ISSN: 2321-7782 (Online)
Impact Factor: 6.047

Volume 4, Issue 7, July 2016

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

Cache Memory: An Analysis on Performance Issues

Dr. Vivek Chaplot
HEAD, Dept. of Computer Science
Bhupal Nobles' University
India

Abstract: ***Cache Memory is very fast memory placed between CPU & Main Memory. Cache memory plays a crucial role in deciding the performance of multi-core systems. In this paper, performance of cache memory is evaluated through following factors cache access time, miss rate and miss penalty. Processor speed is increasing at a very fast rate so a very high speed cache memory is needed for matching the speed of processor. The performance gap between processors and main memory continues to widen, increasingly aggressive implementations of cache memories are needed to bridge the gap. Cache systems are on-chip memory element used to store data. Cache serves as a buffer between a CPU and its main memory. Cache memory is used to synchronize the data transfer rate between CPU and main memory. As cache memory closer to the microprocessor, it is faster than the RAM and main memory. The advantage of storing data on cache, as compared to RAM, is that it has faster retrieval times, but it has disadvantage of on-chip energy consumption.***

Keywords: Cache memory, access, hit ratio, addresses, mapping.

I. INTRODUCTION

Cache Memory affects the execution time of a program. Cache memory gives data at a very fast rate for execution by acting as an interface between faster processor unit on one side and the slower memory unit on the other side. In Cache memory frequently used data or instructions are kept so that it can be accessed at a very fast rate improving overall performance of the computer. There are various level of cache exist in a computer system ,the cache at last level is being largest and slowest, cache at first level is fastest and smallest. Commonly in a processor first level cache (L1) reside in the processor and second level cache (L2) and third level cache (L3) are on separate chip. In multi-core processors, each processor has its own L1 cache. Last level cache is being shared by all the cores. Whenever CPU wants any data or instruction it sends address to the cache memory, address is searched in cache memory firstly, if data is found in cache it is given to the cpu. Finding a data in cache memory is called as cache HIT. If data is not found in cache then it is called as cache MISS, in this case address is searched in main memory for the data or instruction. After data is received from main memory, a block of data is transferred from main memory to cache memory so that all further request can be fulfilled from cache. Performance of cache memory is measured in HIT Ratio. Hit Ratio is No. of Hits / No. of Hits + No. of Miss. The hit ratio approaching towards 1 is considered as best .

Cache misses are of three types: i) Misses which take place when a memory location is accessed for the first time ii) Misses which occur due to insufficient space when two blocks are mapped on the same location and iii) Misses takes place due to small space as cache . Cache miss rate and miss penalty are the two major factors which effect cache performance. Time required to handle a cache miss is called cache penalty. There are various methods to reduce cache miss rates which include victim cache which is a location for temporary storage of cache line which is abolished from cache, Column associative caches reduces miss rate of direct mapped caches, pre-fetching of data.

We can improve cache performance by predicting future access of data or instructions.

Cache memory is based on principal of Locality of References. By this we mean when a program executes on a computer, related storage locations being frequently accessed. Locality is of two type in time (temporal locality) and in space (spatial locality).

Temporal Locality refers to the reuse of specific data and/or resources within relatively small time durations. Spatial Locality refers to the use of data elements within relatively close storage locations.

The paper discusses cache mapping techniques, how to improve cache performance.

II. CACHE MAPPING TECHNOLOGY

There are a three Cache mapping techniques – Direct mapping – Associative mapping – Set associative – mapping.

Direct mapping: In direct mapping both RAM & cache is used to store data. An address space is divided into two parts index part and tag part. The cache is used to store the tag field whereas Index is stored in the main memory. Firstly index part is matched, if indexed part is matched then tag part is matched, if tag part also matches then we have Cache HIT. How many bits should be in tag part it is decided by size of cache memory. There is one problem in direct mapping, some time index matches but tag do not match. But this problem occurs in those address which are very far apart. Direct mapping`s performance is directly proportional to the Hit ratio. This technique is not flexible.

Associative mapping: In this type of mapping the associative memory is used to store content and addresses both of the memory word. This enables the placement of the any word at any place in the cache memory. When the processor wants an address, all tag fields in the cache as checked to determine if the data is already in the cache. Each tag line requires circuitry to compare the desired address with the tag field. All tag fields are checked in parallel. It is considered to be the fastest and the most flexible mapping form.

Set-associative mapping: Set associative mapping is a mixture of direct and associative mapping . The cache lines are grouped into sets. This form of mapping is a modified form of the direct mapping where the disadvantage of direct mapping is removed. Set-associative mapping allows that each word that is present in the cache can have two or more words in the main memory for the same index address.

Improving Cache Performance:

We can improve cache performance by following:

1. Reduce the miss rate 2. Reduce the miss penalty 3. Reduce the time to hit in the cache.

Easiest way to reduce miss rate is to increase cache block size. Higher associatively can improve cache miss rates. There are different replacement algorithms which are used to reduce miss rates and make cache performance better. One way of reducing the gap between CPU cycle and memory latency is to use a multi-level cache. Misses of first level can be handled by introducing second level cache. Different methods are used to make cache performance better. One of them is to make different banks of cache that can be accessed at different latencies.

III. CONCLUSION

In this paper, we discussed cache memory & how to improve the performance of cache memory. Each technique is associated with its design constraints, advantages and limitations. Different techniques could be further enriched. The rate of conflict misses can reduced by using larger block size but larger cache is needed. Using larger block size may increase miss penalty, reduced hit time and power consumption. Larger cache produces slow access time and high cost. Higher associatively produce fast access time but they have low cycle time. Victim cache reduces miss rate at a high cost comparing to Cache miss look aside. In all we can say that there is a room for improving performance of cache memory to a very large extent.

References

1. J. S. Yadav, M. Yadav, and A. Jain, "CACHE MEMORY OPTIMIZATION," International Conferences of Scientific Research and Education, vol. 1, no. 6, pp. 1–7, 2013.
2. S. Paper, R. Sawant, B. H. Ramaprasad, S. Govindwar, and N. Mothe, "Memory Hierarchies-Basic Design and Optimization Techniques Survey on Memory Hierarchies – Basic Design and Cache Optimization Techniques," 2010.
3. J. R. Srinivasan, "Improving cache utilisation," Phd Diss., University Of Cambridge, no. 800, 2011.
4. N. P. Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," ACM SIGARCH Comput. Archit. News, vol. 18, pp. 364–373, May 1990.
5. A. Agarwal and S. D. Pudar, "Column-associative Caches: A Technique For Reducing The Miss Rate Of Direct-mapped Caches," Proc. 20th Annu. Int. Symp. Comput. Archit., pp. 179–190. 1993.
6. S. Ramaswamy, S. Yalamanchili, and C. Architectures, "Improving Cache Efficiency via Resizing + Remapping," Computer Design, 2007. ICCD 2007, 25th International Conference on. IEEE, 2007
7. M. Kampe, P. Stenstrom, and M. Dubois, "Self-correcting LRU replacement policies," Proc. first Conf. Comput. Front. Comput. Front. - CF'04, p. 181, 2004. [25] M. Kharbutli and Y. S. Y. Solihin, "Counter-Based Cache Replacement and Bypassing Algorithms," IEEE Trans. Comput., vol. 57, no. 4, 2008.

