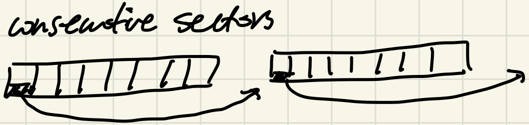
5/13/24

Filesys Designs

Data Layout

→ contiguous, linked

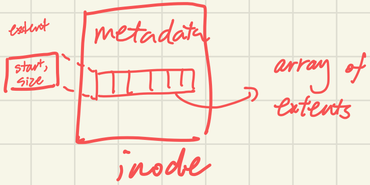
- → can combine these approaches :



- → extent : a contiguous group of sectors / blocks

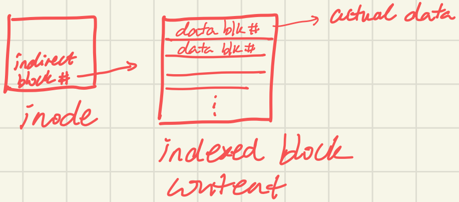
  (any # of consecutive blocks)

  - → track multiple extents via linked approach
  - → can also store an array of extents →



  - → how to find block # given an offset?
    - → find out the right extent first (traverse the array / list)
    - → then find out the relative offset within the extent

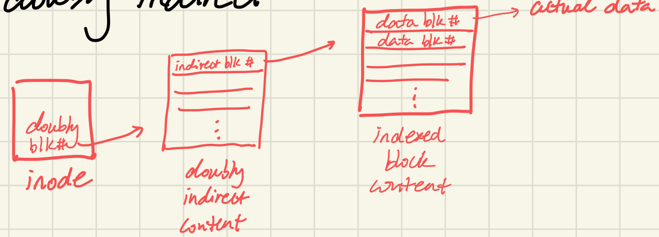
→ indexed / indirection block



disk pointer
= block / sector #

→ multilevel indexed block

→ doubly indirect



→ triply indirect

→ content = array of doubly indirect blk #

→ more indirection ⇒ can track more data, more disk reads to get to actual data blk

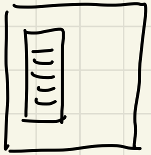
FS Case Study : Fast File System (FFS)

→ Linux ext2, ext3 design

→ designed for fast disk performance
- → reduce random access
- → optimize for small files (90% ↑), support large files

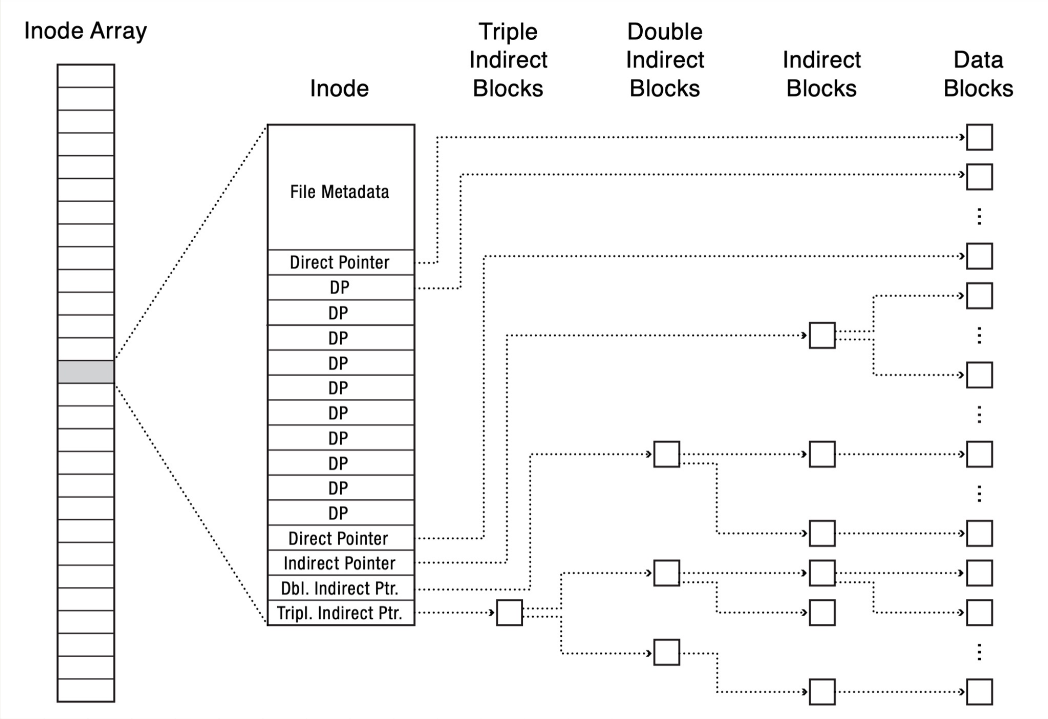
→ FFS data layout

array of disk pointers (blk #)



