Lec 21

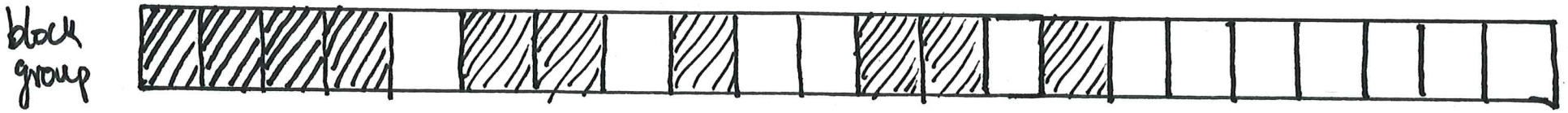
NTFS
+
Transactional Storage

FFS Locality

- Block group: set of nearby cylinders
  - locate files from same dir. in same group
- partition inode t[illegible]roups
- first fit allocation

Xpar first fit
write two block file
"e large file

21.1



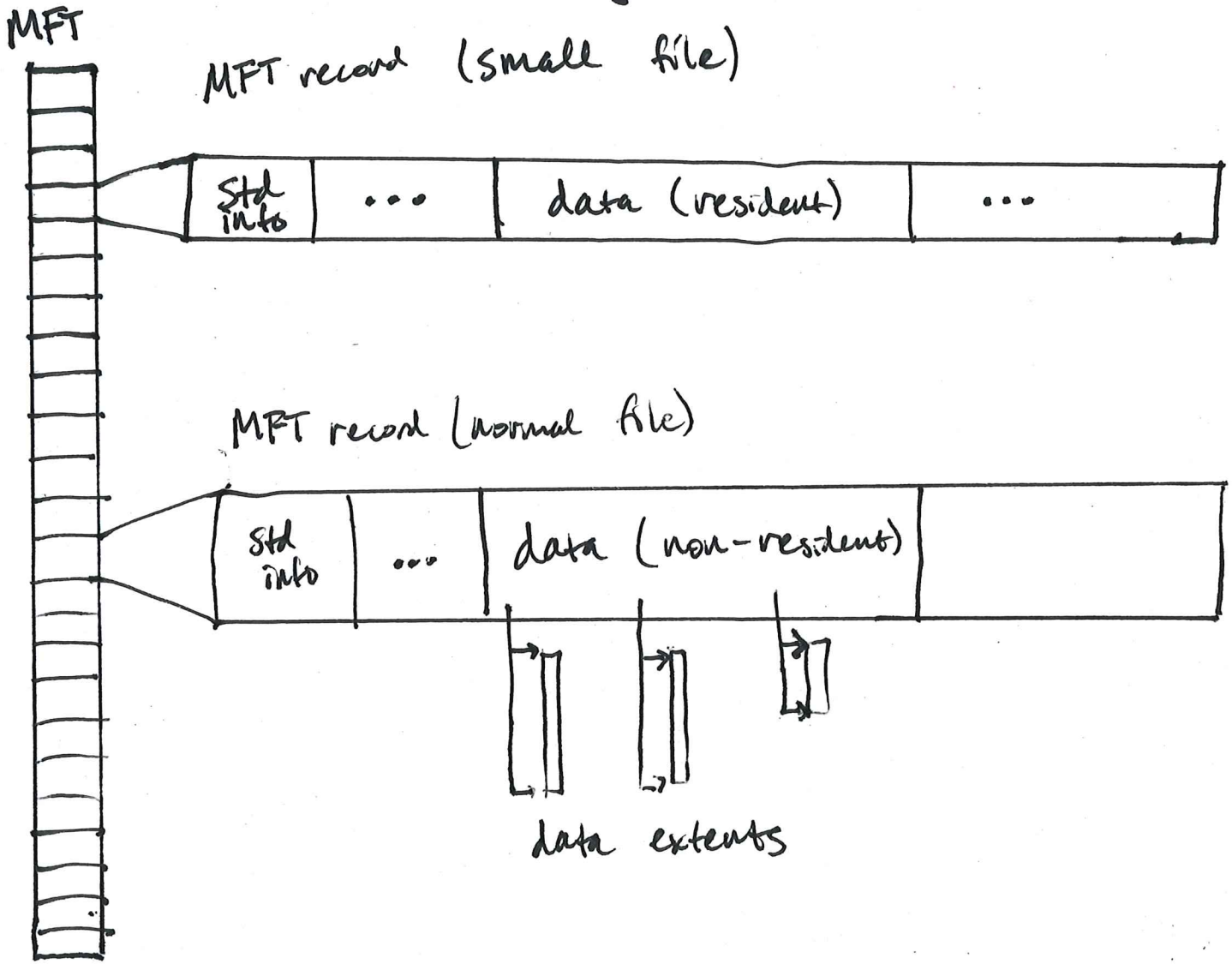
NTFS New Technology File System

- Extent: a contiguous region of the disk (start, length)

