COMPUTER ORGANIZATION AND ARCHITECTURE

Course Code : CSE 2151

Credits : 04

PERFORMANCE CONSIDERATIONS

- Hit Rate and Miss Penalty
- Caches on the Processor Chip
- Other Enhancements

HIT RATE AND MISS PENALTY

- Hit rate:
  - number of hits stated as a fraction of all attempted accesses
- Miss rate:
  - number of misses stated as a fraction of attempted accesses
- The entire memory hierarchy would appear to the processor as a single memory unit that has the access time of the cache on the processor chip and the size of the magnetic disk
- The total access time seen by the processor when a miss occurs -miss penalty

- System with only one level of cache
  - the time to access a block of data in the main memory.
- Let h be the hit rate, M the miss penalty, and C the time to access information in the cache
- The average access time experienced by the processor is
- $t_{avg} = hC + (1 - h)M$

EXAMPLE

- Consider a computer that has the following parameters. Access times to the cache and the main memory are τ and 10τ , respectively. When a cache miss occurs, a block of 8 words is transferred from the main memory to the cache. It takes 10τ to transfer the first word of the block, and the remaining 7 words are transferred at the rate of one word every τ seconds.

- Miss penalty in this computer is given by:

$$M = \tau + 10\tau + 7\tau + \tau = 19\tau$$

EXAMPLE

- Assume that 30 percent of the instructions in a typical program perform a Read or a Write operation, which means that there are 130 memory accesses for every 100 instructions executed. Assume that the hit rates in the cache are 0.95 for instructions and 0.9 for data. Assume further that the miss penalty is the same for both read and write accesses

$$\frac{\text{Time without cache}}{\text{Time with cache}} = \frac{130 \times 10\tau}{100(0.95\tau + 0.05 \times 19\tau) + 30(0.9\tau + 0.1 \times 19\tau)} = 4.7$$

- cache makes the memory appear almost five times faster than it really is
- Let us consider how effective the cache of this example is compared to the ideal case in which the hit rate is 100 percent. 100% hit rate in the cache would make the memory appear twice as fast as when realistic hit rates are used.

$$\frac{\text{Time for real cache}}{\text{Time for ideal cache}} = \frac{100(0.95\tau + 0.05 \times 19\tau) + 30(0.9\tau + 0.1 \times 19\tau)}{130\tau} = 2.1$$

HOW CAN THE HIT RATE BE IMPROVED?

- Make the cache larger
  - increased cost
- increase the cache block size while keeping the total cache size constant, to take advantage of spatial locality
  - the performance of a computer is affected positively by increased hit rate and negatively by increased miss penalty
  - block size should be neither too small nor too large.
  - In practice, block sizes in the range of 16 to 128 bytes are the most popular choices
- miss penalty can be reduced if the load-through approach is used when loading new blocks into the cache.

CACHES ON THE PROCESSOR CHIP

- Most processor chips include at least one L1 cache. Often there are two separate L1 caches, one for instructions and another for data.
- In high-performance processors, two levels of caches are normally used, often implemented on the processor chip.
  - separate L1 caches for instructions and data –fast-10's of KB
  - a larger L2 cache-slower but larger than L1- only affects the miss penalty of the L1 caches-100s of KB or MB
- The average access time experienced by the processor in such a system is:

$$t_{avg} = h_1 C_1 + (1 - h_1)(h_2 C_2 + (1 - h_2)M)$$

where

h_1 is the hit rate in the L1 caches.

h_2 is the hit rate in the L2 cache.

C_1 is the time to access information in the L1 caches.

C_2 is the miss penalty to transfer information from the L2 cache to an L1 cache.

M is the miss penalty to transfer information from the main memory to the L2 cache.

EXERCISE PROBLEM

- Suppose that a computer has a processor with two L1 caches, one for instructions and one for data, and an L2 cache. Let τ be the access time for the two L1 caches. The miss penalties are approximately 15τ for transferring a block from L2 to L1, and 100τ for transferring a block from the main memory to L2. Assume that the hit rates are the same for instructions and data and that the hit rates in the L1 and L2 caches are 0.96 and 0.80, respectively.
  a. What fraction of accesses miss in both the L1 and L2 caches, thus requiring access to the main memory?
  b. What is the average access time as seen by the processor?
  c. Suppose that the L2 cache has an ideal hit rate of 1. By what factor would this reduce the average memory access time as seen by the processor?
  d. Consider the following change to the memory hierarchy. The L2 cache is removed and the size of the L1 caches is increased so that their miss rate is cut in half. What is the average memory access time as seen by the processor in this case?