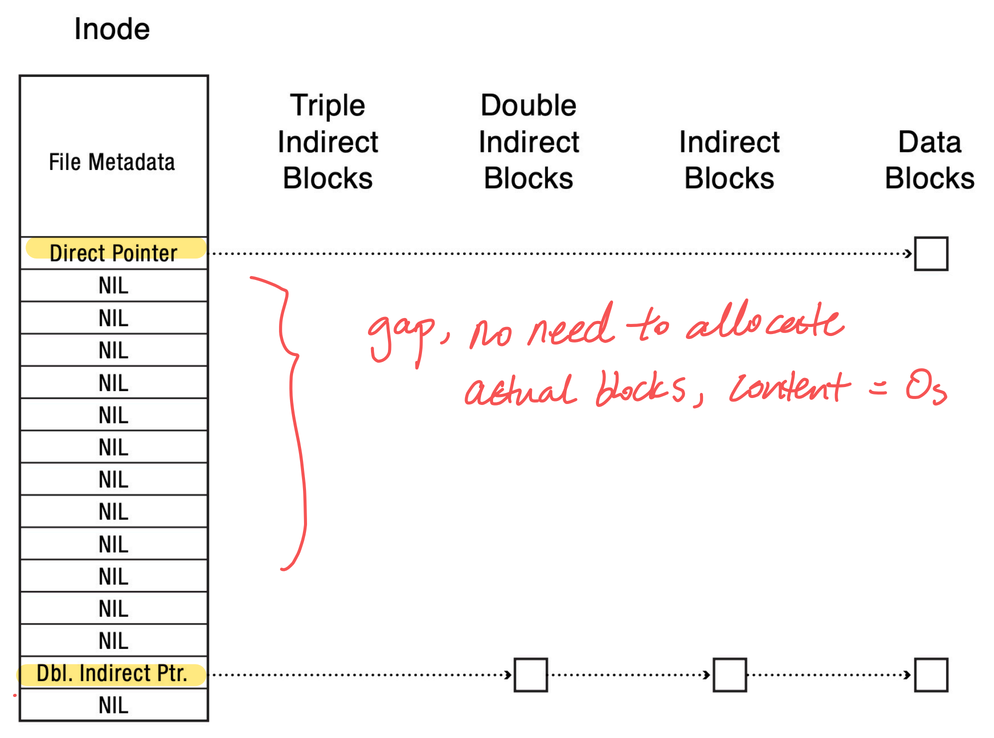
- → block size = 4096 (8 consecutive sectors)
- → 12 direct pointer (points to data blocks) ← small file can fit here
- → 1 indirect pointer
- → 1 doubly indirect
- → 1 triply indirect

FFS



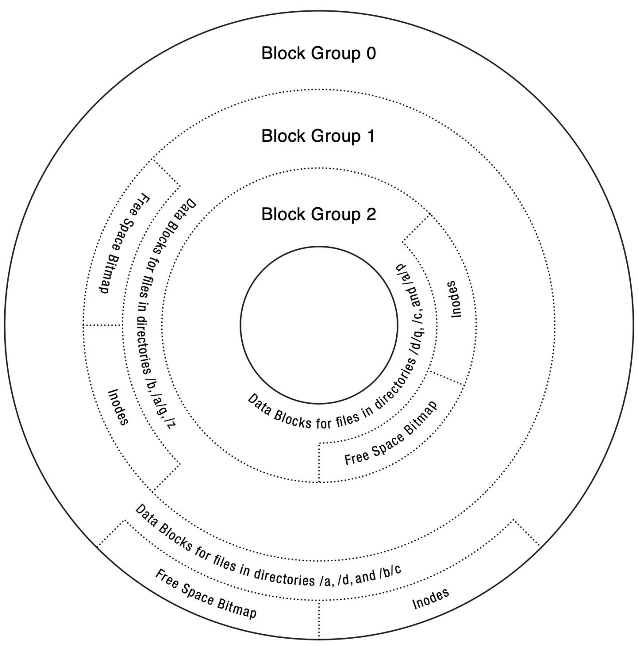
- → file creation :
  - → allocate a new inode (update inode table)
  - → initialize the inode
  - → update parent dir's directory entry & inode
- → write 100 bytes to the newly created file
  - → allocate a block update block bitmap
  - → update inode

- → Gaps in a file:
  - → lseek lets a process sets file offset anywhere (even past end of file)



