HW4 歐佳昀 110590450

Chap. 10

10.21: Assume that 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 frame is available or if the replaced page is not modified and 20 milliseconds if the replaced page is modified.

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?

Time taken to service page Fault for empty page or unmodified page = 8ms.

Time taken to service page Fault for modified page = 20ms

Memory access time = 100ns

Effective Access time = 200ns

P = page fault 的機率

1 milliseconds (ms) is equal to 1,000,000 nanoseconds (ns)

ANS:

page Fault service time = TLB time + page table time + page fetch from disk
Find the effective access time (EAT) EAT = EAT = (1-p)*(100ns) + (p) * ((1-0.7) * (8ms) + (0.7)*(20m)) = 100 - 100p + p*(0.3*8,000,000 + 0.7*20,000,000) 200 <= 100-100p+p(2,400,000 + 14,000,000) 200 <= 100-100p+16,400,000p 100 <= 16,399,900p P <=100/16,399,900 ≒ 0.000006

10.24: Apply the (1) FIFO (2) LRU (3) Optimal (OPT) replacement algorithms for the following page reference string:

3, 1, 4, 2, 5, 4, 1, 3, 5, 2, 0, 1, 1, 0, 2, 3, 4, 5, 0, 1.

Indicate the number of page faults for each algorithm assuming demand paging with three frames.

ANS:

(1) FIFO : 15 fault	(2) LRU : 16 fault	(3) Optimal : 13 fault
[-1,-1, 3], 3, Fault	[-1,-1, 3], 3, Fault	[-1,-1, 3], 3, Fault
[-1, 1, 3], 1, Fault	[-1, 1, 3], 1, Fault	[-1, 1, 3], 1, Fault
[4, 1, 3], 4, Fault	[4, 1, 3], 4, Fault	[4, 1, 3], 4, Fault
[4, 1, 2], 2, Fault	[4, 1, 2], 2, Fault	[4, 1, 2], 2, Fault (5, 4, 1, 3, 5, 2, 0, 1, 1, 0, 2, 3, 4, 5, 0, 1.)
[4, 5, 2], 5, Fault	[4, 5, 2], 5, Fault	[4, 1, 5], 5, Fault (4, 1, 3, 5, 2, 0, 1, 1, 0, 2, 3, 4, 5, 0, 1.)
[4, 5, 2], 4	[4, 5, 2], 4	[4, 1, 5], 4,
[1, 5, 2], 1, Fault	[4, 5, 1], 1, Fault	[4, 1, 5], 1,
[1, 5, 3], 3, Fault	[4, 3, 1], 3, Fault	[3, 1, 5], 3, Fault (5, 2, 0, 1, 1, 0, 2, 3, 4, 5, 0, 1.)
[1, 5, 3], 5,	[5, 3, 1], 5, Fault	[3, 1, 5], 5,
[1, 2, 3], 2, Fault	[5, 3, 2], 2, Fault	[3, 1, 2], 2, Fault (0, 1, 1, 0, 2, 3, 4, 5, 0, 1.)
[0, 2, 3], 0, Fault	[5, 0, 2], 0, Fault	[0, 1, 2], 0, Fault (1, 1, 0, 2, 3, 4, 5, 0, 1.)
[0, 2, 1], 1, Fault	[1, 0, 2], 1, Fault	[0, 1, 2], 1,
[0, 2, 1], 1,	[1, 0, 2], 1,	[0, 1, 2], 1,
[0, 2, 1], 0,	[1, 0, 2], 0,	[0, 1, 2], 0,
[0, 2, 1], 2,	[1, 0, 2], 2,	[0, 1, 2], 2,
[0, 3, 1], 3, Fault	[3, 0, 2], 3, Fault	[0, 1, 3], 3, Fault (4, 5, 0, 1.)
[4, 3, 1], 4, Fault	[3, 4, 2], 4, Fault	[0, 1, 4], 4, Fault (5, 0, 1.) - 3
[4, 3, 5], 5, Fault	[3, 4, 5], 5, Fault	[0, 1, 5], 5, Fault (0,1.) - 4
[4, 0, 5], 0, Fault	[0, 4, 5], 0, Fault	[0, 1, 5], 0,
[1, 0, 5], 1, Fault	[0, 1, 5], 1, Fault	[0, 1, 5], 1,

10.37: What is the cause of thrashing? How does the system detect thrashing? Once it detects thrashing, what can the system do to eliminate this problem?

ANS:

Thrashing is caused by under allocation of the minimum number of pages required by a process, forcing it to continuously page fault. The system can detect thrashing by evaluating the level of CPU utilization as compared to the level of multiprogramming. It can be eliminated by reducing the level of multiprogramming.

