International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 7, July 2014)

Analysis of Computer Architecture & Emergence of Cache Memory Miss Penalty

Kaitepalli Satya Narayana
Asst Prof , Computer Science Engg
Sphoorty Engineering College
Hyderabad, India

Kuchimanchi JayaSri
Asst Prof , Computer Science Engg
Nalla Narasimha Reddy
Educational Group of Institutions
Hyderabad, India

S. Guru Jyothi
Asst Prof , Computer Science Engg
Sphoorty Engineering College
Hyderabad, India

***Abstract*— Temporary computer storage used for quick retrieval of data in order to increase processing speed and data can be stored in a reserved area of RAM a special cache chip that provides faster access than RAM. Frequently accessed data in a rapidly accessible place the computer can respond quickly to requests for those data without having to perform time consuming search of RAM. As CPU clock rates have increased at a much faster rate than DRAM the relative cost of miss penalties has actually increased over time, when a cache miss occurs no need to wait for the complete block to fill before forwarding requested word to CPU. This work presents cache mapping in architecture of computers and how the cache works in several applications to avoid the cache miss penalty are used to improve performance of future architecture in real time. We present the issues in miss penalty cache and performance of cache is easily scalable to meet business requirements.**

***Index Terms*— Cache Memory, Windows, Java, Cloud Computing, Miss Penalty.**

INTRODUCTION I

Computer architecture is a system visible to a programmer that have a direct impact on the logical execution of a program refers to the operational units and their interconnection that realize the specification for example architecture attributes include instruction set the number of bit to represent various data types input/output technique and memory. Many computer manufacturers offer a family of computer model all with the same architecture but with differences in organization consequently the different models in the family have different price and performance characteristics. Contemporary computer contain millions of elementary electronic components recognize the hierarchical nature of most complex system is a set of interrelated subsystem each of the later in turn. The designer need only deal with a particular level of the system at a time at each level the system consists of a set of components and their interrelationships. Computer is processing data on the fly must temporarily store at least those pieces of data that are being worked on at any given moment files of data are stored on the computer for subsequent retrieval and update. The computer must be able to move data between itself and operating environment consists of devices that serve as either sources or destinations of data when data are moved over longer distances to from a remote device the process is known as data communication.

Processors are generally able to perform operations on operands faster than the access time of large capacity main memory which can operate at speeds comparable with the operation of the processor exists it is not economical to provide all the main memory with very high speed memory. Cache memory is similar to virtual memory in that some active portion of a low speed memory is stored in duplicate in a higher speed cache memory when a request is generated the request is first presented to the cache memory and if the cache cannot respond the request is then presented to main memory. Cache miss to be a reference to item that is not resident in cache but is resident in main memory, cache memories is page fault which is defined to be a reference to a page in virtual memory that is not resident in main memory.

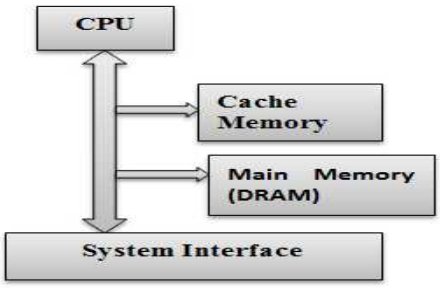

Figure 1 Model of Cache Memory

The basic model of cache every time the CPU performs a read or write the cache may intercept the bus transaction allowing the cache to decrease the response time of the system, cache contains the information requested the transaction is said to be a cache hit or does not contain information is missed. Cache is copy of a small piece main memory important that the cache always reflects what is in main memory, cache is watching the address lines for transaction accessing memory it contains within itself and when a cache takes the information from the data lines the cache is said to have snarf the data allows the cache to be updated and maintain consistency.

SECTION II

2. System cache mode is designed to improve the performance of windows the memory that Microsoft to file caching the

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 7, July 2014)

memory that windows allocates to applications system cache resources are partitioned during startup and do not change. System cache mode is designed to improve the performance of windows server by increasing the system file cache size webservers and other server based file sharing programs generally perform better when information is read from the system cache instead of read repeatedly from hard disk file server performance improves.

Cache is organized into two terms one is cache page, second is cache line main memory is equal classifies the cache pages, the page size is dependent on cache here the cache page is broken into cache line size dependent on the cache of cache design. The organization of cache is classified into cache page and cache line using the mapping technique.

