NUMERICAL ANALYSIS OF CACHE CHARACTERIZING PARAMETERS INCLUDING SIMULATION

[1]Fengyuan Zhang (Northeastern University, China)
[2]Jiarui Chang (Dalian #24)
[3]Guihongxuan Zhang (Pius XI)

Abstract

Cache plays an important role in characterizing the overall performance of the whole system. It is a crucial level of memory, which is being referred by Central Processing Unit, whenever there is a requirement of the data for further processing. In the present work various parameters related to the cache characterization are analyzed, further, one of the web-based simulators is used for better understanding of the Cache behavior.

Keyword: Cache Oblivious, associative memory, threads, parallel searching, locality of reference, block size

1. Introduction

Although CPU has registers to store/process the data in an efficient manner, as registers are onboard, and their working speed is very high. But a limited number of registers can be used, and they are also very costly. Here the role of cache memory comes which acts as an interface between the primary memory and the CPU [1]. The data which is frequently used can be stored in the cache memory so that more data requests can be served easily without interacting with the primary memory whose working speed is less. Present work analyses various cache characterization parameters.

Average memory access time = Hit time + Miss rate x Miss penalty

As per the formula to increase the performance, there must be less miss rate, less miss Penalty and there must be a reduction in hit time [2]. In the present research work, a certain key point related to the cache is highlighted. Further wed based educational tool (Simulator), which is developed by the students who have attended the class of Professor Israel Koren UMass Dept. of Electrical and Computer Engineering is used [3]. The Simulator is designed in a very effective manner and helps a lot in studying the cache behavior. So, the design requirement is that there must be a largest possible primary cache without decreasing the clock or increment in the pipeline stages. There are many causes for which cache misses can occur, a compulsory miss that occurs when there is the first reference to a block and it will always be there even when the cache size is too big [4]. The second type of misses arises due to the capacity of a cache when it is not capable to store complete data. Block replacement strategy can also result from collisions and thus a number of misses increases. Threads play an important role for doing parallel searching even in case of cache(Associate) [5]. Cluster-based image filtering techniques [] along with cache oblivious algorithms is a great combination for producing of efficient results.

2. Literature Survey

Kaur et al. [7] in their research work use the cache efficient algorithm for denoising the large sized images on the cluster. By taking into consideration the locality of reference the number of cache misses can be decreased and on the same hardware better results can be produced.

Bagga et al. [1] in their research paper discuss cluster based programming using cache oblivious algorithm for matrix multiplication. Their research work explores that algorithms

working by taking into consideration memory hierarchy are very much efficient and approximately three times much better results in terms of execution time can be produced. For an increase in the number of cache size the miss rate decreases accordingly.

Ghosh and Mukhopadhyaya [8] raise the idea that algorithm for any task can be divided into two categories: one is cache-dependent and other is cache-independent, also known as cache oblivious. Proposed work also concluded that algorithmic strategy taking into consideration memory hierarch produces better results and helps in decreasing execution time.

Kumar et al. [9] in their research work explain the benefits of cache oblivious algorithms. Their research works prove that the miss rate corresponding to the recursive matrix multiplication is very much less as compared to the normal multiplication miss rates.

3. Present Work

In the present work initially, a simulation study is done using the web-based simulator [3] [10] and the key points related to the cache are discussed. general workflow of the memory hierarchy their corresponding access time and storage time is shown in Fig.1. Fig.2 represents the effect on cache misses with an increase in cache size.

