Original Research Paper Engineering

Analysis of Replacement Policies of Cache Memory

Vidya Dahake Department of ECE, RCOEM Nagpur University, Nagpur-13

Nitesh Malewar Department of Power Engineering, NPTI Nagpur University, Nagpur

ABSTRACT

In current generation computers , memory hierarchy is formed by keeping cache on or outside the processor, registers inside, and virtual memory on (seconday memory) Hard disk. The concept of locality of reference is used to produce memory hierarchy work efficiently. In recent years various advances have been made to improve the cache memory performance on the basis of page fault rate, latency, speed, replacement policies and energy consumption. Cache replacement policy is important design parameter which affects the overall performance of processor and also more important with recent technological moves towards fully associative cache. This paper provides a survey of current generation processors on the basis of various factors effecting cache memory performance. The main point of this paper is the study and performance analysis of the cache replacement policies.

KEYWORDS Page fault rate, Hit rate, Performance of cache , LRU, LFU, FIFO

INTRODUCTION

Cache, a fastest semiconductor memory is used to store frequently subset of data or instruction from relatively slower memory. It avoids having to go to main memory every time when this same information is required. The table shows the caching hierarchy

Cache Type	What Cached	Where Cached	Latency in cycles	Managed By
Registers	4-byte word	CPU registers	0	Compiler
TLB	Address translation	On-Chip TLB	0	Hardware
L1 cache	32-byte block	On-Chip L1	1	Hardware
L2 cache	32-byte block	On-Chip L2	10	Hardware
Virtual Memory	4-KB page	Main Memory	100	Hardware + OS
Buffer Cache	Parts of files	Main Memory	100	OS
Network buffer cache	Parts of files	Local disk	10,000,000	AFS/NFS client
Browser cache	Web pages	Local disk	10,000,000	Web browser
Web cache	Web pages	Remote server disks	1,000,000,000	Web proxy server

The concept of locality of reference is used to get data or instructions from program. At one time the processor accesses a small portion of address space. Cache memory performance is calculated on the basis of page fault rate, miss penalty, and average access time. Page fault Rate is defined as the fraction of memory accesses that are not found in the cache while The percentage of accesses that result in cache hits is known as the hit rate or hit ratio of the cache. Miss Penalty is defined as the total number of cycles CPU is stalled for a memory access determined by the sum of Cycles (time) to

replace a block in the cache, upper level and Cycles (time) to deliver the block to the processor. Average Access Time and CPU execution time is calculated as:

Average Access Time = HT x HR + MP x MR

CPU Execution Time = (CCC + MSC) x CCT

MSC = Number of Misses x MP

= IC x (Misses / Instructions) x MP

= IC x [(Memory Access / Instructions)] x MR x MP

here, HT= Hit Time, HR= Hit Rate

MP= Miss Penalty, MR= Miss Rate,

CCC= CPU Clock Cycles, MSC= Memory Stall Cycles

CCT= clock cycle time.

Number of cycles for memory read and memory write can be different similarly Miss penalty to read can be different from write.

Memory Stall Clock Cycles = (Memory read stall cycles) + (Memory write stall cycles)