2.1. Associative Cache: organization scheme allows any line in main memory to be stored at any location in the cache associative cache does not use cache pages it uses only lines. In associative cache main memory and cache memory divided into lines of equal size which shows that Line1 of main memory is stored in Line 0 of cache, not possible Line 1 could have been stored anywhere within the cache.

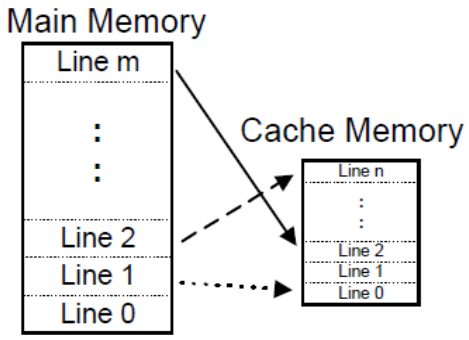

Figure 2 Associative Cache

Associative scheme provides the best performance because any memory location can be stored at any cache location, to meet the timing requirements the current address must be compared with all the addresses present in the TRAM due to this very large number of comparators that increase the complexity and cost of implementing large cache therefore cache is usually used for small cache which is less than 4k.

2.2. Direct Map: this cache is one way set of associative that main memory is divided into cache pages. The size of each page is equal to the size of the cache. Associative cache the direct map cache may only store a specific line of memory within the same line of cache see the example Line 1 of any page in memory must be stored in Line 0 of cache memory if Line 0 of page 0 I stored within the cache and Line 1 of page 1 is requested such way Line 0 of page 0 will be replaced with Line 0 of page 1. Direct map cache is the least complex from all other cache scheme direct map cache only requires that the current requested address be compared with only one cache address.

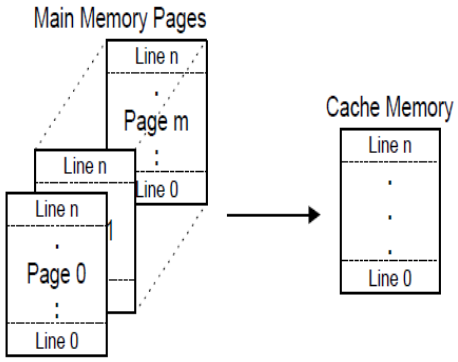

Figure 3 Direct Map Cache

2.3. Set Associative Cache: this cache scheme is a combination of fully associative and direct mapped caching scheme it works by classifying the cache SRAM into equal cache. The cache page size is equal to the size of the cache way. Each cache is treated like a small direct mapped cache. Set associative cache may stores two lines of memory at any time helps to reduce the number of times the cache line data is overwrite.

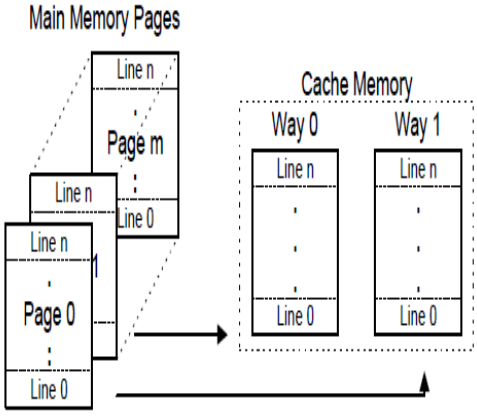

Figure 4 Set Associative Cache

SECTION III

3.1. Cache in Windows: Cache works by converting normal file input output requests for virtual memory mapped files, manager interfaces to applications in several different ways standard interface is the copy interface it is entirely transparent to applications. Copy interface functions for read hit requests an application read request calls that appropriate file system driver in windows where the request is immediately routed to the cache manager. The cache manager maps each open file into the virtual memory reserved for the cache is performed on 256 KB sections of a file at a time. The read responds the cache manager locates the block of data specified and copies it into an application provided data buffer satisfies the original file request and application thread resumes its normal

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 7, July 2014)

processing at the same point the file data requested resides in two places concurrently which is an inefficient use of computer memory. To plug the file cache function into existing applications transparently without requiring extensive modification to allow the application to accept a pointer to file data resident in the system cache.