FFS Locality Heuristics

→ group nearby tracks into block groups


Block Group 0
Block Group 1
Block Group 2
Free Space Bitmap
Data Blocks for files in directories /b, /a/g, /z
Inodes
Inodes
Data Blocks for files in directories /d/q, /c, and /a/p
Free Space Bitmap
Data Blocks for files in directories /a, /d, and /b/c
Free Space Bitmap
Inodes

accessed together
① metadata & data
② files within the same dir

→ place related things into the same block group

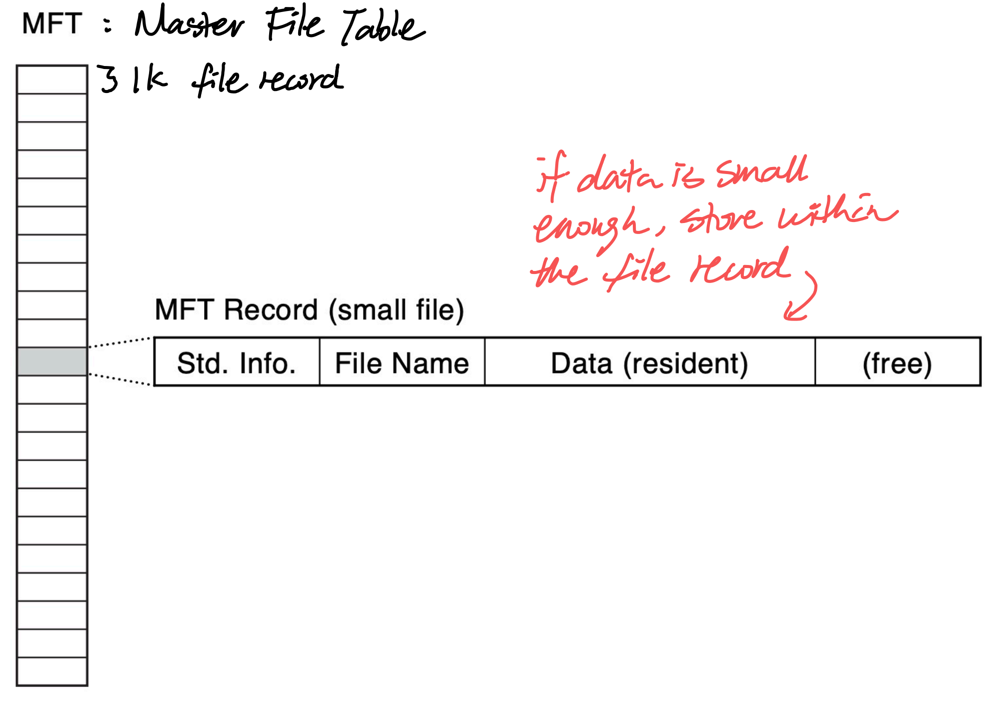
→ place unrelated files into different block groups

↳ different dirs, files w/in different dirs

→ what about large files?

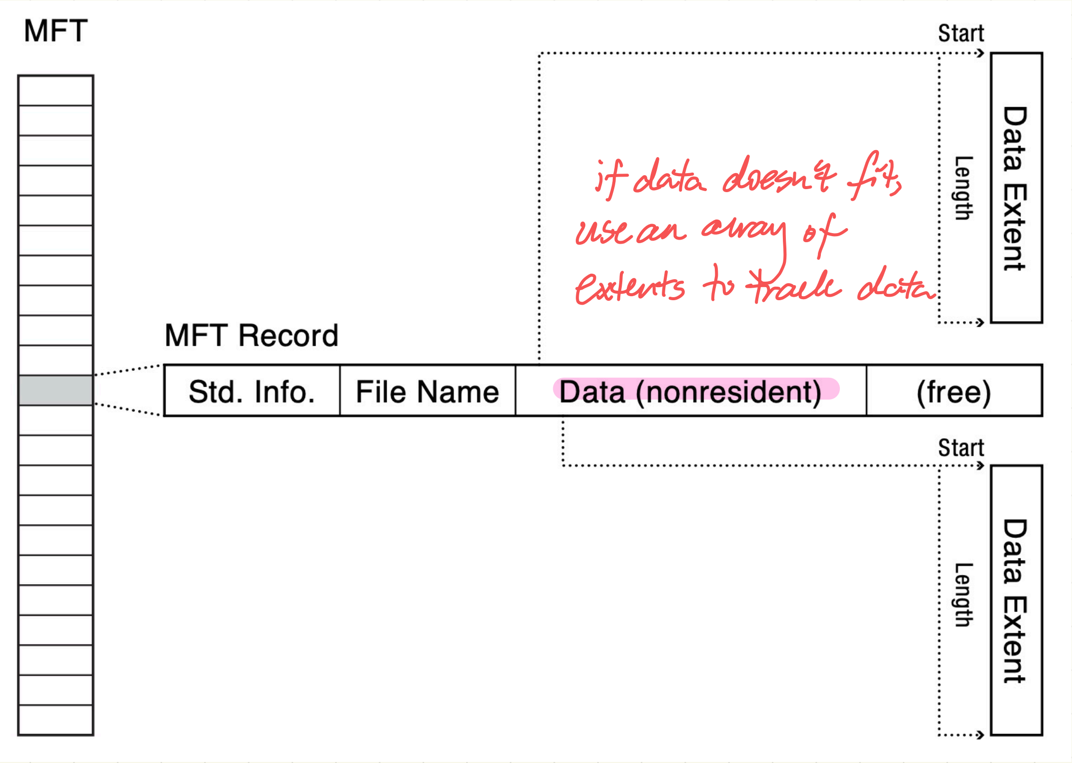
→ spread large files' data across block groups to avoid large files taking over a single block group

