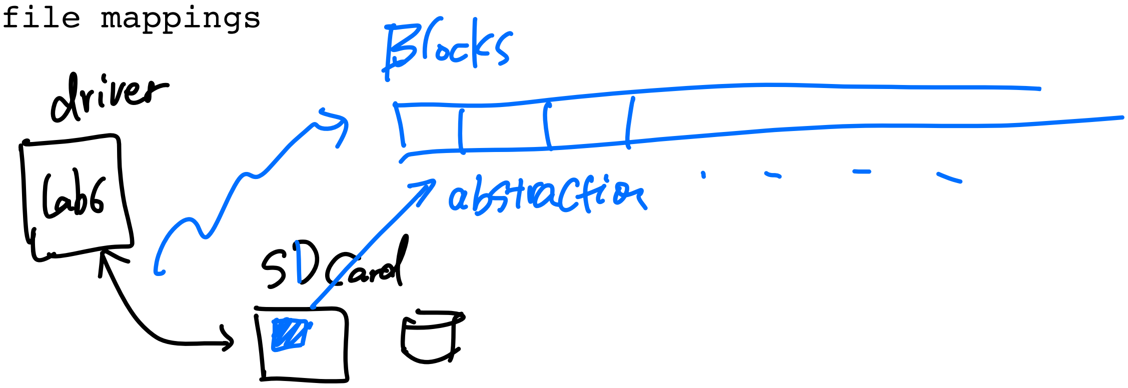
week 8b
CS6640
02/27 2026
https://naizhengtan.github.io/26spring/

☑ 1. intro to fs
☑ 2. Unix files
□ 3. what about now?
  -- PMem introduction
  -- modern(?) file mappings



1. Intro to file systems

Q: what does a FS do?

- org data
- write code → .c → binary
- file < dev
- abstraction of storage
- permissions

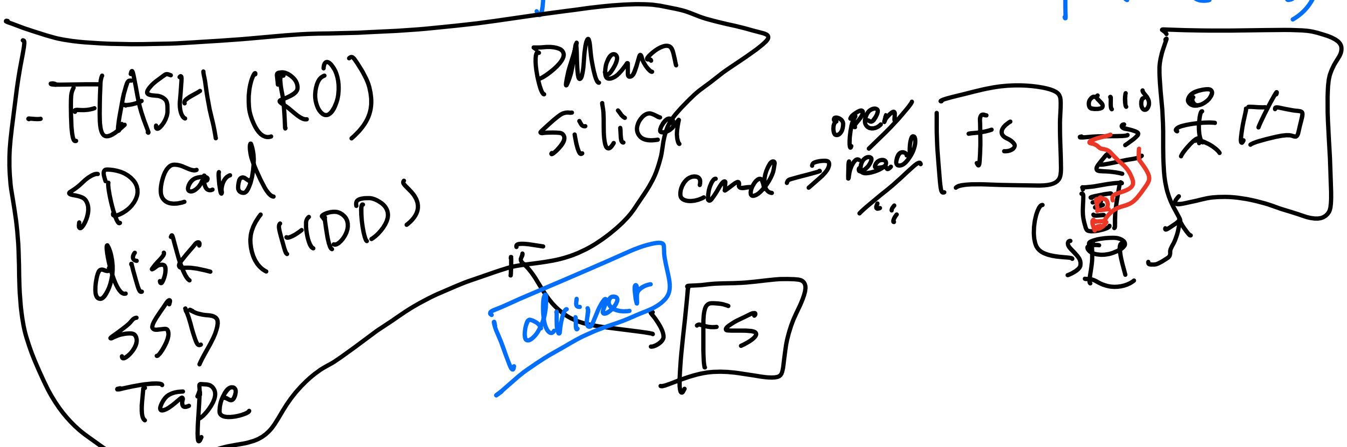
① persistency

HW

② name a set of bytes on disk (file)

③ map from human-friendly-names to file (dirs)

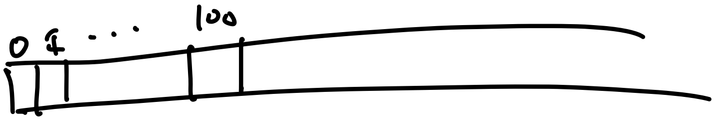
OS/SW



2. Files

Q: what is a file?