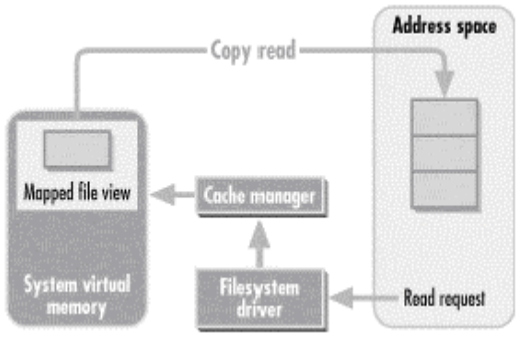

Figure 5 Cache in window

File write requests across the copy interface are subject to deferred write back caching copy interface copies data from the applications file buffers into virtual storage locations associated with the cache. Dirty pages in the cache are backed with real memory.

3.2. Cache in Java: A cache is an area of local memory that holds a copy of frequently accessed data that is expensive to get of compute such data includes a result of a query to a database file or a report. Caching may provide performance improvement for a java application often in orders of magnitude consider an application that shows a 50 line system status report displayed on a web page after the user logs into the system. For each line it sends a complex SQL query to a database to execute each query and to fetech results over the network takes 100 milliseconds on average. The total average time to collect all data for the page is about 5 seconds same result get from cache takes 5 microseconds on a modern 2GHz CPU. A high hit/miss ratio is calculated using hit and miss counters accumulated over a period of time that a cache performs well and also the low hit/miss ratio mean that a cache is too small to capture temporal locality of data access.

3.3. Cache in Clouds: A cache cloud is a group of edge caches from an edge network that cooperate among themselves to efficiently deliver dynamic web content to improve the performance of edge cache networks. When a cache miss the rate it tries to retrieve the document from another cache within the cache cloud then it contact the server to inform and the cache in cloud share the cost of document updates in the sense that the server needs to send a document update message to only one cache in a cache cloud then it distributes to other cache are currently holding the document, cache edge collaborate with each other to optimally utilize their collective resources by adopting smart strategies for document lookups updates placements and replacements.

Cache cloud with N edge cache each of this cache maintains lookup information about a set of documents, the assignment of documents to beacon points can vary over time the load balance is maintained when the load patterns change.

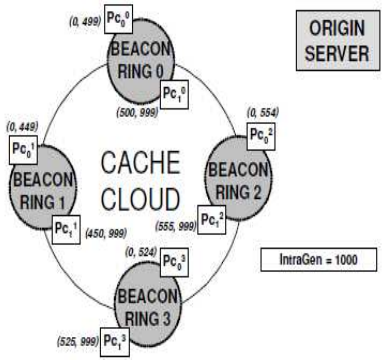

Figure 6 Cache cloud with N Edge

Figure the edge cache of cloud organized into substructures called beacon rings contains one or more beacon points with where each beacon ring has 2 beacon points in a particular beacon ring are collectively responsible for maintaining the lookup information of a unique set of documents. A cache cloud has K beacon rings numbered from 0 to K-1 and a document Dc is mapped to the beacon ring j assigned to serve as the beacon point of Dc. The assignment of the beacon point within a beacon ring is done dynamically and may change over time for maintaining balance of loads among the beacon points and rings.

SECTION IV

A cache is a pool of entries contains data for storing backup each entry as a tag specifies the identity of the data in the pool when a cache client needs to access a data presumed to exist in the backing store checks the cache if an entry can match with tag of desired data the data is used instead. In other situation when the cache is consulted and found not to contain a data with the desired tag has become known as cache miss, before unmatched cache data fetched from the backing store during miss handing is usually copied into the cache it was ready for the next access. When a system writes a data to cache it must be at some point write that data to backing store as well this process is controlled by the write policy here we have two options one write through is done synchronously both to the cache and to the backing store and write back is initially writing is done only to the cache.

Write back cache is complex to implement it needs to track which of its locations have been written over and mark them as dirty for storing back, data in these locations are written back to the backing store only when they are evicted from the cache is a lazy write cache. A read miss in a write back cache requires a block to be replaced by another will often require two memory accesses to service one to write the replaced data from the cache back to the store. Write through cache tend to be the most common because it provides a good balance of

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 7, July 2014)

performance and reliability, data can get into the cache by an application either reading or writing data.

