Brief Overview of Cache Memory

Technical Report · April 2020

DOI: 10.13140/RG.2.2.22359.21921

Ameer Khan
Gift University

Brief Overview of Cache Memory

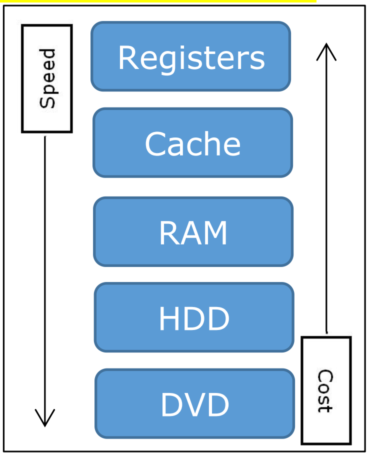
Ameer Khan
Ameer Khan Research & Development Center, 52250, Pakistan
ceo@akrndc.com

Introduction

Cache memory is one of the fastest memories inside a computer which acts as a buffer or mediator between CPU and Memory (RAM). When CPU requires some data element it goes to Cache and it that data element is present in cache, it fetches it; otherwise, cache controller requests the data from memory. Cache contains most frequently accessed data locations. While giving high speed of data access, cache is equally expensive as compared to other memories in a machine. Major purpose of a Cache is to reduce the memory access time because going to primary memory costs a lot more time compared to cache. With the development of high speed processors, memory access has been a bottleneck for the throughput of computational machines for decades. Multiple advances have been carried out to improve the throughput of computers, one of which was the introduction of cache memory. There are two main parts of the cache one of which is Directory which stores the addresses of the lines and the other one is Data line which holds the data that is stored in the cache memory which is further addressed by the directory.[1]

Memory hierarchy

A computer has different types of memories which serve different purpose depending upon their speed and cost. Some of these memories are volatile which means that they lose their states when power is turned off, while others are non volatile which retain their states even when the power is turned off. However, main purpose of all these memories is to store data and provide it to the processing unit when required. To reduce the latency of data transfer between memories and processing units, multiple strategies and techniques have been adopted over the decades. Major classification of memory includes, primary memory and secondary memory. Primary Memory includes the internal memories of a CPU which usually regards to registers, cache and RAM, whereas, secondary memory includes hard disks or compact disks etc.



Cache controller

Cache controller handles the data requests and controls data transfer between Cache and Processor & Cache and Memory. When processor requests a data element, cache controller checks for that element in Cache, and provides it to processor if present. In case, the required data element is not present in cache, the cache controller requests that data from memory. The read and write requests to memory are handled by the memory controller.

Cache controllers handles the request by dividing it into three parts tag, set index and data index. Set index is used to locate the corresponding line of the cache memory. If the valid bit represents that the line is active, tag is compared. In case of success in both these cases, the element is fetched and it is considered as Cache Hit, otherwise, it is a Cache Miss.

Cache Hit

If the data requested by the processor is present in the cache, it is accessed and provided to processor by cache controller and this is regarded as Cache Hit. Throughput of a system largely depends upon the frequency of cache hits because in this case, Processor has to face a very short latency.[2]

Cache Miss

If the data requested by the processor is not present in the cache, it is then requested from the memory and brought into the cache to make it available for the processor and this is regarded as Cache Miss. In this case, the data request is delayed by a reasonable amount of time and the processor has to wait for the data to arrive from the memory which is a comparatively time taking process.[2]

Miss Penalty

In case of a Cache Miss, the data is brought from the memory into cache to make it available for the processor, which is a time taking process. The time taken to bring the data to cache from memory, in case of cache miss, is regarded as miss penalty.[2]

Time to Hit

This is the time taken for the processing of a data request in case of a cache hit.

Levels of Cache

There are mainly three levels of cache (L1, L2, L3) which are categorized based on their speed and capacity. Going from L1 to L3, memory access time and storage capacity increases.[3]

A) L1:

This is the fastest cache and is placed close to or alongside of the processor to make data access faster. Level 1 cache is separate for all processors in multiprocessors machines and this is where requested data is checked first. Usually its size is up to 256KB, however, in some processors like Xeon it can be up to 1 MB. Instruction and Data is separate in this cache. However, this separation depends upon the architecture of the cache design.

B) L2:

This is slower than the L1 cache and greater in size. Its size is up to 8MB. Level 2 cache keeps the data that is expected to be accessed by the processor in coming clocks. Level 2 cache is also separate for all cores.

C) L3:

This is the slowest cache and greatest in size as compared to other cache memories. Level 3 cache is up to 50MB.

Data writing methods

There are two main techniques for data writing:

A) Write-Back:

In write-back a value is updated in cache but is not simultaneously updated into memory. Update occurs when update bit is set to 1.[4]

B) Write-Through:

In write-through a value is updated in cache and memory simultaneously. So, all the writes go to memory as well, which makes the write functions slower.[5]

Locality

A) Spatial:

Spatial locality means that the data elements to be accessed are placed close to each other in space.[6]

B)Temporal:

Temporal Locality means that the data elements are accessed frequently in time.[7]

Single Core & Multi Core Machines

Cache memory in **Single Core** is quite simple. It is used to access the data required by the CPU. If found in cache, the data is provided promptly to the CPU and in case it is not present in the cache it is fetched and loaded from the memory into cache and then provided to CPU.

In **Multi Core** machines, the working of cache is relatively complicated because each core has its own cache but uses the same main memory. So, when something is updated by a processor, it is updated in the cache of that processor making it difficult to keep consistency and coherency between all the cache memories of all cores and main memory.