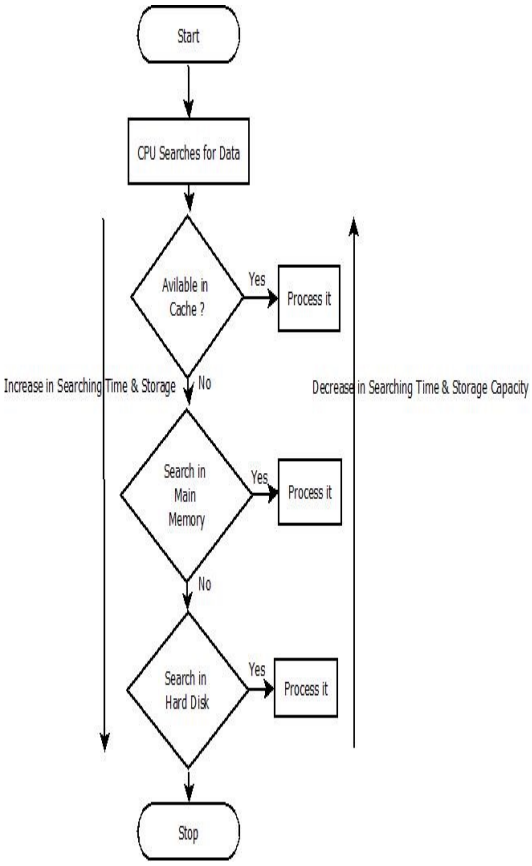

Fig.1: Data access corresponding to the memory hierarchy

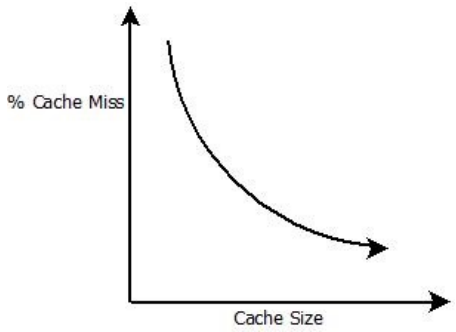

Fig.2: Effect of Cache Size on % of Cache Misses.

In the first case various cache characterizing parameters like address bit pattern, its partitioning based on main memory size, cache memory size, block size etc. are studied. Then in the second case, average access time corresponding to the various read or write policies are discussed. In the third case study related to how data is accessed to and from the cache for various Replacement policies like LRU, FIFO is discussed. Then in the last, a number of key facts are discussed after analysis for increasing system performance.

Case-1 Study of address bit pattern and its partitioning based on main memory size, cache memory size, block size etc

Equations/formulae used:

Number of blocks in cache = Cache size / Block size

- Number of bits in Tag = Total bits - Index bits - Offset bits
- Number of sets in cache = Cache size/ (Set size * Block size) • Number of bits in Tag = Total bits - Index bits - Offset bits

Table 1. Representing various Parameters of Cache Address Structure

S.No.	Memory Size	Cache Size	Block Size	Cache Scheme	Set Size	Number of blocks /Sets in cache	Number of bits in Tag
1.	528KB	128KB	4B	Direct Mapping	4 Blocks	2^{15}	2
2.	1MB	256KB	4B	Set Associative	4 Blocks	2^{14}	4
3.	2 MB	128KB	8B	Direct Mapping	8 Blocks	2^{14}	4
4.	4MB	256KB	8B	Set Associative	8 Blocks	2^{12}	7

i. In the Cache, **Compare Bits** are compared with the Tag Bits.
ii. The **Set Select Bits** are used to select a particular Set in the Cache.
iii. The **Byte Select Bits** are used to select a particular byte in the accessed block.

Case 1.1

Memory size = 528KB = 2^{19}
Block size = 4Bytes = 2^2

- Number of blocks in cache = Cache size/Block size = 128KB/4B = $2^{17}/2^2 = 2^{15}$
- Number of bits in Tag = Total bits - Index bits - Offset bits = 19-15-2 = 2

Case 1.2

Memory size = 1MB = 2^{20}
Block size = 4Bytes = 2^2

- Number of sets in cache = Cache size/ (Set size * Block size) = 256KB/ (4 blocks * 4B) = 218/ (22 * 22) = 214
- Number of bits in Tag = Total bits - Index bits - Offset bits = 20-14-2 = 4

Case 1.3

Memory size = 2MB = 2^{21}
Block size = 2Bytes = 2^1

- Number of blocks in cache = Cache size/Block size = 128KB/2B = $2^{17}/2^1$ = 216
- Number of bits in Tag = Total bits - Index bits - Offset bits = 21-16-1 = 4