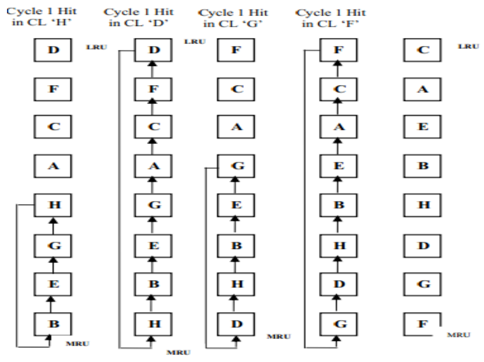
STRATEGY OF CACHE DESIGN

The three main units of a processor are data, execution, and storage unit. The data unit is responsible for organizing data of a program to be fetched and decode. The execution unit perform arithmetic and logical operations and execute instructions. The storage unit establishes interface through a temporary storage between other two units. The essential components of storage unit are cache memory, Translation Look-aside Buffer (TLB). Address Space Identifier Table (ASIT), a Buffer Invalidation Address Stack (BIAS) and write through buffers may also be available in storage unit. Technology has made it possible to fabricate millions transistors on a single chip because of which a small portion is needed to make a powerful processor. To minimize inter-chip data transfers, on-chip memory is placed inside the processor. Table1 shows cache design strategy and specifications of a various recent processors launched by Intel and AMD. There are various techniques of mapping for determining cache organization. A mapping technique is used to map large number of main memory blocks into a small number of lines of the cache memory and tag bits within every cache line examine which block of main memory is currently available in a particular cache line. Out of three mapping approaches i.e. direct mapping, associative mapping and set associative mapping, set associative caches are considered best because of highest hit rate and less access time. But beyond a certain limit, increasing cache size has more of an impact than increasing associativity. Replacement policies plays an important role in the design of cache memory because it makes decision to select a particular line of cache memory is to be replaced with the desired main memory block. First in First Out (FIFO), optimal and Least Recently Used (LRU) are algorithms used for making such decision. Least Recently Used (LRU) is the most effective policy because it is easy to implement, according to this more recently used words are likely to be referenced again. Write Caching, Write Back and Write Through are three main caching policies that decide how consistency is maintained between cache lines and corresponding main memory blocks. In write back policy write operations are made to cache only, the main memory is updated only when the corresponding cache line is flushed from cache memory. In write through policy write operations are performed to main memory along with the cache memory. The write back policy can result in inconsistency if two caches hold same line, and line is updated in one cache, then the other cache will unknowingly holds an invalid value. Inconsistency can also occur with the write-through policy, unless the other caches monitor to memory traffic or get direct notification of update. So the considerable traffic is generated in both write through and write back policy, a single bit error of any of these cannot be tolerated unless Error Correcting Code (ECC) is provided.

Mostly, Level 1 cache consists of Level 1 data cache and Level 1 instruction cache. When the first on chip cache made an appearance then some designs consisted of a single cache to store references to both instructions and data. Recently, it has become common to split cache memory into two parts one for instructions and the other for data. When the processor attempts to fetch an instruction from main memory then first it consults with the L1 instruction cache similarly, when the processor attempts to fetch data from main memory then first it consults with the L1 data cache. The main advantage of a unified cache is higher hit rate than split caches because it balances load between data and instruction fetches automatically. In unified cache only one cache design and implementation needed. On the other hand the split cache removes problem of contention between fetch/decode and execution unit. This contention can reduce performance by interfering with instruction pipeline. But, many implementation attribute of processors like replacement algorithm, mapping function and write policies are not publicly available. Intel x86 processors and AMD processors employ a direct-mapped Level 1 cache, and Level 2 cache between 2 to 4 way set associative. The L3 and higher level caches could be between 16-way to 64-way set associative. Most of them use LRU (least recently used replacement policy), and a write-back cache.

CACHE REPLACEMENT POLICIES

Today's processors include multiple levels of cache memory and the high associativity [4] has made it important to recheck the effectiveness of various cache replacement policies. In cache memory, when all the lines in a set of cache become full and a new block from main-memory needs to be placed in cache, then the cache controller has to be remoce a line from cache set and replace it with the new block from main-memory. The modern processors employ cache replacement policies such as LRU (Least Recently Used)[5], Random[6], FIFO(First in First Out)[4], optimal. In all these policies, except Random, determine which block of cache memory to replace by looking only at the past references. Least Recently Used replacement needs a number of status bits to maintain record of each cache block accessed. If the set-associativity increases, the number of these bits also increases. Random replacement policy can be used to reduce the complexity and cost of LRU replacement policy but at the expense of performance. Recent studies describe cache design space with relatively finite associativity, and consider only true Least Recently Used replacement policy [7]. The Least Recently Used replacement policy uses access pattern of a program memory to predict that most recently accessed cache line will most likely to be accessed again, and the cache line which has been Least Recently Used will be replaced by cache controller. The LRU stack is as



