PERSISTENCE: SOLID-STATE DEVICES

Vojtech Aschenbrenner

(Instead of Shivaram Venkataraman)

CS 537, Fall 2024

ANNOUNCEMENT #1

Project 6 is out!

Deadline 1: Nov 27th, no slip days applicable

Deadline 2: Dec 6th, slip days applicable

Just 6 days left (6*24*2/3 = 96 hours)!!

START TODAY!

ANNOUNCEMENT #2

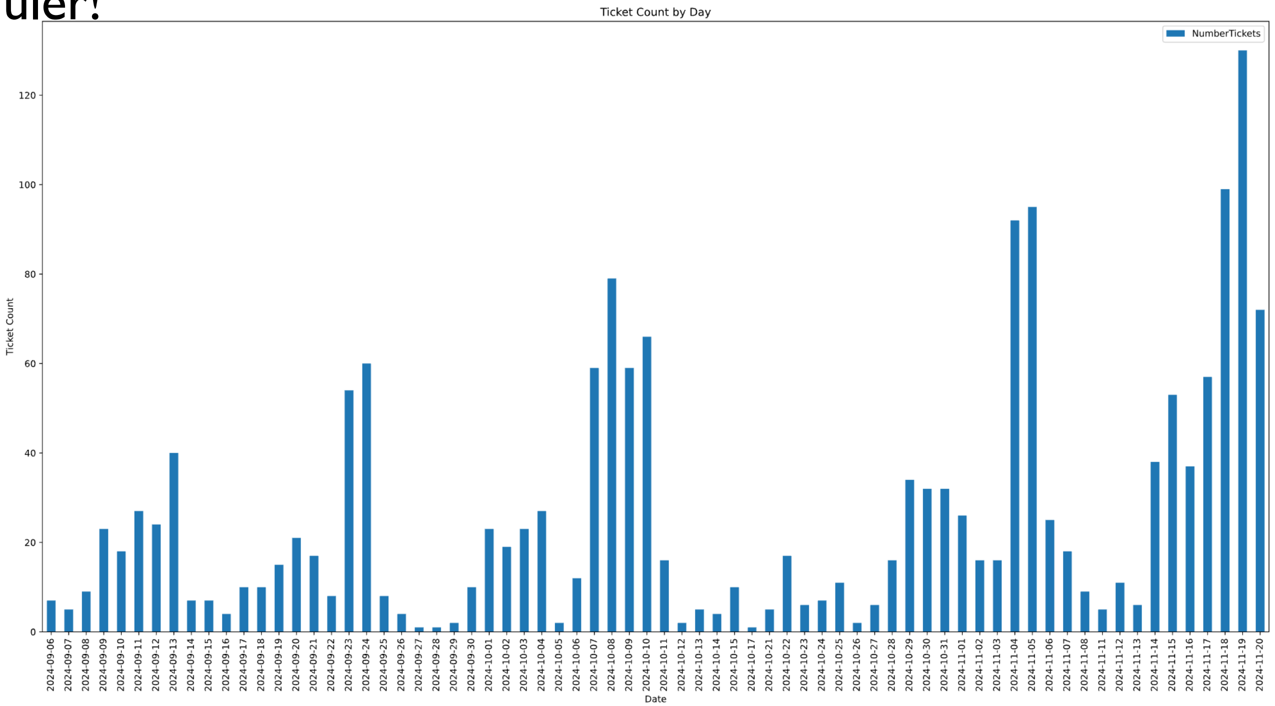
Overcrowded office-hours sessions?

Sounds like a bad scheduler!

What is the solution?

How to make it worse?

How to make it better?