EXERCISE PROBLEM: SOLUTION

(*a*) The fraction of memory accesses that miss in both the L1 and L2 caches is

$$(1 - h_1)(1 - h_2) = (1 - 0.96)(1 - 0.80) = 0.008$$

(*b*) The average memory access time using two cache levels is

$$\begin{aligned} t_{avg} &= 0.96\tau + 0.04(0.80 \times 15\tau + 0.20 \times 100\tau) \\ &= 2.24\tau \end{aligned}$$

(*c*) With no misses in the L2 cache, we get:

$$t_{avg}(\text{ideal}) = 0.96\tau + 0.04 \times 15\tau = 1.56\tau$$

Therefore,

$$\frac{t_{avg}(\text{actual})}{t_{avg}(\text{ideal})} = \frac{2.24\tau}{1.56\tau} = 1.44$$

(*d*) With larger L1 caches and the L2 cache removed, the access time is

$$t_{avg} = 0.98\tau + 0.02 \times 100\tau = 2.98\tau$$

OTHER ENHANCEMENTS: WRITE BUFFER

- Write-through protocol
  - Each Write operation results in writing a new value into the main memory.
  - If the processor must wait for the memory function to be completed, processor is slowed down by all Write requests.
  - Processor does not need immediate access to the result of a Write operation; not necessary for it to wait for the Write request to be completed. Write buffer included for temporary storage of Write requests.
  - Processor places each Write request into this buffer and continues execution of the next instruction.
  - The information rom the buffer is sent to the main memory whenever the memory is not responding to Read requests.
  - Read requests be serviced quickly, because the processor usually cannot proceed before receiving the data being read from the memory. These requests are given priority over Write requests
  - The Write buffer may hold several Write requests.
  - subsequent Read request may refer to data that are still in the Write buffer.
  - To ensure correct operation, the addresses of data to be read from the memory are always compared with the addresses of the data in the Write buffer.
  - In the case of a match, the data in the Write buffer are used.

OTHER ENHANCEMENTS: WRITE BUFFER

- Write-back protocol
  - Write commands issued by the processor are performed on the word in the cache.
  - When a new block of data is to be brought into the cache as a result of a Read miss, it may replace an existing block that has some dirty data.
  - The dirty block must be written into the main memory.
  - If the required write-back is performed first, then the processor must wait for this operation to be completed before the new block is read into the cache.
  - It is more prudent to read the new block first.
  - The dirty block being ejected from the cache is temporarily stored in the Write buffer and held there while the new block is being read.
  - Afterwards, the contents of the buffer are written into the main memory.
  - Write buffer also works well for the write-back protocol.

OTHER ENHANCEMENTS: PREFETCHING

- new data are brought into the cache when they are first needed.
- Following a Read miss, the processor has to pause until the new data arrive, thus incurring a miss penalty.
- To avoid stalling the processor, it is possible to prefetch the data into the cache before they are needed
- Through
  - software -prefetch instruction
  - hardware- using circuitry that attempts to discover a pattern in memory references and prefetches data according to this pattern
- Prefetch-Executing this instruction causes the addressed data to be loaded into the cache, as in the case of a Read miss.
- The hope is that prefetching will take place while the processor is busy executing instructions that do not result in a Read miss, thus allowing accesses to the main memory to be overlapped with computation in the processor
- Inserted either by programmer or compiler
- Overhead
  - Increases length of program
  - Data may not be used by instructions that follow-if the prefetched data are ejected from the cache by a Read miss involving other data

OTHER ENHANCEMENTS: LOCKUP-FREE CACHE

- Software prefetching does not work well if the action of prefetching stops other accesses to the cache until the prefetch is completed
- While servicing a miss, the cache is said to be locked.
- modify the basic cache structure to allow the processor to access the cache while a miss is being serviced
- A cache that can support multiple outstanding misses is called lockup-free.
- cache must include circuitry that keeps track of all outstanding misses

TOPICS COVERED FROM

- Textbook 1:
  - Chapter 8: 8.7