Early Result from Adaptive Combination of LRU, LFU and FIFO to Improve Cache Server Performance in Telecommunication Network

Tanwir, Gamantyo Hendrantoro, Achmad Affandi
Department of Electrical Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
tanwir10@mhs.ee.its.ac.id gamantyo@ee.its.ac.id affandi@ee.its.ac.id

***Abstract*— telecommunications system network server is a multimedia storage medium, load server is storing data transmission can be reduced with an additional caches servers which store data while making it easier for clients to access informations. The more clients to access information causing increasing caches capacity is needed deletion of caches with using a combination of algorithm LRU, LFU and FIFO Queue method, in time of the initial data to be deleted (FIFO), the other algorithm will detect if such data has the most references (LFU) or LRU algorithm so that frequently accessed data to be stored is cached it will reduce delay time, Throughput and Loss Browsing.**

Keywords*— *Algorithms; LRU; LFU; FIFO; Cache Server

I. INTRODUCTION

The development of the telecommunications system network where the number of internet users primarily information technology today is increasing, this is caused by the increasing number of people to send information through the Internet, with a wide variety of information formats (text, image and video). The number of such users are generally not followed by the addition of appropriate bandwidth, often resulting in problems in the delivery of data.

Cache Server is a place to store data temporarily. This method is intended to improve the transfer of the data store that never accessed the cache, so that when the data is accessed form of the same data, then access can be done more quickly. Cache memory is located between the main memory registers and then processing the data is not directly refer to the main memory.

Several research on the cache server that functions to accelerate data access [1] on the computer where the cache stores information that has been accessed by a buffer, when a data is already stored in memory with the address. If the redial data, the cache monitor will check back in a cache location so as to accelerate the performance of memory and speed up access to data on the computer. By evaluating the performance of various system cache memory [2] [3] and [4] obtained the development of cache memory to an XOR circuit [5] [6].

Computer technologies evolve with the introduction of the multimedia processor with MMX. Processors with this capability can improve the multimedia experience that MMX is a forerunner of instruction SIMD (Single Instruction Multiple Data) that since it was developed several processors ranging Pentium, Pentium 2, Pentium 3, Pentium 4 processor development [7], then the Pentium dual-core, dual core2 and core i, with a capacity larger cache memory in the form of level 1 (L1), level 2 (L2) and level 3 (L3).

Special challenges in the development of cache memory is relatively smaller capacity of main memory but has a relatively higher rate than main memory. Until now, the cache memory is divided into three levels, namely L1, L2 and L3 [8]. Cache memory has the highest access speed and price of the most expensive [8]. Developing memory size ranging from 8 KB, 64 KB and 128 KB. Cache memory level 2 (L2) has a larger capacity ranging from 256 KB to 2 MB. However, L2 cache memory has a lower speed than the L1 cache memory. L2 cache memory is separated by the so-called external processor cache.

In this research analyzed the increase in Internet access causes the smaller bandwidth so that the cache server an increase in the amount of data that is causing the delay time so that the communication between client access servers are often disconnected.

II. DESIGNING FOR PERFORMANCE

A. Size Cache

One solution to the problem of determining the amount of cache proxy service requests from the client to adjust the amount of cache servers are implemented in the form of services as shown in Fig. 1.

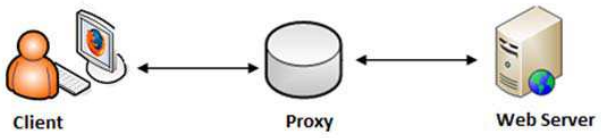

Fig. 1. Basic Cahe Proxy

1) *Server Cache:* First Client requested web objects to the proxy, a proxy check the data, whether the client requested web objects exist, continue to provide the requested object proxy client.

2) *Server Proxy:* The first object of a web client sends a request to the proxy and then check the data, if the client requested web objects exist, if the object does not have the proxy server forwards the request to the client. If the requested object is no further client server sends to the client. The mechanism of the proxy server performance [9] shown in Fig. 2.

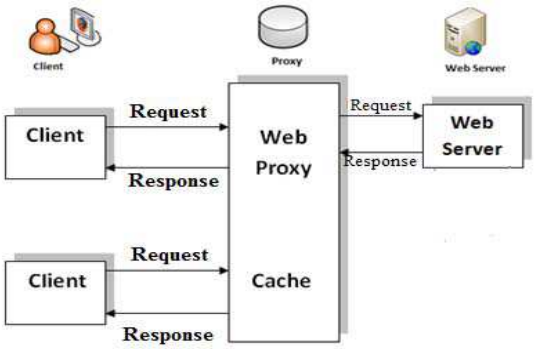

Fig. 2. Performance Mechanism Cahe Proxy

B. Mapping

Channel cache memory is less than the main memory block, it is necessary algorithms for mapping main memory blocks into the cache memory channel. The mapping algorithm is divided into three methods, namely direct mapping, associative mapping and Set Assosiative mapping.