AGENDA / LEARNING OUTCOMES

SSD

RECAP

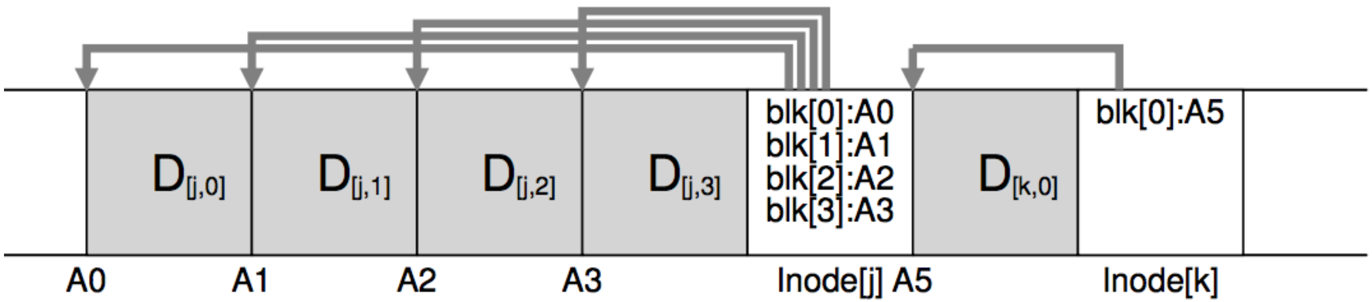
LFS STRATEGY

File system buffers writes in main memory until "enough" data

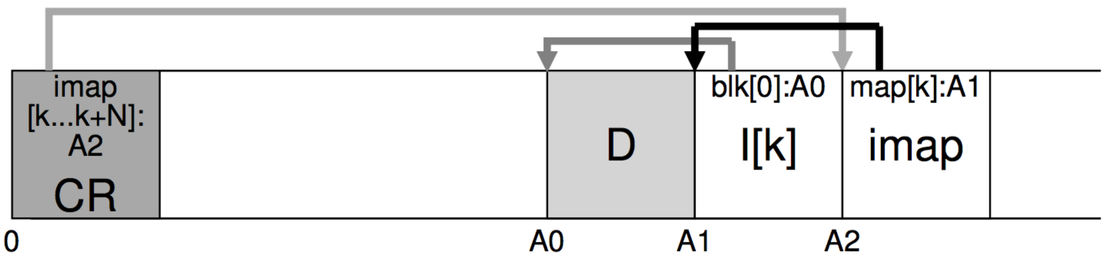
- Enough to get good sequential bandwidth from disk (MB)

Write buffered data sequentially to new **segment** on disk

Never overwrite old info: old copies left behind

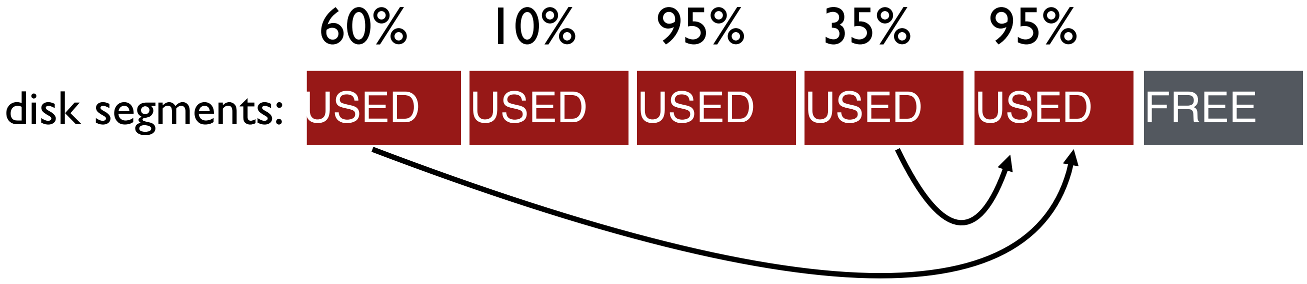


READING IN LFS



1. Read the Checkpoint region
2. Read all imap parts, cache in mem
3. To read a file:
   1. Lookup inode location in imap
   2. Read inode
   3. Read the file block

GARBAGE COLLECTION



compact 2 segments to one

When moving data blocks, copy new inode to point to it

When move inode, update imap to point to it

SEGMENT SUMMARY

Is an inode the latest version?