Coherency:

In multi-core systems, all cores have their own cache and it is expected of the cache memories to work so smoothly that the data provided to a processor is correct, at a given point in time. When one core updates a value in its cache, all the other copies of the same value should become invalidated and should not be provided to any processor for any operation unless the updated value is propagated to all the locations. This phenomenon is called cache coherency. It can be explained in simple words as, All the cores see the same data or they are all provided the correct data at all times.[9]

Consistency:

Cache to memory consistency refers to the fact that the data copies in cache and memory are consistent. That includes values in data and ordered arrangement of instructions.[9]

Cache Mappings

A) Fully Associative:

In fully associative cache, a new cache line can be placed anywhere in the cache. It has comparators on all the elements of the directory. It is slower than other types of mappings but it is highly versatile.[8]

B) Directly Mapped:

In directly mapped cache, a cache line has a unique address where it is placed in cache. Each block in memory is mapped to only one line of the cache, so we don't need to check in other lines. It has one comparator that is used for comparison which makes it faster, however, less versatile.[1]

C) n-Way set Associative:

These are intermediate schemes between Fully Associative and Directly Mapped Cache. In these schemes a new cache line is to be placed in any of 'N' cache lines. In 4-Way Set Associative Cache, for example one line can be mapped to 4 different locations of the cache. This is relatively faster and relatively versatile when compared to above mentioned mapping schemes.[1]

Unified vs Split Cache

A) Unified Cache:

In unified cache, code and data are located in the same cache and the portion of cache taken by both code and data can vary accordingly to the situation. So, all the fetch or load requests of both code and data come to the same cache. In unified cache we only need to design and handle one cache.[10]

B) Split Cache:

In split cache, code and data are placed separately on two different cache portions. In this cache, the size of code and data portion is not flexible and can not be changed according to the situation. In this cache, all the load requests of data come to data cache portion and all the fetch requests of the code come to code cache portion. It is very effective pipe-lining. This is also known as I&D cache.[10]

Write Buffer for Cache:

In ARM (advanced RISC machines), a write buffer is used between cache and memory to improve the performance of memory write requests. A first in first out (FIFO) stack is used in this write buffer. Data is transferred to write buffer at the speed of the clock of the system.[11]

Victim Cache:

Victim cache holds the recently replaced elements from the cache. It is fully associative and used to reduce miss rates. In case of a cache miss, victim cache is checked before going to the memory for a data element. If it is found in victim cache, that block of victim cache is swapped with the main cache.[12]

Measuring Cache Performance

AMAT (Average Memory Access Time) is a measure that is used to test performance of a memory. It is calculated using following formula:

$$AMAT = Hit\ Time + Miss\ Rate \times Miss\ Penalty$$

Performance of a cache can be improved by following things:

1. Reducing Miss Rate
2. Reducing Miss Penalty
3. Reducing Time to Hit

At the start, there is compulsory miss because it is the first access by the processor to the memory. Due to limited size of the cache, data that was used in the past and not required now is replaced with the data that is required now and in that case cache miss occurs. There can be misses due to collision too. Reducing the number of misses will increase the number of Hits which eventually increases the throughput by increasing memory access time significantly.[13]

References

[1] Hill, M.D., 1987. Aspects of cache memory and instruction buffer performance (No. UCB/CSD-87-381). CALIFORNIA UNIV BERKELEY DEPT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCES.

[2] Kowarschik, M. and Weiß, C., 2003. An overview of cache optimization techniques and cache-aware numerical algorithms. In Algorithms for memory hierarchies (pp. 213-232). Springer, Berlin, Heidelberg.

[3] Knotts, B.W., NCR Corp, 1997. Coherent copyback protocol for multi-level cache memory systems. U.S. Patent 5,671,391.

[4] Steps, S.C., HP Inc, 1989. Write-back cache system using concurrent address transfers to setup requested address in main memory before dirty miss signal from cache. U.S. Patent 4,858,111.

[5] Martinez Jr, M.W., Bluhm, M., Byrne, J.S., Courtright, D.A., Duschatko, D.E., Garibay Jr, R.A. and Herubin, M.R., Cyrix Corp, 1996. Coherency for write-back cache in a system designed for write-through cache including write-back latency control. U.S. Patent 5,524,234.

[6] Kumar, S. and Wilkerson, C., 1998, July. Exploiting spatial locality in data caches using spatial footprints. In Proceedings. 25th Annual International Symposium on Computer Architecture (Cat. No. 98CB36235) (pp. 357-368) IEEE.

[7] Song, Y. and Li, Z., 1999. New tiling techniques to improve cache tempora locality. ACM SIGPLAN Notices, 34(5), pp.215-228.

[8] Singh, J.P., Stone, H.S. and Thiebaut, D.F., 1992. A model of workloads and its use in miss-rate prediction for fully associative caches. IEEE transactions on computers, (7), pp.811-825.

[9] Petrot, F., Greiner, A. and Gomez, P., 2006, August. On cache coherency and memory consistency issues in NoC based shared memory multiprocessor SoC architectures. In 9th EUROMICRO Conference on Digital System Design (DSD'06) (pp. 53-60). IEEE.

[10] Coutinho, L.M., Mendes, J.L.D. and Martins, C.A., 2006, October. Mscsim-multilevel and split cache simulator. In Proceedings. Frontiers in Education. 36th Annual Conference (pp. 7-12). IEEE.

[11] Miyake, J., Panasonic Corp, 1996. Cache memory with a write buffer indicating way selection. U.S. Patent 5,564,034.

[12] Peled, G. and Spillinger, I., Intel Corp, 2001. Trace victim cache. U.S. Patent 6,216,206.

[13] Sun, X.H. and Wang, D., 2012. APC: a performance metric of memory systems. ACM SIGMETRICS Performance Evaluation Review, 40(2), pp.125-130.