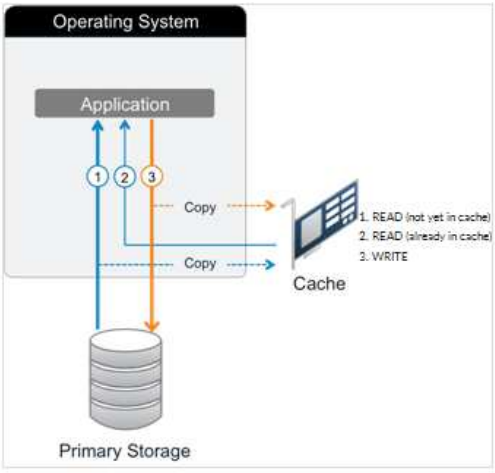

Figure 7 Write Through Cache

Write cache except that the cache cannot receive blocks of data from a write operation initially be placed into the cache but if a write causes an update to a block already residing in the cache then the cache block is updated. It is used when the application is writing a lot of data accessed shortly after the write occurs. Write back cache stores both read and write directly in the cache so it can significantly speed up both reading and writing operations. This approach requires handling a number of error conditions to protect against data corruption if the cache and the primary storage get out of synchronous.

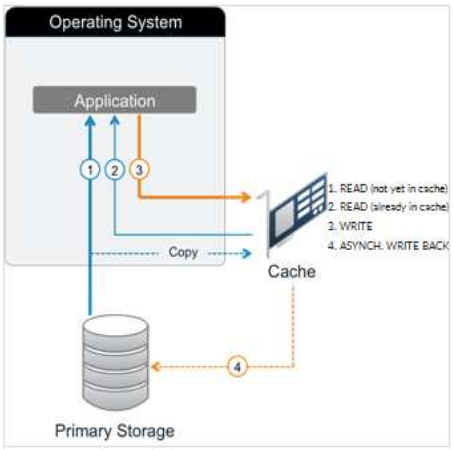

Figure 8 Write Back Cache

SECTION V

5. Problem Definition: Cache is measured in the frequency of cache hit where a lookup is resolved by the content stored in cache and frequency of cache miss where a lookup cannot be resolved by the content stored in the cache

The effectiveness of caching is commonly measured in the frequency of cache hit, where a lookup is resolved by the content stored in cache; and the frequency of cache miss, where a lookup cannot be resolved by the content stored in the cache.

Cache access time can be defined as T = P (cache hit)*(cost of hit) + P (cache miss)*(cost of miss)

Consider cache has a hit rate of 99%, and the access time is 2 clock cycles; a cache has a miss rate of 1%, and the memory access time is 4 clock cycles. The effective access time is the following:

T = 99%*2 + 1%*4 = 1.98 + .04 = 2.02 (clock cycles)

Therefore, with caching, 10 MB of cache effectively provides an illusion of 4 GB of memory storage running at the speed of hardware cache.

Compulsory misses occur because data are brought into the cache for the first time for example running a program for the first time since booting the machine. A program may require a large hash table that exceeds the cache capacity, such that no caching policy can effectively improve the performance of the program. These misses are not compulsory or capacity misses. Since a cache entry can be potentially assigned to multiple pieces of data, should two pieces of data be active, each will preempt the other from the cache on reference, causing cache misses or the policy to choose which cache entry to replace when the cache is full.

Average memory access time is a useful measure to evaluate the performance of a memory cache miss rates and miss penalty is different for each case with different instructions and data accesses. To calculate CPU time as single quantity yields a useful CPU time formula.

$$\text{CPU time} = \text{IC} \times \left(\text{CPI}_{\text{execution}} + \frac{\text{Memory access}}{\text{Instruction}} \times \text{Miss rate} \times \text{Miss penalty} \right) \times \text{Clock Cycle Time}$$

The increasing speed gap between CPU and main memory has made the performance of the memory system increasingly important and some organizations characterize the effort of the system architects in reducing average memory access time.

5.2. Cache Algorithms:

52.1. Least Recently Used: To implement Least Recently Used necessary to maintain a linked list of all pages in memory with the most recently used page at the front and the least recently used page at the rear. Consider A, B, C each one as a point of data in our cache and two empty spots we access D and E added to the cache as well filling it up. Suppose we access A again. A is already in the cache so the cache does not change however this access counts as a use making A the most recently used. Now if we were to access F we would have to throw something out to make room for F. at this point B has been used least recently so we throw it out and replace it with

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 7, July 2014)

F. now to access B again it would be exactly as the first time we accessed it.

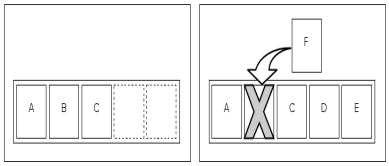

Figure 9 Example of Least Recently Used Cache.