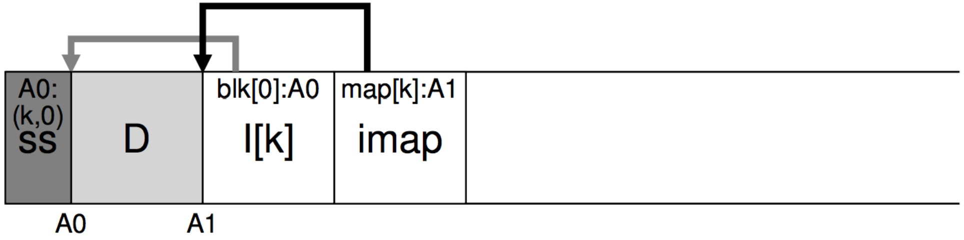
Check imap to see if this inode is pointed to

Fast!

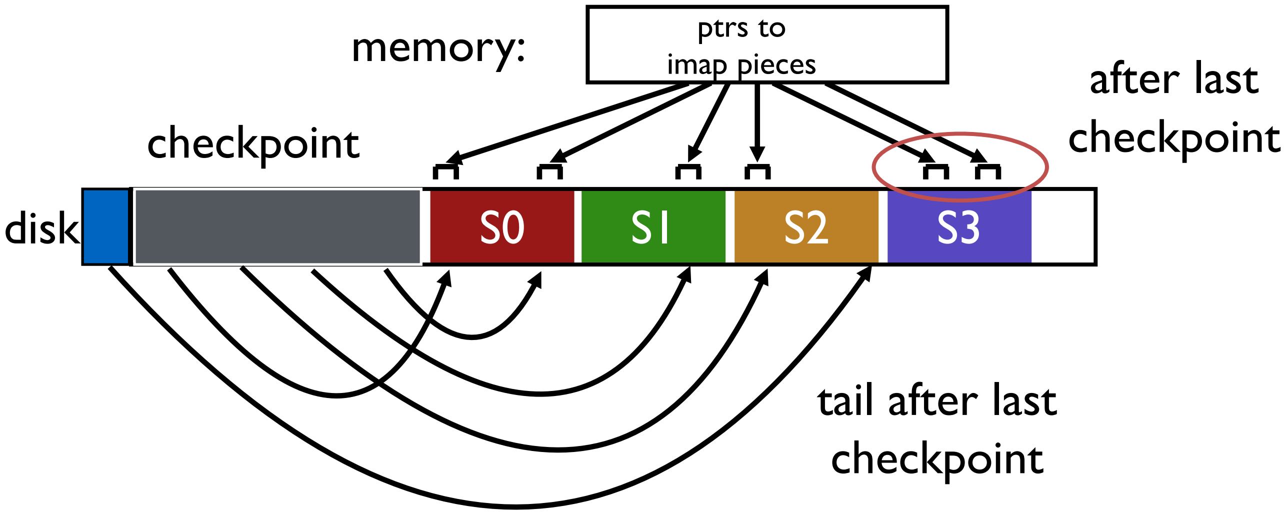
```
(N, T) = SegmentSummary[X];

inode = Read(imap[N]);

if (inode[T] == X)
    // block D is alive
else
    // block D is garbage
```



CRASH RECOVERY



CHECKPOINT SUMMARY

Checkpoint occasionally (e.g., every 30s) Or?

Upon recovery:

- read checkpoint to find most imap pointers and segment tail
- find rest of imap pointers by reading past tail

What if crash <u>during</u> checkpoint?