Case 1.4

Memory size = 4MB = 2^{22}
Block size = 8Bytes = 2^3
Number of sets in cache = Cache size/ (Set size * Block size) = 256KB/(8 blocks * 8B) =

$2^{18}/(2^3 * 2^3) = 2^{12}$

Number of bits in Tag = Total bits - Index bits - Offset bits = 22-12-3 = 7

TAG		INDEX														OFFSET	
18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Compare Bits		Set Select Bits															Byte Select Bits	

Fig.2: Showing Simulated view of Address Partitioning for S.No.1 [10]

TAG				INDEX												OFFSET	
19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Compare Bits				Set Select Bits														Byte Select Bits	

Fig.3: Showing Simulated view of Address Partitioning of S.No.2 [10]

TAG				INDEX												OFFSET		
20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Compare Bits				Set Select Bits														Byte Select Bits		

Fig. 4: Showing Simulated view of Address Partitioning of S.No.3 [10]

TAG							INDEX												OFFSET	
21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Compare Bits							Set Select Bits												Byte Select Bits	

Fig. 5: Showing Simulated view of Address Partitioning of S.No.4 [10]

Case- 2 Study of average access time due depending on the read or write policies

Equations/Formulae Used

i. Read Hit Contribution(RHC) = %Reads * Hit_rate * HitTime
ii. Read Miss Contribution(RMC)= %Reads*MissRate*((MissPenalty+Hit Time) +(%Dirty*MissPenalty))
iii. Write Hit Contribution(WHC) = %Writes * HitRate * HitTime
iv. Write Miss Contribution(WMC)= %Writes * MissRate * MissPenalty
v. Total Average Memory Access Time (TAMAT) = (Read Hit Contribution +Read Miss Contribution + Write Hit +Write Miss Contribution)/4

Table 2. Values of various Cache Characterizing parameters

S.No.	Cache Size: KByte	Associativity Sets	Block Size Bytes	Write policy	% Writes	% Dirty	Miss Penalty (cycles)	Hit Time	Mem. Write (cycles)	RHC	RMC	WHC	WMC	TAMAT
1.	4	8	64	Write Back/ No-Write/ Allocate	22	10	40	1	6	0.75	1.32	.21	.33	2.62
2.	8	4	64	Write Back/Allocate on Miss	30	15	45	2	8	1.34	1.37	.577	.588	3.88
3.	32	8	128	Write Through/No-Write Allocate	35	20	40	3	9	2.07	.37	2.66	.03	5.1
4.	256	8	128	Write Back/ Allocate on Miss	30	25	35	4	8	2.7	.09	1.1	.04	4.1

Table 3. Showing Cache Time Analysis S. No 1[11]

Cache Time Analysis					
Cache Specification:	Cache Size = 4 KB	Associativity = 8	Words / Block = 16	Hit Rate = 0.9623	Miss Rate = 0.0377
Parameters:	% Writes = 22%	% Reads = 78%	% Dirty = 10%	Hit Time = 1	Miss Penalty = 40
Write Policies:	Write Back	No Write Allocate	-	-	-

Table 4. Showing Cache Time Analysis for parameters based on S. No 2 [11]

Cache Time Analysis					
Cache Specification:	Cache Size = 8 KB	Associativity = 4	Words / Block = 16	Hit Rate = 0.9635	Miss Rate = 0.0365
Parameters:	% Writes = 30%	% Reads = 70%	% Dirty = 15%	Hit Time = 2	Miss Penalty = 45
Write Policies:	Write Back	Allocate on Write Miss	-	-	-

Table 5. Showing Cache Time Analysis for parameters based on S. No 3 [11]

Cache Time Analysis					
Cache Specification:	Cache Size = 32 KB	Associativity = 8	Words / Block = 32	Hit Rate = 0.9877	Miss Rate = 0.0123
Parameters:	% Writes = 30%	% Reads = 70%	% Dirty = 20%	Hit Time = 3	Miss Penalty = 40
Write Policies:	Write Through	No-Write Allocate	-	-	-