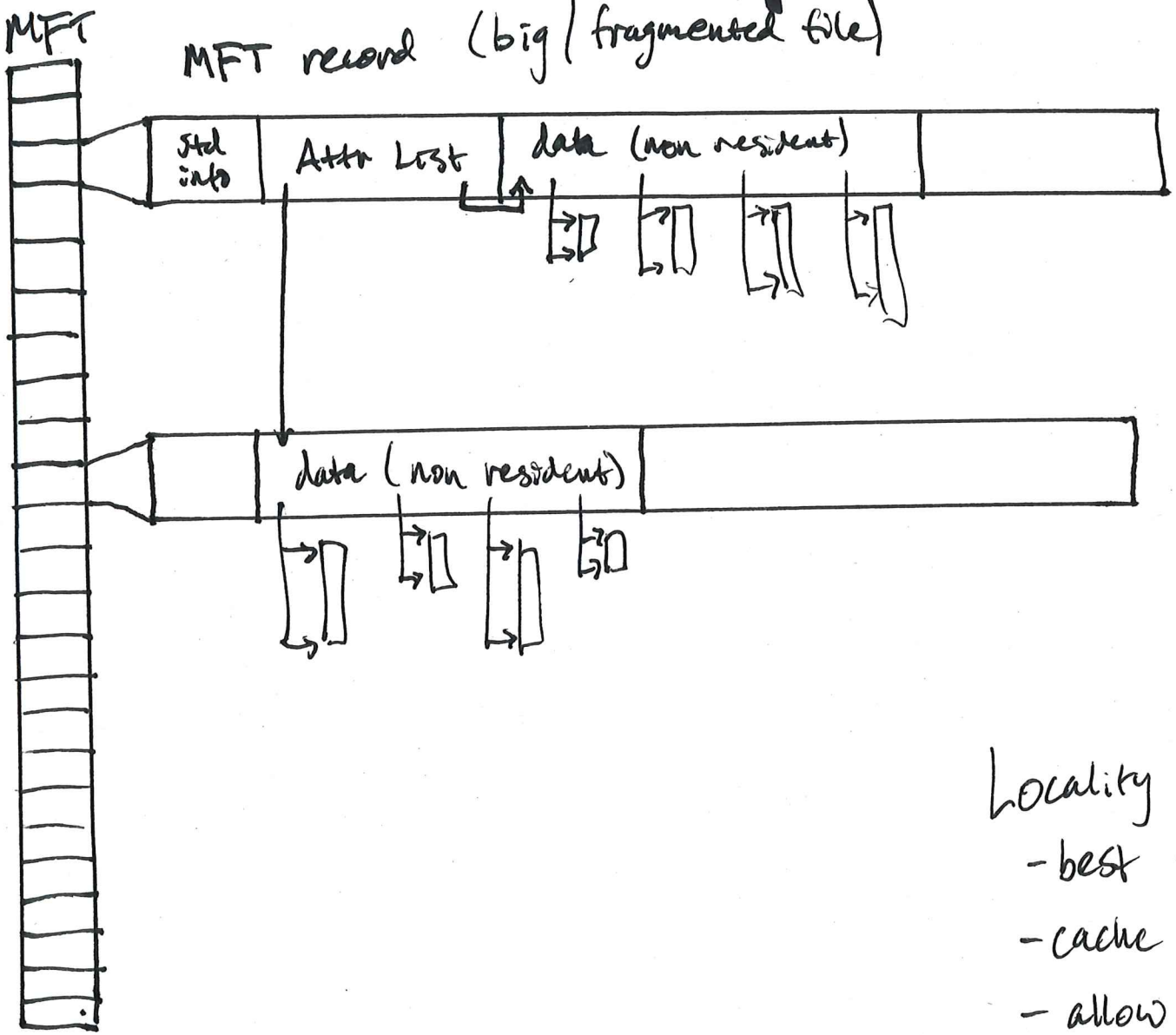
- flexible tree of extents represents file data

- Master file table (MFT)
  - similar to FAT | inode table: array of fixed size records
  - each entry is 1kb and stores attributes
    - attribute can be resident (in entry) or non-resident (extents)

NTFS example



NTFS example

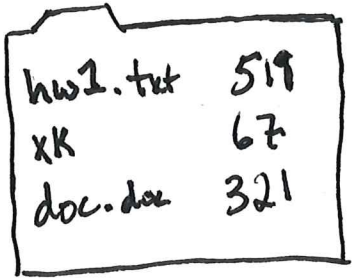


Locality heuristics:

- best fit
- cache only part of allocation map
- allow user to declare file size

Directories

- directories are files
  - but their content is understood by the FS



- directories map names to file numbers
  - including subdirectories → recursive traversal
- FFS: linked list of (name, number) pairs
- NTFS: B-tree of (name, number) pairs

Reading a file

- Assume Unix FFS. App wants to read /a/b/c

1. read root directory "/", with pre-arranged inumber
   - inode array at fixed block address: read the block for #2
2. scan the data for "/" using direct/indirect pointers; look for name = "a"
3. read the corresponding inumber for "a", say 123
4. scan data for inode 123 looking for "b"
5. read inode for b, say 332
6. scan data for inode 332 looking for "c"
7. read inode for c, say 451. (this is a file, not a directory)
8. scan/return data for inode 451

Reliable Storage

- disks can fail (completely or partially)
  - ↳ RAID, backup, replication
- power can go out at inopportune time
  - in the middle of a write
    - to user data file (bad)
    - to file system data structure (very bad)

- users want durability
  - previously completed writes can be retrieved

Challenge of reliable Storage

- single logical FS op → multiple physical ops
  - to inode, to indivect block, to data block, ...
- physical device processes concurrent requests
  - mag disk: schedule to minimize seeks
  - flash: parallel lanes
- what happens when power goes out?
  - device may have capacitors to complete some ops
- how can we keep our data consistent?

Transactions

- logical group of operations

Atomic: all ops complete or none do
  - challenge: physical ops atomic one-by-one

Durable: completed ops not lost

Isolated: concurrent txns appear sequential

Consistent: data invariants preserved

Approach #1: Careful Ordering

- design order of operations so that crash at any point can be handled

- when rebooting after crash, fix any problems

- typical approach used until ~1990s

  FAT, FFS ...

FAT append to file

- allocate data block
- write data
- write new MFT entry for data block
- update tail pointer to new MFT entry
- update access time at root MFT

Recovery
- scan MFT, deallocate unlinked entries
- update access times

21.11