CHECKPOINT STRATEGY

Have two checkpoint regions

Only overwrite one checkpoint at a time

Use checksum/timestamps to identify newest checkpoint

QUIZ 19

https://tinyurl.com/cs537-fa24-q19

LFS VS FFS

File System Logging Versus Clustering: A Performance Comparison

Margo Seltzer, Keith A. Smith
Harvard University

Hari Balakrishnan, Jacqueline Chang, Sara McMains, Venkata Padmanabhan
University of California, Berkeley

A Critique of Seltzer's LFS Measurements

John Ousterhout / john.ousterhout@scriptics.com

Until ... SSDs enter the picture

SSDS

NAND FLASH

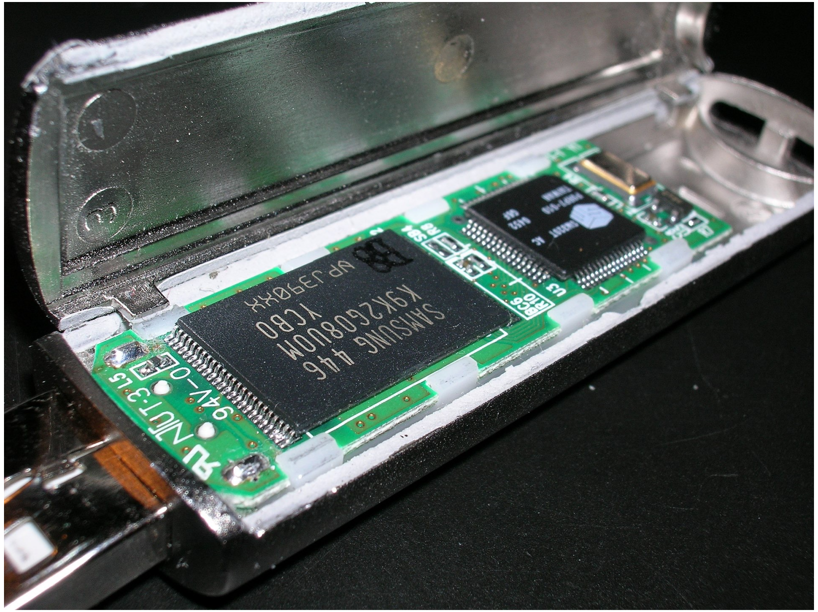
Single Level Cell (SLC) = 1 bit per cell

Multi Level Cell (MLC) = 2 bits per cell

Triple Level Cell (TLC) = 3 bits per cell

Quad Level Cell (QLC) = 4 bits per cell

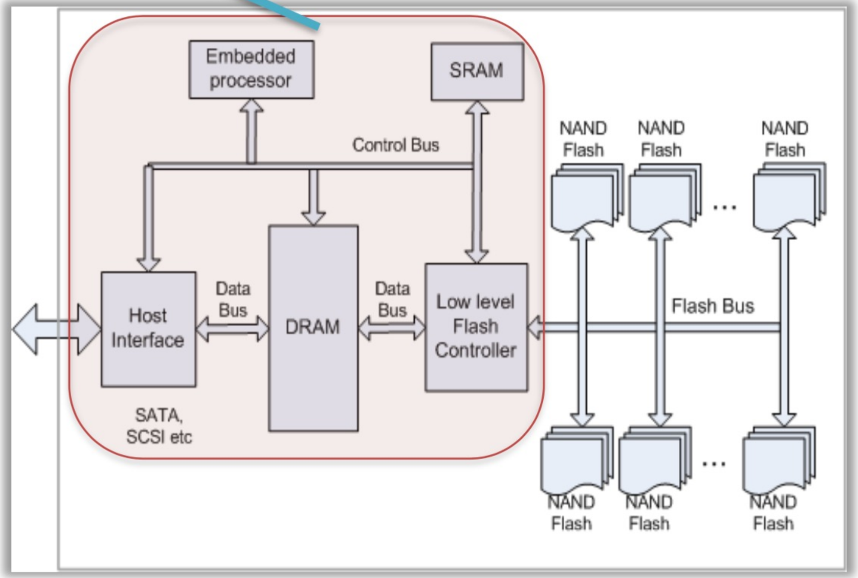
(Penta Level Cell (PLC) = 5 bits per cell)



SSD STRUCTURE

What does it remind to you?