Case Study: New Technology File System (NTFS)   Windows FS

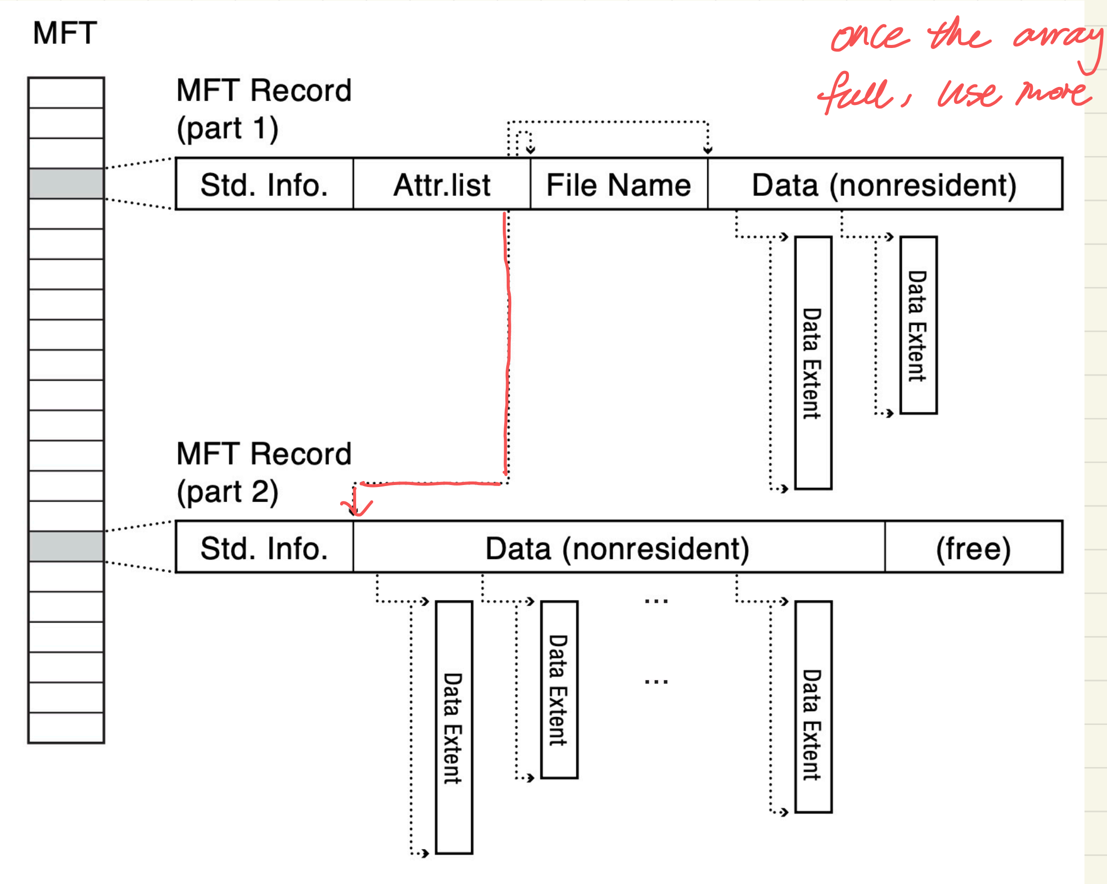


if data is small enough, store within the file record

can cache the entire MFT in memory



if data doesn't fit, use an array of extents to track data



once the array of extents is full, use more file records to store more array of extents

(linked list of array of extents)