1) Direct Mapping:

a) Each block in the main memory is mapped to a particular line in the cache. i = j modulo C where i is the line number in the cache that is used to put the main memory block j.

b) If M = 64 and C = 4, then the mapping between the line with the block to be as follows:

Line 0 can hold blocks 0, 4, 8, 12, ...
Line 1 can hold blocks 1, 5, 9, 13, ...
Line 2 can hold blocks 2, 6, 10, 14, ...
Line 3 can hold blocks 3, 7, 11, 15, ..

c) In this method, the address in the main memory is divided into 3 fields, namely:

- Tag identifier.
- Line number identifier
- Word identifier (offset)

d) Word identifier contains information about the location of the word in other addressable unit in a particular line in the cache.

e) Line identifier contains information about the number of physical (not logical) line in the cache.

f) Tag identifier is stored in the cache along with the block on the line:

- For each memory address made by the CPU, a certain line that stores a copy of the address specified, if the block where the location data is copied from the main memory to cache.
- Tag is on the line will be checked to see whether the block in question is on the line

2) Associative Mapping.

a) Allows block placed on any line that is not being used

b) Expected to overcome the major drawbacks Direct Mapping.

c) test each cache to find the desired block.:

- Checking every tag on line
- Very slow for large caches.

d) Line numbers become meaningless. Address of main memory is divided into two fields, namely tags and word offset.

e) Perform a search for all the tags to find the block.

f) The cache is divided into two parts:

- lines in the SRAM
- tag in associative memory

3) Set Associative Mapping.

a) A combination of Direct with Full Associative Mapping.

b) Ividing the cache into a number of sets each have a number line.

c) Each block can be placed in any line with the number the set: number = j modulo v

d) If a the set can accommodate x line, it is called a cache has a the set associative cache Xway.

e) Almost all of the cache that is used today by the organization 2 or 4-way set associative mapping [10].

III. Cache Memory Simulator

Cache memory is a temporary data storage location. Placement of cache memory is intended to reduce the gap between processor speed and main memory. This method is intended to improve the transfer of data to the data storage ever accessed in the memory cache. To improve the performance of cache memory, we need a change of algorithm Least Recently Used (LRU) replace the oldest data block in the cache memory and does not have a reference. Other algorithms Least Frequently Used (LFU) change the page that has the least reference and First In First Out (FIFO) replace the initial block of data entry.

The shape of the telecommunications measurements carried out as shown in Fig. 3. Flowchart that illustrates an information system data access services on time to request, in the form send requests implementation where the data will be checked on the proxy cache if there is no data will be given

information to proxy if there is then the request will be forwarded next web server request data will be stored back in the proxy cache in the form of requests that have been served. The length of the data can be stored depends on the algorithm used replacement. It is very influential on time connectivity.

The working principle of LRU replacement algorithm, LFU and FIFO queue using methods that at the time the initial data entry cache deletion detection (FIFO), with a combination of three algorithms so that initial data entry is not so removed for work LFU principle which states that data has many references will be maintained then the data is often used to make browsing is not removed and LRU algorithm so that reduces the delay time and throughput.

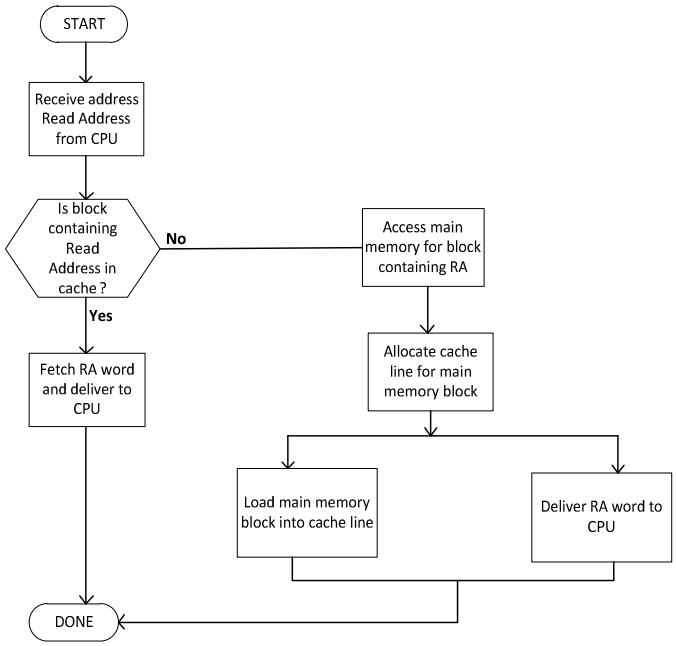

Fig. 3. Flowchart

Overall use of software performance benchmark tests to measure the microprocessors used in client server relationship as shown in Fig. 4.

