Everything from the book “**COMPUTER ORGANIZATION AND EMBEDDED SYSTEMS, 6th Edition”**

1. 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 15*τ* for transferring a block from L2 to L1, and 100*τ* for transferring a block from the main memory to L2. For the purpose of this problem, 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?

**Solution:** The average memory access time is *tavg* = *hC* + *(*1 − *h)M.*

With L1 and L2 caches, the average memory access time is

*tavg* = *h*1*C*1 + *(*1 − *h*1*)(h*2*C*2 + *(*1 − *h*2*)M)*

(*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

*tavg* = 0*.*96*τ* + 0*.*04*(*0*.*80 × 15*τ* + 0*.*20 × 100*τ)*

= 2*.*24*τ*

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

*tavg(*ideal*)* = 0*.*96*τ* + 0*.*04 × 15*τ* = 1*.*56*τ*

Therefore,

*tavg(*actual*)/ tavg(*ideal*) =*  2*.*24*τ/* 1*.*56*τ*

= 1*.*44

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

*tavg* = 0*.*98*τ* + 0*.*02 × 100*τ* = 2*.*98*τ*

1. Consider a long sequence of accesses to a disk with an average seek time of 6 ms and an average rotational delay of 3 ms. The average size of a block being accessed is 8K bytes. The data transfer rate from the disk is 34 Mbytes/sec.
2. Assuming that the data blocks are randomly located on the disk, estimate the average percentage of the total time occupied by seek operations and rotational delays.
3. Repeat part (*a*) for the situation in which disk accesses are arranged so that in 90 percent of the cases, the next access will be to a data block on the same cylinder.

**Solution:** It takes 8K*/*34M = 0*.*23 ms to transfer a block of data.

1. The total time needed to access each block is 6 + 3 + 0*.*23 = 9*.*23 ms.

The portion of time occupied by seek and rotational delay is 9*/*9*.*23 = 0*.*97 = 97%.

1. In 90% of the cases, only rotational delays are involved. Therefore, the average time to access a block is 0*.*9 × 3 + 0*.*1 × 9 + 0*.*23 = 3*.*89 ms. The portion of time occupied by seek and rotational delay is 3*.*6*/*3*.*89 = 0*.*92 = 92%.
2. A1024 × 1024 array of 32-bit numbers is to be normalized as follows. For each column, the largest element is found and all elements of the column are divided by the value of this element. Assume that each page in the virtual memory consists of 4K bytes, and that 1M bytes of the main memory are allocated for storing array data during this computation. Assume that it takes 10 ms to load a page from the disk into the main memory when a page fault occurs.
3. Assume that the array is processed one column at a time. How many page faults would occur and how long does it take to complete the normalization process if the elements of the array are stored in column order in the virtual memory?
4. Repeat part (*a*) assuming the elements are stored in row order?
5. Propose an alternative way for processing the array to reduce the number of page faults when the array is stored in the memory in row order. Estimate the number of page faults and the time needed for your solution.

**Solution:** Each 32-bit number comprises 4 bytes. Hence, each page holds 1024 numbers.

There is space for 256 pages in the 1M-byte portion of the main memory that is allocated for storing data during the computation.

(*a*) Each column is stored in one page; there is a page fault to bring each column to the main memory, for a total of 1024 page faults.

Processing time = 1024 × 10 ms = 10.24 s.

(*b*) Processing of each column requires two passes, the first to find the largest element and the second to perform the normalization. When processing the first column, each element access results in a page fault that brings all elements of the corresponding row into the main memory. After 256 elements have been examined, the main memory is full. Accessing the next 256 elements results in page faults that replace all the data in the memory, and the process repeats. Thus, a page fault occurs for every access to every element in the array.

Processing time = 2 × 1024 × 1024× 10 ms = 20,972 s = 5.8 hours.

(*c*) A more efficient alternative for this arrangement of the data is to complete the first pass for only one quarter of each column for all columns, then process the second quarter, and so on. The second pass is handled in the same way. In this case, each pass through the array results in 1024 page faults, for a total of 2048.

Processing time = 2048 × 10 ms = 20.48 s.