Table 6. Showing Cache Time Analysis for parameters based on S. No 4[11]

Cache Time Analysis					
Cache Specification:	Cache Size = 256 KB	Associativity = 8	Words / Block = 32	Hit Rate = 0.9972	Miss Rate = 0.0028
Parameters:	% Writes = 30%	% Reads = 70%	% Dirty = 25%	Hit Time = 4	Miss Penalty = 35

Write Policies:	Write Back	Allocate on Write Miss	-	-	-

Case-3 To visualize how data is accessed to and from the cache for various Replacement policies like LRU, FIFO

S. No.	Cache Size	# Sets	Replacement Policy	Task A for Multi-tasking
1.	16	4	LRU	2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,
2.	16	4	FIFO	2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,

Set#	16 Block, 4-way set-associative cache			
0	1618202224 (Compulsory Miss)	363840 (Compulsory Miss)	-	-
1	-	-	-	-
2	2468101214 (Compulsory Miss)	2628303234 (Compulsory Miss)	-	-
3	-	-	-	-

Fig. 7: Cache status based for LRU replacement policy [12]

Compulsory Misses:	**4**	**Total Cache Queries:**	**4**
Capacity Misses:	**0**	**Total Misses:**	**4**
Conflict Misses:	**0**	**Miss Rate:**	**100 %**
Cache Hits	**0**	**Hit Rate:**	**0 %**

Fig. 8: Cache query result for LRU replacement policy [12]

Cache Query Sequence Trace															
2468101214	1618202224	2628303234	363840												

Fig. 9: Cache Query Sequence Trace for LRU replacement policy [12]

Set#	32 Block, 8-way set-associative cache			
0	1618202224(Compulsory Miss)	363840 (Compulsory Miss)		
1	-	-		
2	2628303234 (Compulsory Miss)	-		
3	-	-		
4	-	-		
5	-	-		
6	2468101214 (Compulsory Miss)	-		
7	-	-		

Fig. 10: Cache status based for FIFO replacement policy

Compulsory Misses:	**4**	**Total Cache Queries:**	**4**
Capacity Misses:	**0**	**Total Misses:**	**4**
Conflict Misses:	**0**	**Miss Rate:**	**100 %**
Cache Hits	**0**	**Hit Rate:**	**0 %**

Fig. 11: Cache query result for a FIFO replacement policy

Cache Query Sequence Trace			
2468101214	1618202224	2628303234	363840

Fig.12: Cache Query Sequence Trace for FIFO replacement policy

4. Some of the key facts after analysis for increasing system performance are

i. For x unique block address, with cache block addresses of A length having Least recently used replacement policy and then there will be x/A **miss ratio**.

ii. **Cache block size** is a crucial parameter in designing a computer's cache system, smaller cache block size will result in handling more number of blocks and will increase the hit ratio and thus miss penalty can be decreased.

iii. If a number of sets corresponding to the cache are V and the main memory block number is J, then set number for the **first line** will be (J mod V) and the **last line** will be (J mod V) + (N-1) where N-way set associate cache is used.

iv. There is a total of **8 different colors** needed to sure that no two synonyms map to dissimilar set in the processor's cache.

v. For a Central processing unit having **memory mapped I/O** it is the responsibility of the operating system to ensure I/O protecting through system calls.

vi. Although **Unified Cache** has lower hit rate, but it has special capability of **Load Balancing** between data fetch and the instruction.

vii. In **Associative Cache** has more hardware cost, it uses parallel search and data can be placed anywhere inside it.

viii. The **locality of reference** can decrease if there are several operands involved.

ix. It is possible to put more than one word into a single cache block for the **spatial locality**.

x. Regarding **temporal locality**, the approach used is to access the data items that are being used in the past.

xi. During **mapping of the main memory with the cache,** if a cache has C sets (for 2C blocks and 2 blocks per set) than any block K in the main memory will be mapped to the (KmodC) of the cache.

