Chapter 9

9.5 Assume we have a demand-paged memory. The page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified, and 20 milliseconds if the replaced page is modified（在存在空闲页帧的条件下，处理一次缺页的时间是8毫秒。如果没有空闲页面，但待换出页面并未更改，处理一次缺页的时间也是8毫秒。如果待换出页面已被更改，则需要20毫秒。）. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page-fault rate for an effective access time of no more than **200 nanoseconds**?

Answer:

First the following relationship between different time scale measurements are used: 1 second = 1000 ms, 1 ms = 1000 us, and 1 us = 1000 ns.

Assume that the probability of page fault is p. Then the average memory access time can be expressed by:

$$T_{avg} = (1-p).access\_time\_of\_hit + p.(access\_time\_of\_page\_fault)$$

In the case of hit, the access time contains only memory access time, which is 100 ns. When there is a page fault, we have an additional cost to do the page replacement. Since 70% of pages to be swapped out are dirty pages, the average time of page replacement can be calculated as:

$$T_{rplace} = (1-0.7).replace\_time\_of\_no\_write\_back + 0.7.(replace\_time\_with\_dirty\_page\_writeback)$$

From the assumption of the question, we know that the page replacement time with dirty page write back is 20 ms ($20*10^6$ ns). The page replacement without write back is 8 ms ($8*10^6$ ns). Thus we can rewrite the average access time as:

$$\begin{aligned} T_{avg} &= (1-p)*100 + p.(T_{rplace} + 100) \\ &= (1-p)*100 + p((1-0.7)*8*10^6 + 0.7*20*10^6 + 100) \\ &= (16.4*10^6 - 100)p + 100 \end{aligned}$$

If we want to bound the average access time below 200 ns, we must have:

$$T_{avg} < 200$$

$$(16.4*10^6 - 100)p + 100 < 200$$

$$p < \frac{100}{16.4*10^6 - 100} \approx 6*10^{-6}$$

Thus the probability of page fault should be less than 0.000006.

9.10 Consider a demand-paging system with the following time-measured utilizations:

CPU utilization 20%

Paging disk 97.7%
Other I/O devices 5%
Which (if any) of the following will (probably) improve CPU utilization?
Explain your answer.
a. Install a faster CPU.
b. Install a bigger paging disk.
c. Increase the degree of multiprogramming.
d. Decrease the degree of multiprogramming.
e. Install more main memory.
f. Install a faster hard disk or multiple controllers with multiple hard disks.
g. Add prepaging to the page fetch algorithms.
h. Increase the page size.
Answer: The system obviously is spending most of its time paging, indicating over-allocation of memory. If the level of multiprogramming is reduced resident processes would page fault less frequently and the CPU utilization would improve. Another way to improve performance would be to get more physical memory or a faster paging drum.
a. Get a faster CPU—No.
b. Get a bigger paging disk—No.
c. Increase the degree of multiprogramming—No.
d. Decrease the degree of multiprogramming—Yes.
e. Install more main memory—Likely to improve CPU utilization as more pages can remain resident and not require paging to or from the disks..
f. Install a faster hard disk, or multiple controllers with multiple hard disks—Also an improvement, for as the disk bottleneck is removed by faster response and more throughput to the disks, the CPU will get more data more quickly.
g. Add prepaging to the page fetch algorithms—Again, the CPU will get more data faster, so it will be more in use. This is only the case if the paging action is amenable to prefetching (i.e., some of the access is sequential).
h. Increase the page size—Increasing the page size will result in fewer page faults if data is being accessed sequentially. If data access is more or less random, more paging action could ensue because fewer pages can be kept in memory and more data is transferred per page fault. So this change is as likely to decrease utilization as it is to increase it.

9.13 A page-replacement algorithm should minimize the number of page faults. We can do this minimization by distributing heavily used pages evenly over all of memory, rather than having them compete for a small number of page frames.We can associate with each page frame a counter of the number of pages that are associated with that frame. Then, to replace a page, we search for the page frame with the smallest counter.
a. Define a page-replacement algorithm using this basic idea. Specifically address the problems of (1) what the initial value of the counters is, (2) when counters are increased, (3) when counters are decreased, and (4) how the page to be replaced is selected.
b. How many page faults occur for your algorithm for the following reference string, for four page frames?
1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2.

c. What is the minimum number of page faults for an optimal page replacement strategy for the reference string in part b with four page frames?

Answer:

a. Define a page-replacement algorithm addressing the problems of:

1. Initial value of the counters—0.
2. Counters are increased—whenever a new page is associated with that frame.
3. Counters are decreased—whenever one of the pages associated with that frame is no longer required.
4. How the page to be replaced is selected—find a frame with the smallest counter. Use FIFO for breaking ties.

b. 14 page faults

c. 11 page faults

9.0 Consider the following page reference string:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.

How many page faults would occur for the following replacement algorithms, assuming three, four frames? Remember all frames are initially empty, so your first unique pages will all cost one fault each.

- LRU replacement
- FIFO replacement
- Optimal replacement

Answer:

Number of frames	LRU	FIFO	Optimal
3	15	16	11
4	10	14	8