資工三 穆冠蓁 410705007

1. (a) Combined Scheme 的效能通常很差. 因為其中包含 Double Indivect. 需存取多個 pointer 才能真正讀取到 Data Blocks.

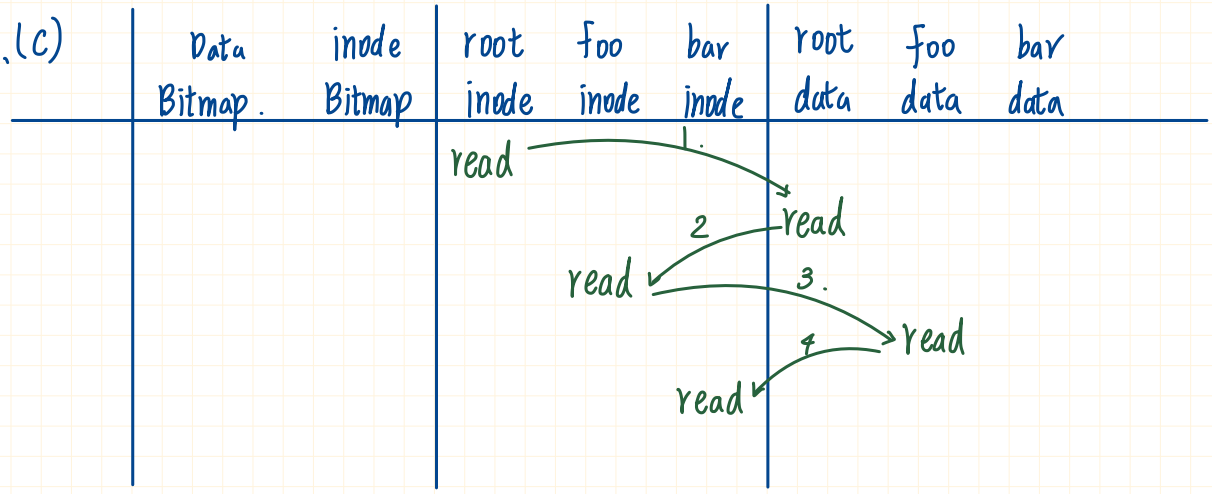
Small Files: 通常使用 direct index. 存取一次 pointer 就可讀 Data Blocks

Large Files: 通常需使用 Indirect index, Double indivect index. 才能存下所有 Data blocks. 相較於 small files 較耗時

(b) Combined Scheme 是為了應付檔案大小不同. 所以讓小檔案使用 direct index. 而大檔案才使用 Indirect index, Double Indirect

2. (a) 在 read, write 前需要先透過 open() 將在 disk 上的 metadata load 進 Cache. 減少每次 read, write 需要到 disk 上存取 metadata. 浪費時間

(b), (c)



1. read root Datablock
2. Find the entry of foo in root Datablock, read foo inode
3. read foo Data block
4. Find the entry of bar in foo Datablock, read bar inode

(c) 5 Disk Blocks,

Root inode → Root Data block → foo inode → foo data → bar inode

(d) 由於需從 root 開始查詢 inode, data, 因此 pathname 越長, 需遍尋更多 directory, I/O 次數也越多

(e) 假如是以 linked list 存 file entries. 若 directory file 裡 file 越多, linked list 就會越長. 搜尋 entry 時間就越長.

3. (a) 3 writes, write inode, write actual data block, write data bitmap

(b) ① Only Data block is written to Disk: FS is consistent (Inode, Data bitmap 一致)
→ 已寫入的 data 會遺失

② Only inode is written to Disk: FS is inconsistent.
→ if trust inode, user would read garbage data

③ Only bitmap is written to Disk: FS is inconsistent.
→ if trust bitmap, 空間浪費, Data Bitmap 認為已經被 allocate (space leak).

(c) 在修改 Data Structure (Inode, Bitmap, Data block) 前, 事先寫 log 記錄接下來要做的事情, 若是發生 system crash,
- 在寫 log 時 crash: Data 未寫入, 但 FS is consistent
- 在寫 data 時 crash: 可以透過 log 修復未寫入的 data → consistent.

4. (a) 由於一個 OS 可能有多個 File system, 因此若沒有 VFS, user 需自行在不同 API 之間切換, 以讀取不同 FS 的 file, 對 user 很麻煩.

(b) user 只需將 read(file) 送到 VFS, 由 VFS 負責判斷 file 在哪顆 HD, 再呼叫對應 FS 的 function 來讀取

5. 先將 data 寫到 memory, 等到資料量夠多, 再用 sequential 的方式, 一次寫入 disk 中, 讓許多小的 random write 合成一個大的 sequential write

6. 這種記錄 directory 的方式, 讓 directory 變成一個也可以存在 data block 的特殊檔案類型, 不需修改結構, 就能支援「目錄」.