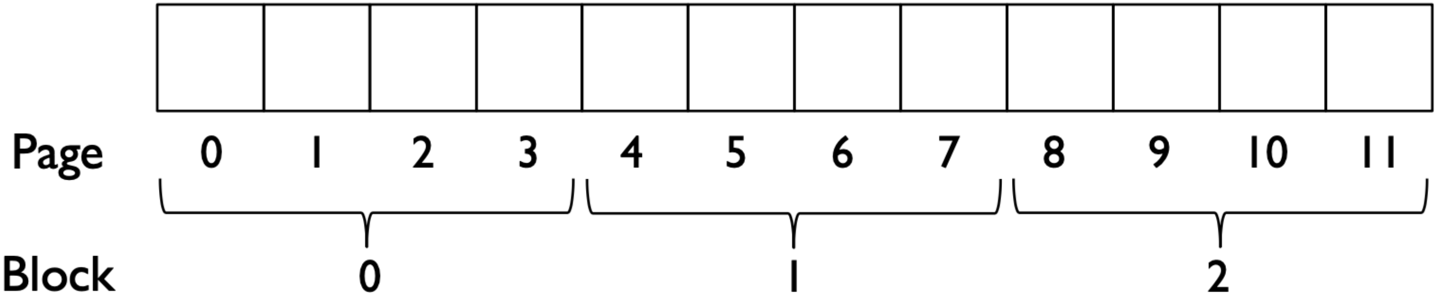
Flash Translation Layer
(Proprietary firmware)



Simplified block diagram of an SSD

Massively Parallel
-
New Interface (NVMe)
-
Extensible

SSD PROPERTIES



Page ~ 4KB,
Block ~ 128 KB
or 256 KB

Read

Write

Failures: Block likely to fail after a certain number of P/E cycles (~10,000 for MLC flash, ~100,000 for SLC flash)

SSD OPERATIONS

Read a page: Retrieve contents of entire page (e.g., 4 KB)

- Cost: 25 (SLC), 50 (MLC), 75 (TLC) microseconds
- Independent of page number, prior request offsets

Erase a block: Resets each page in the block to all 1s

- Cost: 1.5 (SLC), 3 (MLC). 4.5 (TLC) milliseconds
- Much more expensive than reading!
- Allows each page to be written

Program (i.e., write) a page: Change selected 1s to 0s

- Cost is 250 (SLC), 750 (MLC), 1100 (TLC) microseconds
- Faster than erasing a block, but slower than reading a page

FLASH TRANSLATION LAYER

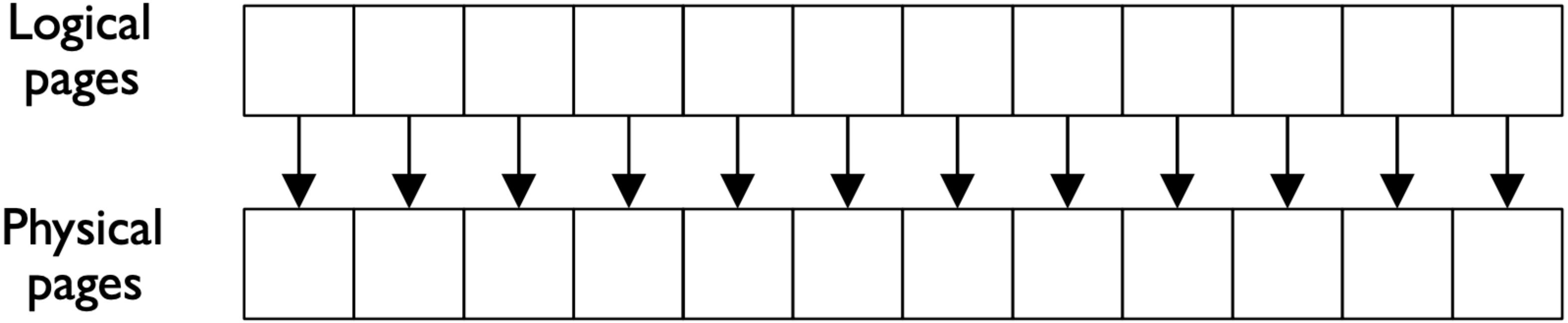
1. Translate reads/writes to logical blocks into reads/erases/programs on physical blocks