Input will be a series of data sets one per line each data set will consist of an integer N and a string of two or more characters, N represents the size of the cache for the data set (1<N<26) string of characters consists of uppercase letters and exclamation marks, an uppercase letter represents an access to that piece of data. For example the sequence ABC, DEAF, D means to access A,B, and C print the contents of the cache access D,E, D and F then print the contents of the cache then access B and again print the contents of the cache.
Output of the each data set should be line Simulation S where S is 1 for the first data set 2 for second data set then for each exclamation mark in the data set should output the contents of the cache on one line as a sequence of characters representing the pieces of data currently in the cache. The characters should be sorted in order from least recently used to most recently used with least recently occurring first.

5.2.2. First In First out: Elements come in same order as they come in when put a call is made for a new element and assuming that the max limit is reached for the memory store the element that was placed first in the store is the candidate for first out. It takes a random sample of the elements and evicts the smallest using a sample size of 15 elements empirical testing shows that an element in the lowest quartile is evicted 99% of the time.

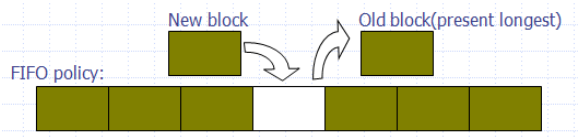

Figure 10 example of First In First Out

First In First Out strategy just requires a queue Q to store references to the pages in the cache, pages are queued in Q simply performs a dequeue operation on Q to determine which page to evict. This policy requires additional work per page replacement.

CONCLUSION VI

This work presents the cache memory organization as in Window, Java, Cloud computing to avoid the miss penalty, specific approach of miss penalty cache formula is characterized, improves the quality of the cache memory in many architecture. Our analysis shows the cache algorithms and mapping in computer organization in real time to achieve a high rate of accuracy in the case of storage. Future work concentrates on study of Miss Penalty cache.

REFERENCES

[1]. D. A. Menasce. Scaling Web Sites Through Caching. *IEEE Internet Computing*, July/August 2003.
[2]. L. Ramaswamy and L. Liu. An Expiration Age-Based Document Placement Scheme for Cooperative Web Caching. *IEEE-TKDE*, May 2004.
[3]. R. Kirner and P. Puschner, (2008) "Obstacles in worst-case execution time analysis", Proceedings of 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing, pp.333-339.
[4]. A.J. Smith, (2009) "Line (block) size choice for CPU cache memories", IEEE transaction on Computers, vol. 100, no. 9, pp. 1063-1075.
[5]. K. Chakraborty, P.M. Wells, and G.S. Sohi, (2007) "A case for overprovisioned multi-core system: Energy efficient processing of multi-threaded programs", Technical Report of CS-TR-2007-2607, University of Wisconsin-Madison
[6]. S. Laha, J.H. Patel and R.K. Iyer, (2002), "Accurate low-cost methods for performance evaluation of cache memory systems", IEEE Transactions on Computers, vol. 11, pp. 1325-1336.
[7]. J.D. Gee, M.D. Hill, D.N. Pnevmatikatos, and A.J. Smith, (2002), "Cache performance of SPEC92 benchmark suite", Proceedings of IEEE Micro, vol. 13, pp. 17-27.
[8]. V.V. Srinivas and N. Ramasubramanian, (2011), "Understanding the performance of multi-core platforms", International Conference on Communications, Network and Computing, CCIS-142, pp. 22-26.

About the authors:

Kaitepalli Satya Narayana M.Tech Computer Science Engineering having seven years of experience, working as Asst Prof at Sphoorty Engineering College. His research areas include Computer Organization & Architecture.

Kuchimanchi JayaSri M.Tech Computer Science Engineering having six years of experience, working as Asst Prof at Nalla Narasimha Reddy Educational Group of Institutions. Her research areas include Computer Organization & Architecture.

S. Guru Jyothi M.Tech Computer Science Engineering having five years of experience, working as Asst Prof at Sphoorty Engineering College. Her research areas include Computer Organization & Architecture.