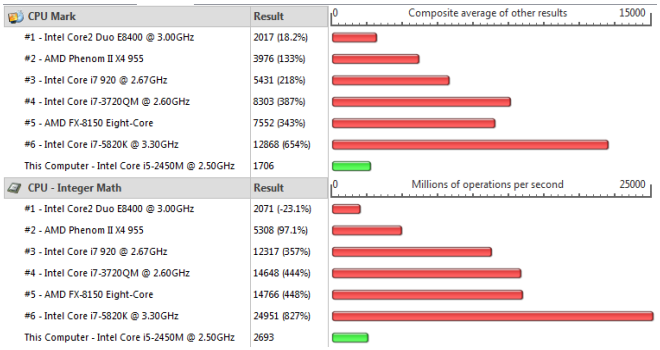

Fig. 4. Measurement Data Processor

The measurement results of data specifications Intel Core i5 2450M processor L1 D cache 32 Kbytes, 32 Kbytes of L1 I, 256 Kbytes L2 and L3 3 Mbytes. For the measurement of average Disk Speed Writing and Reading with the same data input browsing is google.com, Scribd, email and linkedin as shown in Fig. 5

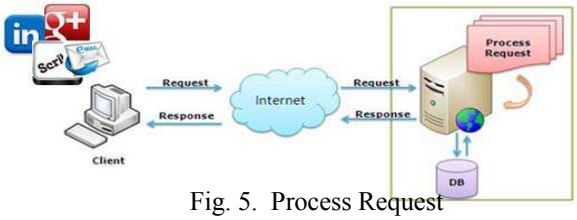

Fig. 5. Process Request

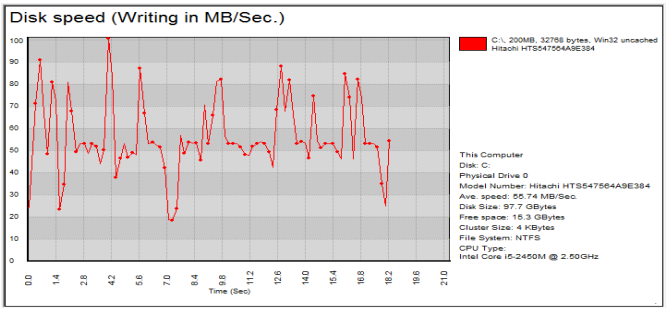

Fig. 6a. Disk Speed Writing

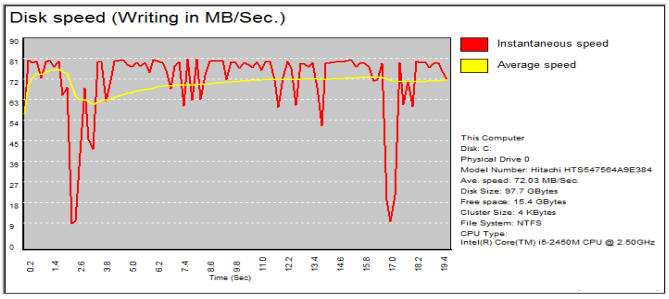

Fig. 6b. Disk Speed Writing

Performance on a client-server system with the present measurement results using a performance review software writing Benchmark obtained average speed of 55.74 MB / sec on Intel Core i5-2450M @ 2.5 GHz as shown in Fig. 6a, with a combination of LRU algorithm , LFU and FIFO obtained average speed Writing became 72.03 MB / sec as shown in Fig. 6b.

