	• It has Low Density • High Leakage • Its strength is limited
Non-Uniform Cache Access	• It provide user friendly programming • Data sharing between tasks is very fast	• It provides less scalability between memory and CPU • It is more difficult and expensive
Energy Efficient Replacement Algorithm	• Energy is utilized by power controlling arrangements • It is a power conscious Algorithm	• It requires long period of time • It is inactive only short period of time
Replacement Algorithm for Flash Memory	• It becomes visible rather than other Algorithms • It requires smaller size	• Multiple cache topologies are required • It is not working good when size is larger than 16 cores
Memory Hierarchy Programming	• It has restricted set of concepts • Hierarchical groups are required	• Effective use of Bandwidth • Unlimited scope

Conclusion. The performance of processors has increased the advances in micro-architecture helped with semi-conductor technology but not equivalent to the performance improvements in memory. Because of excellent architecture policies and integration microprocessors floating-point ability is improved but the speed of the memory is not improved. So cache memory is used to decrease memory access time, average bandwidth and memory latency. In this paper describe some new improvements in the cache memory to locate latency, energy consumption and increase access time of memory controlling component. For the cache, optimization estimated the performance of many techniques and algorithms.

Our finding is encapsulated in Table 1. It is difficult to find the best cache optimization method in all cases. Every method has its own design, benefits, and problems. Some techniques might be improved. For example,

the use of bigger block size, bigger cache and estimation techniques can be decreased the rate of conflict misses. So the use of bigger block size can decrease hit time and power consumption and improve the miss penalty. The Bigger cache provides extra cost and slowest access time. It is related to cache coherence issue. Greater associativity provides quick access time, however, lowest cycle time. Victim cache decreases miss rate on extra cost. For the purpose to decrease cache misses at a greater rate as compared to LRU, FIFO, LFU methods LR plus five LF is a good method. So LR plus five is an extra difficult method. ULCC removed the cache pollution and quick access time but greater in difficulty. Pipelining is very difficult technique miss penalty is decreased by it. In the future, our research direction is to improve the multi-level cache through addressing its coherence problems.

REFERENCES

[1] Wu, X., Li, J., Zhang, L., Speight, E., Rajamony, R., & Xie, Y. (2009, June). Hybrid cache architecture with disparate memory technologies. In *ACM SIGARCH computer architecture news* (Vol. 37, No. 3, pp. 34-45). ACM.

[2] Farooq, M. S., Khan, S. A., Ahmad, F., Islam, S., & Abid, A. (2014). An evaluation framework and comparative analysis of the widely used first programming languages. *PloS one*, *9*(2), e88941.

[3] Lee, B. M., & Park, G. H. (2012, November). Performance and energy-efficiency analysis of hybrid cache memory based on SRAM-MRAM. In *SoC Design Conference (ISOCC), 2012 International* (pp. 247-250). IEEE.

[4] Khan, Y. D., Khan, N. S., Farooq, S., Abid, A., Khan, S. A., Ahmad, F., & Mahmood, M. K. (2014). An Efficient Algorithm for Recognition of Human Actions. *The Scientific World Journal*, *2014*.

[5] Kim, C., Burger, D., & Keckler, S. W. (2002, October). An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches. In *Acm Sigplan Notices* (Vol. 37, No. 10, pp. 211-222). ACM.

[6] Abid, A., Hussain, N., Abid, K., Ahmad, F., Farooq, M. S., Farooq, U., ... & Sabir, N. (2016). A survey on search results diversification techniques. *Neural Computing and Applications*, *27*(5), 1207-1229.

[7] Ahmed, M. W., & Shah, M. A. (2015). Cache Memory: An Analysis on Optimization Techniques. *International Journal of Computer and IT*, *4*(2), 414-418.

[8] Ding, X., Wang, K., & Zhang, X. (2011, February). ULCC: a user-level facility for optimizing shared cache performance on multicores. In *Acm sigplan notices* (Vol. 46, No. 8, pp. 103-112). ACM.

[9] Farooq, M. S., Khan, S. A., Ahmad, F., Abid, K., Farooq, U., & Abid, A. (2014). A Survey On Diversification Techniques For Unabmiguous But Under-Specified Queries. J. Appl. Environ. Biol. Sci, 4(7S), 271-276.

[10] Abid, A., Hassan, B., Abid, K., Farooq, U., Farooq, M. S., & Naeem, M. A. (2018). Sports Culture in South Asia: Effects of Modern Bowling Action Rules on Cricket, an Information Technology Perspective. *South Asian Studies*, *33*(1), 211-220.

[11] AbdelFattah, A., & Samra, A. A. (2012). Least recently plus five least frequently replacement policy (LR+ 5LF). *Int. Arab J. Inf. Technol.*, *9*(1), 16-21.

[12] Naseer, A., & Farooq, M. S. (2015). EXPLORING CAUSES OF REQUIREMENT CHANGE. *VFAST Transactions on Software Engineering*, *4*(2), 1-8.

[13] Beckmann, N., & Sanchez, D. (2013, September). Jigsaw: Scalable software-defined caches. In *Parallel Architectures and Compilation Techniques (PACT), 2013 22nd International Conference on* (pp. 213-224). IEEE.

[14] Xu, Y., Li, Y., Lin, T., Wang, Z., Niu, W., Tang, H., & Ci, S. (2014). A novel cache size optimization scheme based on manifold learning in content centric networking. *Journal of Network and Computer Applications*, *37*, 273-281.

[15] Bauer, M., Clark, J., Schkufza, E., & Aiken, A. (2011). Programming the memory hierarchy revisited: Supporting irregular parallelism in sequoia. *ACM SIGPLAN Notices*, *46*(8), 13-24.

[16] Chen, T. Y., Yeh, T. T., Wei, H. W., Fang, Y. X., Shih, W. K., & Hsu, T. S. (2012, November). CacheRAID: An efficient adaptive write cache policy to conserve RAID disk array energy. In *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing* (pp. 117-124). IEEE Computer Society.

[17] Farooq, M. S., Khan, S. A., Ahmad, F., Abid, K., Farooq, U., & Abid, A. (2014). A Survey On Diversification Techniques For Unabmiguous But Under-Specified Queries. J. Appl. Environ. Biol. Sci, 4(7S), 271-276.

[18] Bauer, M., Clark, J., Schkufza, E., & Aiken, A. (2011). Programming the memory hierarchy revisited: Supporting irregular parallelism in sequoia. *ACM SIGPLAN Notices*, *46*(8), 13-24.

[19] Chen, T. Y., Yeh, T. T., Wei, H. W., Fang, Y. X., Shih, W. K., & Hsu, T. S. (2012, November). CacheRAID: An efficient adaptive write cache policy to conserve RAID disk array energy. In *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing* (pp. 117-124). IEEE Computer Society.

[20] Chishti, Z., Powell, M. D., & Vijaykumar, T. N. (2005, June). Optimizing replication, communication, and capacity allocation in CMPs. In *Computer Architecture, 2005. ISCA'05. Proceedings. 32nd International Symposium on* (pp. 357-368). IEEE.

[21] Chang, J., & Sohi, G. S. (2006). *Cooperative caching for chip multiprocessors* (Vol. 34, No. 2, pp. 264-276). ACM.

[22] Hardavellas, N., Ferdman, M., Falsafi, B., & Ailamaki, A. (2009). Reactive NUCA: near-optimal block placement and replication in distributed caches. *ACM SIGARCH Computer Architecture News*, *37*(3), 184-195.