2025/06/08 (日)

- Chap.10: ~~10.21~~, 10.24, ~~10.37~~
- Chap.11: ~~11.13~~, ~~11.17~~, ~~11.21~~
- Chap.14: 14.14, 14.15

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?

8 ms

20 ms for page fault

not modify or empty

modify

Memory access 100 ns

$(1-p) \times$ Memory access $+ p \times$ (page fault + page out + page in)

EAT ≤ 200 ns

令 EAT 的 p 為機率 (所求)

AVG page fault

$0.3 \times 8 + 0.7 \times 20$

$= 2.4 + 14 \Rightarrow 16.4$ ms

16.4×10^6 ns

1.64×10^7

$(1-p) \times 100 + 1.64 \times 10^7 p \leq 200$

$100 - 100p + 1.64 \times 10^7 p \leq 200$

$$p \leq \frac{100}{16399900} \approx 6.1 \times 10^{-6}$$

#

10.24 Apply the (1) FIFO, (2) LRU, and (3) optimal (OPT) replacement algorithms for the following page-reference strings:

- 2,6,9,2,4,2,1,7,3,0,5,2,1,2,9,5,7,3,8,5
- ~~0,6,3,0,2,6,3,5,2,4,1,3,0,6,1,4,2,3,5,7~~
- ~~3,1,4,2,5,4,1,3,5,2,0,1,1,0,2,3,4,5,0,1~~
- ~~4,2,1,7,9,8,3,5,2,6,8,1,0,7,2,4,1,3,5,8~~
- ~~0,1,2,3,4,4,3,2,1,0,0,1,2,3,4,4,3,2,1,0~~

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

not in 簡報

window = 3

2,6,9,2,4,2,1,7,3,0,5,2,1,2,9,5,7,3,8,5

① fifo

2、26、269、269、469、429、421、721、731、730、530、520、521、521、921、951、957、357、387、385

18個 page fault

一是下一個要換的。

② LRU

2、26、269、269、249、249、241、271、371、370、350、250、251、251、291、295、795、735、738、538

17個 Page fault

一是當下被換的(上數字是換完的)

③ opt

6之後不用　4之後用　9最晚用　7最晚

2, 26, 269, 269, 249, 249, 219, 217, 213,
✓ ✓ ✓ X ✓ X ✓ ✓ ✓

210, 215, 215, 215, 215, 915, 975, 375, 385, 385
3最晚 ✓　5沒了 ✓　X　X　X　2沒了 ✓　1沒了 ✓　9沒了 ✓　7沒了 ✓　X

13 faults　一步不換的原因。

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?

Thrashing原因 → page fault導致 Page Replacement 太常發生。

Cause of thrashing: ① 低 CPU 利用率

Detect thrashing: ② 偵測 process 的 Page fault rate，高過一定門檻 → Thrashing

Eliminate the problem:
① Working Set Model
根據 locality 分配 frames
② Page-fault Frequency
根據 fault 狀況調整 frame 數量
③ Swap out Processes
暫停 Process 釋放記憶體

舉例其中三種
實際上有更多。

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

b. SCAN

c. C-SCAN

1805 → 2150 (往大的走)

a FCFS

2150 → 2069 → 1212 → 2296 → 2800 → 544 → 1618 → 356 → 1523 → 4965 → 3681

81 + 857 + 1084 + 504 + 2256 + 1074 + 1262 + 1167 + 3442 + 1284 = 13011

b SCAN

2150 → 2296 → 2800 → 3681 → 4965 → 4999 → 1618 → 1523 → 1212 → 544 → 356

146 + 504 + 881 + 1284 + 34 + 3381 + 95 + 311 + 668 + 188 = 7492

C. C-SCAN

2150 → 2296 → 2800 → 3681 → 4965 → 4999 → 356 → 544 → 1212 → 1523 → 1618 → 2069

146 + 504 + 881 + 1284 + 34 + 4999 + 356 + 188 + 668 + 311 +

95+451=9917

11.17 Describe some advantages and disadvantages of using NVM devices as a caching tier and as a disk-drive replacement compared with using only HDDs.

NVM 相比 HDD

優：高速、耗電更低、更耐用（無機械式元件）

缺：貴、有讀寫限制

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

a. Read operations on single blocks

b. Read operations on multiple contiguous blocks

a. RAID 1 較 RAID 5 大、快

↓ 從任一硬碟讀，可選當下較快的

↘ 其中一個有↓從該 block 的硬碟讀即可。

b. RAID 5 較 RAID 1 大、快

以兩顆硬碟為例，RAID 1 只依靠一顆硬碟去處理所有 blocks 的讀取。

RAID 5 可以將要讀取的 block 分在二顆硬碟上，吞吐量大。

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 the 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?

X: 分割後商

Y: 分割後餘

Z: Address 起點

(a) Contiguous

將 address 以 512 分割

address = Z + X × 512 + Y

(b) 1, 直接讀即可

(a) Linked

address 以 511 分割

X是區塊 Y是偏移量, 需走到 X+1 塊找 Y 為address.

(b) 5塊 (0→1→2→3→4)

(a) Indexed

將 address 以 512 分割

X為所求區塊

(b) 2次

(存 index block 一次)

對應 block 一次

14.15 Consider a file system that uses inodes to represent files. Disk blocks are 8 KB 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?

8KB / 4bytes => 2048 個指標

12 × 8 => 96KB

single $2048 \times 8KB = 16384KB \approx 16MB$

double $2048^2 \times 8KB = 33554432KB \approx 32GB$

Triple $2048^3 \times 8KB = 68719476736KB$

$\approx 64TB$

total $64TB + 32GB + 16384KB + 96KB$

⋮

64.03TB