Chap.11

11.13: Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4,999.

The drive is currently serving a request at cylinder 2,150, and the previous request was at cylinder 1,805.

The queue of pending requests, in FIFO order, is

2,069; 1,212; 2,296; 2,800; 544; 1,618; 356; 1,523; 4,965; 3,681

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests, for each of the following disk-scheduling algorithms?

(a) FCFS = 13011

(b) SCAN = 7492

(c) C-SCAN = 9917

Method	1	2	3	4	5	6	7	8	9	10	11	12	13	total
FCFS	2150	2069	1212	2296	2800	544	1618	356	1523	4965	3681			13011
SCAN	2150	2269	2800	3681	4965	4999	2069	1618	1523	1212	544	356		7492
C-SCAN	2150	2269	2800	3681	4965	4999	0	356	544	1212	1523	1618	2069	9917

ANS:

(a) FCFS = 81+857+1084+504+2256+1074+1262+1167+3442+1284=13011

(b) SCAN = 146 + 504 + 881 + 1284 + 34 (4999-4965) + 2930 (4999-2069) + 451 + 95 + 311 + 668 + 188 = 7492

(c)C-SCAN=146+504+881+1284+34(4999-4965)+4999(4999-0)+356(356-0)+188+668+311+95+451=9917

11.20: Consider a RAID level 5 organization comprising five disks, with the parity for sets of four blocks on four disks stored on the fifth disk.

How many blocks are accessed in order to perform the following?

(a) A write of one block of data.

(b) A write of seven contiguous blocks of data.

ANS:

(a) write one block of data requires : read the parity block, read the old data stored in the target block, computation of the new parity based on the differences.
between new and old contents of the target block, and write the parity block and the target block.

(b) Assume the seven contiguous blocks begin at a four-block boundary. A write of seven contiguous blocks of data could be performed by writing the seven contiguous blocks, writing parity block of the first four blocks, reading the eight block, computing the parity for the next set of four blocks , and writing the corresponding parity block onto disk.

11.21: Compare the throughput achieved by a RAID level 5 organization with that achieved by a RAID level 1 organization.

(a) Read operations on single blocks.

(b) Read operations on multiple contiguous blocks.

(a)

The amount of throughput depends on the number of disks in the RAID system. A RAID Level 5 comprising of a parity block for every set of four blocks spread over five disks can support four to five operations simultaneously.

A RAID Level 1 comprising of two disks can support two simultaneous operations. Of course, there is greater flexibility in RAID Level 1 as to which copy of a block could be accessed and that could provide performance benefits by taking into account position of disk head.

(b)RAID Level 5 organization achieves greater bandwidth for accesses to multiple contiguous blocks since the adjacent blocks could be simultaneously accessed. Such bandwidth improvements are not possible in RAID Level 1.

Chap.14

14.14: Consider a file system on a disk that has both logical and physical block sizes of 512 bytes. Assume that the information about each file is already in memory

For each of the three allocation strategies (contiguous, linked, and indexed), answer these questions:

(a) How is the logical-to-physical address mapping accomplished in this system? (For indexed allocation, assume that a file is always less than 512 blocks long)

(b) If we are currently at logical block 10 (the last block accessed was block 10) and want to access logical block 4, how many physical blocks must be read from the disk?

ANS:

Let Z be the starting file address (block number).

Contiguous:

(a) Divide the logical address by 512 with X and Y the resulting quotient and remainder respectively. Add X to Z to obtain the physical block number. Then Y is the displacement into that block.

(b) 1.

Linked:

(a) Divide the logical physical address by 511 with X and Y the resulting quotient and remainder respectively. Chase down the linked list (getting X+1 blocks). Y +1 is the displacement into the last physical block.

(b) 4.

Indexed:

(a) Divide the logical address by 512 with X and Y the resulting quotient and remainder respectively. First get the index block into memory. The physical block address is contained in the index block at location X. Y is the displacement into the desired physical block.

(b) 2.

14.15: Consider a file system that uses inodes to represent files
Disk blocks are 8KB in size, and a pointer to a disk block requires 4 bytes
This file system has 12 direct disk blocks, as well as single, double, and triple indirect disk blocks
What is the maximum size of a file that can be stored in this file system?

ANS:
Number of ptr point to block =>block/ptrs = 8K/4 = 2048
12 direct disk blocks *8KB = 96KB
1 single disk block = 2048 * 8KB = 16384KB
1 double disk block = 2048^2 * 8 KB = 33554432 KB
1 triple disk block = 2048^3 * 8 KB = 68719476736 KB
Total = 64.03 TB