xii. Keeping the **block size and capacity of a cache unchanged**, even if the associativity of a processor's cache is doubled, there will no effect on the width of the data bus from the processor to main memory.

xiii. There is certain pre-requisite for **inclusion between cache levels** (say L1 and L2), L1 must have a write-through policy, L2 must be greater than or at least of the same size of L1.

xiv. The **Data Structure** plays a crucial role during data transfer from the cache. For example, B+ trees are associated with memory and Binary Search Trees (BST) are associated with the disk. In general, if there is 20 node access used by the BST, B+ will use only 4 block reads as B+ trees have more structure compatibility while storing data.

5. Conclusion

In the present work, various cache characterization parameters are analyzed, and many key facts are discussed. Some of the factors like locality of reference (Temporal and Spatial), mapping from main memory to cache block size, inclusion between cache levels, the data structure used play a crucial role for increasing system's performance. These factors must always be taken into consideration while designing an algorithm. Future work includes the deep study of various other parameters for a better understanding of the memory hierarchy.

References

[1] Bagga, Sachin, et al. "RMI Approach to Cluster Based Cache Oblivious Peano Curves." Computational Intelligence & Communication Technology (CICT), 2016 Second International Conference on. IEEE, 2016.

[2] J. Emer, "Cache Optimizations," [Online]. Available: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-823-computer-system-architecture-fall-2005/lecture-notes/l08_caches_2.pdf. [Accessed 29 December 2017].

[3] P. I. Koren, "Biography," UMass Dept. of Electrical and Computer Engineering, [Online]. Available: http://www.ecs.umass.edu/ece/koren/. [Accessed 29 December 2017].

[4] Kaur, Gurpinder, Sachin Bagga, and Kulvinder Singh Mann. "Hadoop Approach to Cluster Based Cache Oblivious Peano Curves." Advance Computing Conference (IACC), 2017 IEEE 7th International. IEEE, 2017.

[5] Bagga, Sachin, Akshay Girdhar, and Munesh Chandra Trivedi. "SPMD based time sharing intelligent approach for image denoising." Journal of Intelligent & Fuzzy Systems 32.5 (2017): 3561-3573.

[6] Bagga, Sachin, et al. "RMI Approach to Cluster Based Image Decomposition for Filtering Techniques." Advances in Computer and Computational Sciences. Springer, Singapore, 2018. 387-399.

[7] Kaur, Manmeet, Akshay Girdhar, and Sachin Bagga. "Cluster Based Approach to Cache Oblivious Average Filter Using RMI." The International Symposium on Intelligent Systems Technologies and Applications. Springer International Publishing, 2016.

[8] M. Ghosh and S. Mukhopadhyaya, "Cache oblivious algorithm of average filtering in image processing," 2012 International Conference on Informatics, Electronics & Vision (ICIEV), Dhaka, 2012, pp. 149-154.

[9] C. Sree Kumar and Bhawani Shankar Pattnaik, "Miss rate analysis of cache oblivious matrix multiplication using sequential access recursive algorithm and normal multiplication algorithm," 2013 International Conference on Emerging Trends in Communication, Control, Signal Processing and Computing Applications (C2SPCA), Bangalore, 2013, pp. 1-6

[10] Geoff Gallo, Navin Vemuri,Eric Fallon, "Cache Address Structure 1," UMass Dept. of Electrical and Computer Engineering, [Online]. Available: http://www.ecs.umass.edu/ece/koren/architectur

e/Cache/default.htm. [Accessed 29 December 2017].

[11] Geoff Gallo, Navin Vemuri,Eric Fallon, "Cache Time Analysis 2," [Online]. Available: http://www.ecs.umass.edu/ece/koren/architecture/Cache/frame2.htm. [Accessed 29 December 2017].

[12] Geoff Gallo, Navin Vemuri,Eric Fallon, "Block Replacement Simulator 3," [Online]. Available: http://www.ecs.umass.edu/ece/koren/architecture/Cache/frame1.htm. [Accessed 29 December 2017].