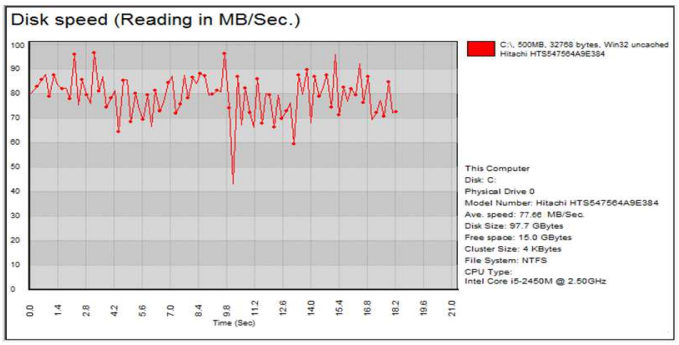

Fig. 7a. Disk Speed Reading

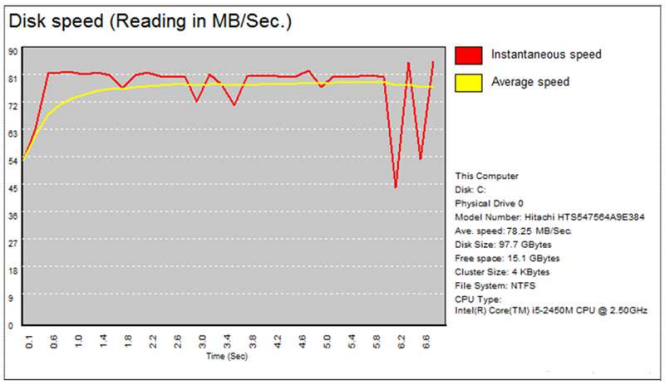

Fig. 7b. Disk Speed Reading

While the Fig. 7a acquired measuring disk Speed Reading Average of 77.66 MB/sec on Intel Core i5-2450M @ 2.5 GHz with a combination of algorithms LRU, LFU and FIFO obtained Average Speed Reading becomes 78.25 MB/ sec as shown in Fig. 7b.

IV. Conclusion

The more users on the network telecommunication system Client - Server lead to increased data cache server then there is delay time and bandwidth is smaller, with using combination algorithm LRU, LFU and FIFO so that at the time of deletion of caches done early detection of incoming data (FIFO) with regard the number of reference data used in browsing so that indirectly deleted (LFU), thereby reducing delay time, throughput and loss browsing.

References

[1] H. Sun, C. Liu, W. Xu, J. Zhao, N. Zheng, and T. Zhang, "Using Magnetic RAM to Build Low-Power and Soft Error-Resilient L1 Cache," IEEE Trans. Very Large Scale Integr. VLSI Syst., vol. 20, no. 1, pp. 19–28, Jan. 2012.

[2] M. Soryani, M. Sharifi, and M. H. Rezvani, "Performance Evaluation of Cache Memory Organizations in Embedded Systems," in Fourth International Conference on Information Technology, 2007. ITNG '07, 2007, pp. 1045–1050.

[3] S. Laha, J. H. Patel, and R. K. Iyer, "Accurate low-cost methods for performance evaluation of cache memory systems," IEEE Trans. Comput., vol. 37, no. 11, pp. 1325–1336, Nov. 1988.

[4] G. S. Sohi, "Cache memory organization to enhance the yield of high performance VLSI processors," IEEE Trans. Comput., vol. 38, no. 4, pp. 484–492, Apr. 1989.

[5] H. Vandierendonck, P. Manet, and J. Legat, "Application-Specific Reconfigurable XOR-Indexing to Eliminate Cache Conflict Misses," in Design, Automation and Test in Europe, 2006. DATE '06. Proceedings, 2006, vol. 1, pp. 1–6.

[6] A. K. Goksel, R. H. Krambeck, P. P. Thomas, M.-S. Tsay, C. Y. Chen, D. G. Clemons, F. D. LaRocca, and L.-P. Mai, "A content addressable memory management unit with on-chip data cache," IEEE J. Solid-State Circuits, vol. 24, no. 3, pp. 592–596, Jun. 1989.

[7] Z. Zhang, Z. Zhu, and X. Zhang, "Design and optimization of large size and low overhead off-chip caches," IEEE Trans. Comput., vol. 53, no. 7, pp. 843–855, Jul. 2004.

[8] Y. Niranjan, S. Tiwari, and R. Gupta, "Average memory access time reduction in multilevel cache of proxy server," in Advance Computing Conference (IACC), 2013 IEEE 3rd International, 2013, pp. 44–47.

[9] Y.-W. Horng, W.-J. Lin, and H. Mei, "Hybrid prefetching for WWW proxy servers," in 1998 International Conference on Parallel and Distributed Systems, 1998. Proceedings, 1998, pp. 541–548.

[10] H. R. Zarandi and S. G. Miremadi, "Hierarchical multiple associative mapping in cache memories," in Engineering of Computer-Based Systems, 2005. ECBS '05. 12th IEEE International Conference and Workshops on the, 2005, pp. 95–101