- user : a seq of bytes



- OS : a set of blocks on disk

Q: what does a file do?

file-mapping : $\underline{\langle \overset{\text{"/tmp/labs"}}{file}, \overset{100}{offset} \rangle} \xrightarrow{\text{"file"}}$ disk addr

inode

$\langle Proc, VA \rangle \xrightarrow{PT} PA$

① array

② linked-list

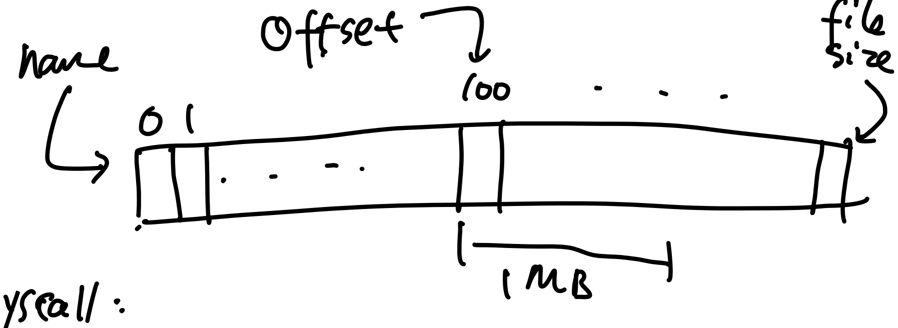
③ tree → Btree

→ imbalanced tree

④ hash map

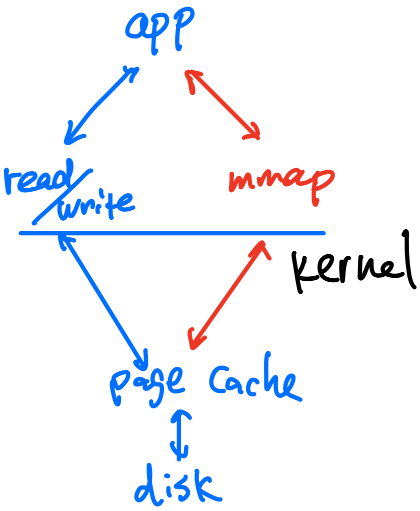
⑤ Neural Net

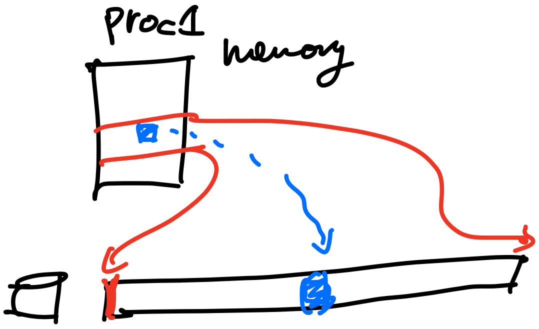
user's view of a file:

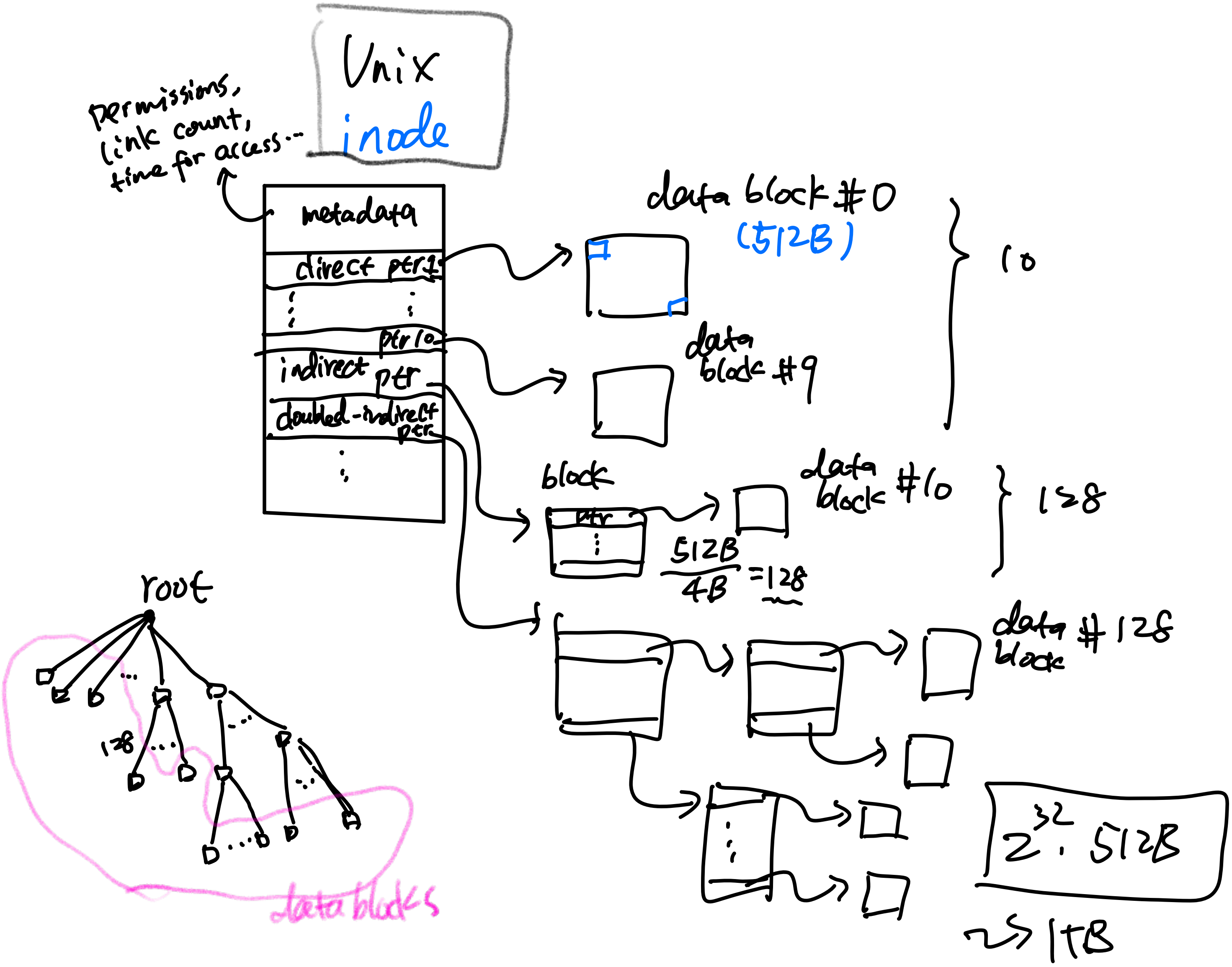


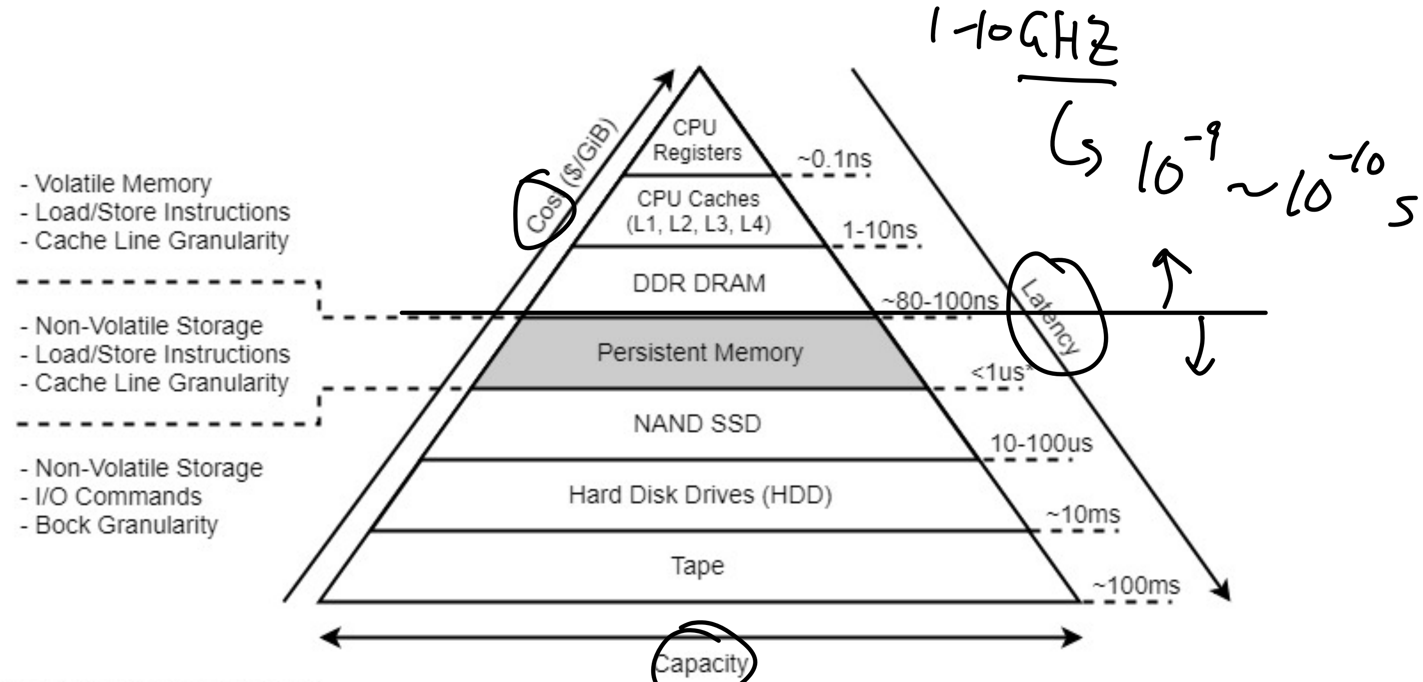
syscall:

