磁碟管理與檔案管理

Disk Access Time

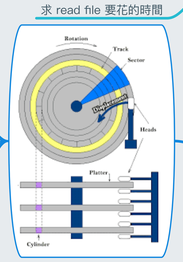
= seek time + latency time + transfer time

- 磁頭(head)移動到 track 所需的時間 — Seek time
- 將 sector 旋轉到磁頭位置所需的時間 — Latency time (Rotation/Revolution)
- Data 傳輸的時間 — Transfer time
- 其中 seek time 通常 >>> 其他兩個

計算

- rpm = revolution per minute — 給 rpm 求平均 Revolution time
- 求 transfer rate(每秒傳輸資料量)
- 求 read file 要花的時間

Disk system



系統組成

- Disks
- Disk sides
- Track(磁軌)
- sector(Block、磁區)
- cylinder

計算

- 求 Disk system size

File Allocation Methods

Contiguous Allocation 連續式

優點

- 需要找到足夠的連續空間才可配置
- File Name / start / size — File allocation info.
- 平均 seek time 少(連續 Block 物理上都附近)
- 支援循序存取(大家都支援)
- 支援隨機存取

缺點

- 以磁碟重組解決 (相當於記憶體中的 Compaction) — 外部破碎
- 檔案大小無法動態擴充
- Create File 須事先宣告大小

Linked Allocation 連結式

優點

- Blocks 之間以 Link 串連 — 只要有足夠的 Block 即可配置
- File Name / start / End — File allocation info.
- 無外部破碎
- 檔案大小擴充容易
- Create File 不用 size-declaration

缺點

- 平均 seek time 長
- 無法支援隨機存取
- 可靠度差
- 因為必須在硬碟中讀取 link info = I/O 負擔 — 循序存取速度較慢

實例

- 微軟 FAT method (File Allocation table) — 不在硬碟中儲存 linking info、存在儲存於記憶體的 FAT
  - linking info 存取較快
  - 循序存取較快

Index Allocation 索引式

優點

- 除了檔案本身配置的 Blocks 之外 額外配置 Index Blocks 來記錄 Allocated Blocks No.
- File Name / Index Block No. — File allocation info.
- 無外部破碎
- 可支援循序存取與隨機存取
- 檔案大小擴充容易

缺點

- Index Block 需佔用額外的硬碟空間
- Index Block 本身通常不會被完全使用(空間浪費)
- 可能需要多個 Index Block 儲存配置資訊 — 解決方案
  - 多個 Index Blocks 之間以 Linking 串連 — 缺點
    - 平均 I/O 負擔增加
    - 隨機存取不直接
  - MultiLevel Index Structure — 優點
    - 任一個 Data Block 的 I/O 次數相同
    - 大型檔案適用
  - 小型檔案不適用 — 缺點
    - Index Blocks 可能超過 Data Block 數
- Unix I-Node (結合前兩個)

實例

- Unix I-Node
  - 以 15 個 entry 的 I-Node 為例
    - Data Block No. — 1 ~ 12
    - pointer of single-level index — 13
    - pointer of two-level index — 14
    - pointer of three-level index — 15
  - 優點
    - 大小檔案皆適用
  - 計算
    - I-Node 支援的 Maximum File size
    - 存取特定 Block 所需的 I/O 次數
    - 循序存取到某 Block 的 I/O 次數

File 管理

緣由

避免對檔案的任何操作都要 search(配置資訊) & I/O request

- File open
  - 首次使用 File 時，search 結束後將資訊存於 kernel memory 中的某個 table(open file table)
  - 之後的操作就是存取記憶體中的指標
  - Search file time 下降
  - I/O 次數下降
  - e.g. C 語言開檔後回傳指標 然後就都是指標操作
- File close — 將 open file table 中的資訊存回硬碟中的實體目錄，並將之自 table 中移除

File protection

- physical protection — 備份
- logical protection
  - Name
  - password
  - Access list
  - Access group

Unix Access group

- user
  - Owner
  - group(member)
  - others
- 權限
  - read — r
  - write — w
  - execute — x
- e.g. “chmod 754”
  - 111 –> 7 — r w x — Owner
  - 101 –> 5 — r _ x — group(member)
  - 100 –> 4 — r _ _ — others

Consistency Semantics

- Unix Semantics
  - 訂票系統的座位表
  - 需要互斥存取
  - 所有更動需要同步
- Session Semantics
  - 雲端上的空白表單
  - 無須互斥存取
  - 各地內容不須一致
- Immutable Semantics
  - 公文或者公告文件(PDF)
  - Read-only
  - 檔名唯一