Even though the Least Recently Used policy is more efficient, it requires a number of bits to maintain a record for each block, contains details such that when a block is processed before. In LRU algorithm, each time when a cache hit or miss occurs then the block shifting in LRU frame requires more time and more power. Random cache replacement policy can be used to minimize the complexity and cost of LRU. Random replacement policy selects a candidate block to be removed randomly from all the cache lines in the set. This policy does not requires to keep any information of access history. It has been used in ARM processors for its simple designing. In optimal cache replacement policy a counter is assigned to each cache block, which loaded in cache memory. For each reference of block the counter is incremented by one. When the cache is full and has a new block to be inserted, then the block with the lowest counter is dicarded.

METHODOLOGY

Here we used SMP3.0, is a trace-driven simulator for the analysis and teaching of cache memory systems on symmetric multiprocessors. [8]. This simulator is a cost effective technique of performance evaluation of computer system design, specially for cache design, TLB, and paging system. The simulator has a full graphic and user-friendly interface, and it operates on PC systems with Windows. In this, we used some SPEC92 Benchmarks such as: HYDRO, NASA7, CEXP, MDLJD, EAR, COMP, WAVE, SWM and UPCOMP for the analysis of replacement policies.

SIMULATION SETUP

Number of Processor = 1

Cache Coherence Protocol = MESI

Bus Arbitration = optimal

Word Wide (bits) = 16

Blocks in Main Memory = 8192

Block size = 32 bytes

Main Memory size = 256 K Bytes

Blocks in Cache = 128

Cache size = 4KB

Mapping = 8 way- set associative mapping

Writing Strategy = Write Back

Replacement Policies = RANDOM, FIFO, optimal, LRU

CONCLUSION

In this, we studied the recent advances made in the design of cache for improving memory management unit access time, energy consumption. In this we describes the cache replacement policies algorithm in the form of their performance analysis. The analysis of experimental results shows that LRU is the most scalable cache replacement policy. From this paper, we found that speed of processors growing continuously, so eliminating cache misses is more important parameter in the overall performance of the Processor. So caches becoming more set associative and more significance.

REFERENCES

1. C. Kozyrakis, "Advanced Caching Techniques." 2008.
2. Swadhesh Kumr, Dr. PK Singh, "An Overview of Hardware Based Cache Optimization Techniques," International Journal of Advance Research in Science and Engg, vol. no. 4, Special Issue (01), September 2015.
3. Gavrichenkov, "First Look at Nehalem Microarchitecture", November2008,http://www.xbitlabs.com/articles/cpu/display/Nehalem microarchitecture.html.
4. Intel Xscale-Core ,Developer's Manual,December 2000, http://developer.intel.com
5. Ackland B., Anesko D., Brinthaupt D., Daubert S.J, Kalavade A.., Knoblock J., Micca E., Moturi M., Nicol C.J., O'Neill J.H., Othmer J., Sackinger E., Singh K. J.,Sweet J., Terman C. J., and Williams J., "A Single-Chip,1.6 Billion, 16-b MAC/s Multiprocessor DSP," IEEE Journal of Solid-state circuits, Vol. 35, No. 3, March 2000, pp. 412-423.
6. David A. Patterson, John L. Hennessy. "Computer organization and design: the hardware/software interface". 2009
7. Cantin J. F, Hill M. D., Cache Performance of the SPEC CPU2000Benchmarks,http://www.cs.wisc.edu/multifacet/misc/spec2000 cachedata
8. Miguel A. VegaRodríguez, Juan M. SánchezPérez,Juan A. GómezPulido "An Educational Tool for Testing Caches on Symmetric Multiprocessors".Microprocessors and Microsystems, Elsevier Science, vol. 25, no. 4, pp. 187194. June 2001. ISSN 01419331.
9. 2nd IEEE International Conference on Engineering and Technology (ICETECH), 17th & 18th March 2016, Coimbatore, TN, India. 978-1-4673-9916-6/16/$31.00 ©2016 IEEE An Overview of Modern Cache Memory and Performance Analysis of Replacement Policies