open/close/read/write/lseek ...






Unix
inode
permissions,
link count,
time for access..
metadata
direct ptr1
ptr10
indirect ptr
doubled-indirect ptr
data block #0
(512B)
10
data block #9
block
ptr
data block #10
128
512B / 4B = 128
data block #128
$2^{32} \cdot 512B$
~> 1TB
root
128
data blocks



(*) See vendor specifications

Figures borrowed from "PMDK Introduction" https://docs.pmem.io/persistent-memory/getting-started-guide/what-is-pmdk

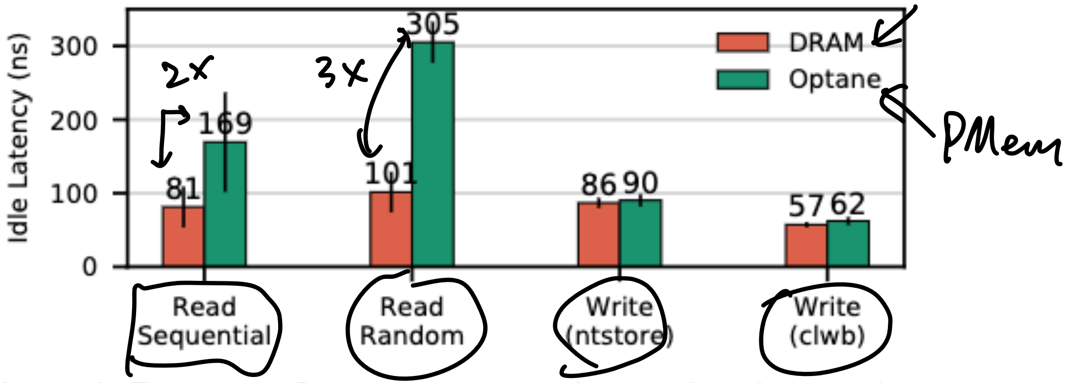


Figure 2: **Best-case latency** An experiment showing random and sequential read latency, as well as write latency using cached write with `clwb` and `ntstore` instructions. Error bars show one standard deviation.

Above figures are borrowed from An Empirical Guide to the Behavior and Use of Scalable Persistent Memory

* FS design parameters:

  - small files (most files are small)
    vs.
    large files (much of the disk is allocated to large files)

  - sequential access vs. random accesses

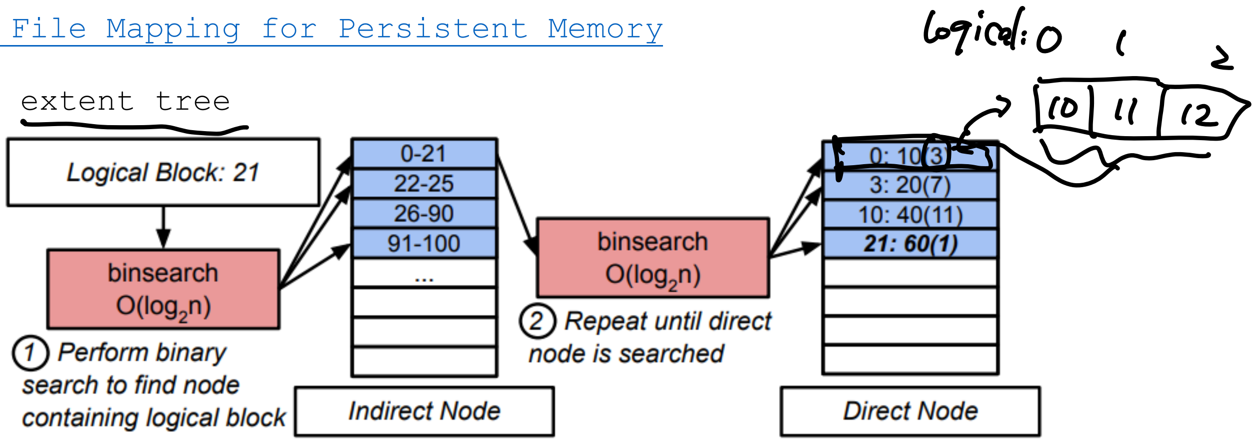
  - prefetching

  - disk utilization (metadata overhead and fragmentation)