Disk Free space 管理

Bit vector(BitMap)

- 每個 Block 用一個 bit 紀錄於 memory
- 優點
  - 簡單易實施
  - 容易找到連續的 free space
- 缺點 — 不適用大型 Disk
  - Block 多，則 BitMap 大
  - 太佔記憶體空間
  - 可能因為 Virtual memory 技術導致儲存於硬碟中、增加額外 I/O 負擔

Link list

- 直接在 Disk free space 上儲存 pointer、以 link list 的方式串接
- 優點 — 大型 Disk 適用
- 缺點
  - 不易找出連續 free Blocks — 解決方案：Grouping
  - 在 Disk 上依序 traverse 是大量的 I/O 負擔 — 解決方案：Counting

Grouping

- 在 Disk free space 上除了指向下一個 free Block 的 pointer 之外，額外儲存和些許 free Block 的編號
- 優點 — 較容易找到連續 free Blocks

Counting

- 在 Disk free space 上紀錄其後的連續 free Block 數目
- 優點 — 若連續可用空間多，可大幅降低串列長度

Disk scheduling algo.

磁碟排班中沒有最好的，也沒有最佳的

FCFS

- 先來後到
- 簡單易實施
- 公平無飢餓
- 效能不佳

SSTF

- 最近的優先
- 效能不錯
- 不公平、可能會飢餓

SCAN

- 一去一返來回掃描，盡頭折返
- 電梯演算法(原文書如此敘述、事實上應該是 LOOK)
- 效能不錯
- 某些狀況下不公平 — 折返途中，離盡頭近的會有兩次機會
- 盡頭折返會有不必要的浪費

C-SCAN

- 一去一返來回掃描，盡頭折返、返程不提供服務
- 比 SCAN 相對公平

LOOK

一去一返，方向上無其他 request 時返回(非盡頭)

C-LOOK

一去一返，方向上無其他 request 時返回(非盡頭)、返程不提供服務

計算 — 求 Track 移動總數

其他 & 名詞

striping

- 平行運作
- 提升效能

mirror

- 鏡像運作
- 提升可靠度

parity

- 使用一個額外硬碟做偶同位運算 — 偶同位
  - 多顆硬碟的二進位資料其對應位置的所有 bit
  - 偶數個 1 為 0
  - 奇數個 1 為 1
- 低成本的可靠度提升選擇
- 掛兩顆硬碟一樣沒救

RAID Redundant Array of Independent Disk 磁碟冗餘陣列

- 0 — striping
- 1 — mirror
- 2
  - 企圖降低 mirror 成本 但只少一個、可靠度又跟 3 一樣而已
  - 雞肋
- 3
  - Bit level striping & parity check
  - 額外多一顆硬碟
- 4
  - Block level striping & parity check
  - 額外多一顆硬碟
- 5
  - Block level striping & parity check
  - parity check Block 分散存於各個硬碟、避免集中使用特定硬碟
  - 額外多一顆硬碟
- 6 — Cost 太高、無實際產品
- RAID 0 + 1
  - 先 0 再 1
  - 壞掉要換整組 mirror
- RAID 1 + 0
  - 先 1 再 0
  - 壞掉把特定 Disk 的 mirror 換上即可

Raw-I/O

- 將磁碟視為大型陣列使用
- 無檔案系統管理、支援
- 存取速度快
- 操作不易
- 通常用於資料庫底層

Bootstrap Loader

- 載入 O.S. 的 Loader
- 存在 ROM 中
- 更改不易
- 其 size 受限於 ROM 硬體 — 現代多以分層方式處理
  - 以小型的 loader 載入完整的 Bootstrap Loader
  - Disk 中劃分特定 Block 作為 Boot Blocks 其中存放完整的 Bootstrap Loader
  - 包含 Boot Blocks 的 partition 稱為 Boot Disk 或 system disk (e.g. C:\)

Bad sector 處理

- Mark — 插旗表示之
- Spare sector
  - 使用備份的、預留的磁區替換之
  - 透過 Device Controller 管理 — 可能破壞磁碟排班之效能 — 分散 Spare sector 於每個 track 或 cylinder 可改善之
- sector slipping
  - 滑動處理
  - 以壞掉磁區的下一個當作該磁區的 spare sector 其後的磁區全部往後移動(滑動)