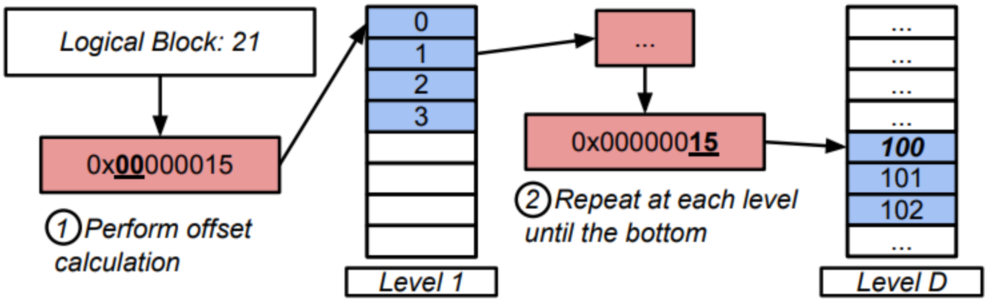
* Q: If you were to design a fs, what's your file-mapping structure?

Rethinking File Mapping for Persistent Memory

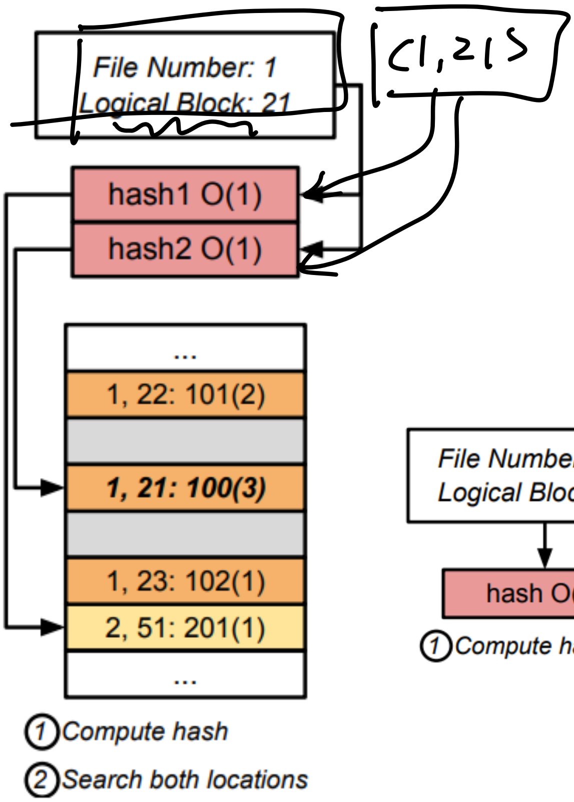
1. Per-file extent tree



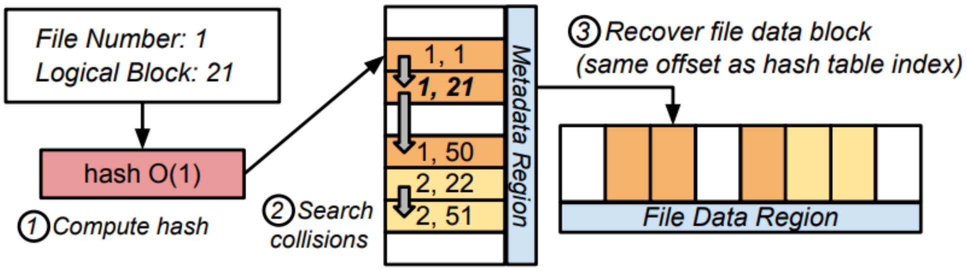
2. Per-file radix tree



3. Global cuckoo hash table



4. Global hash table (HashFS)


File Number: 1
Logical Block: 21
hash O(1)
① Compute hash
② Search collisions
1, 1
1, 21
1, 50
2, 22
2, 51
Metadata Region
③ Recover file data block (same offset as hash table index)
File Data Region