2. Reduce write amplification (extra copying needed to deal with block-level erases)

3. Implement wear leveling (distribute writes equally to all blocks)

Typically implemented in hardware in the SSD, but in software for ZNS SSDs (interface?)

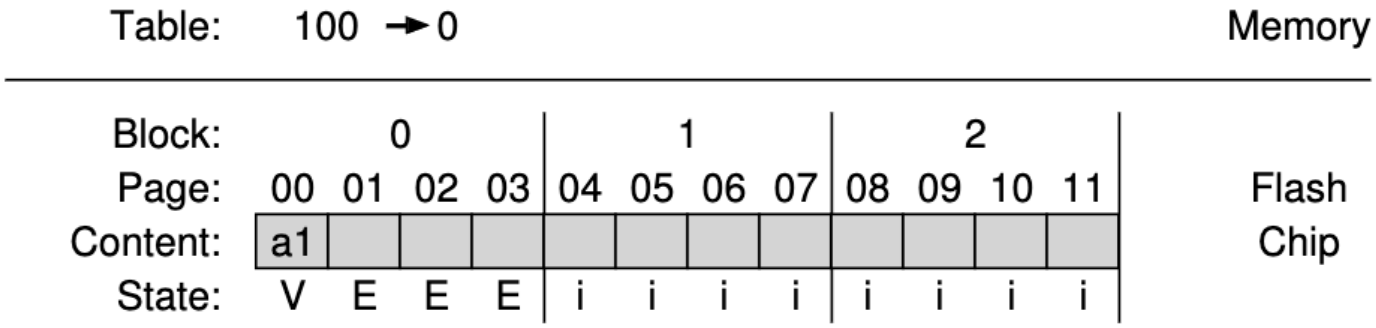
FTL: DIRECT MAPPING



Cons?

FTL: LOG-BASED MAPPING

Idea: Treat the physical blocks like a log



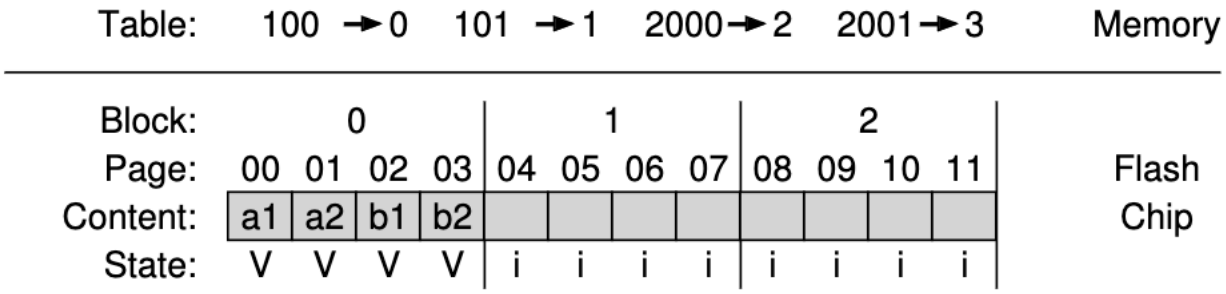
FTL: LOG-STRUCTURED ADVANTAGES

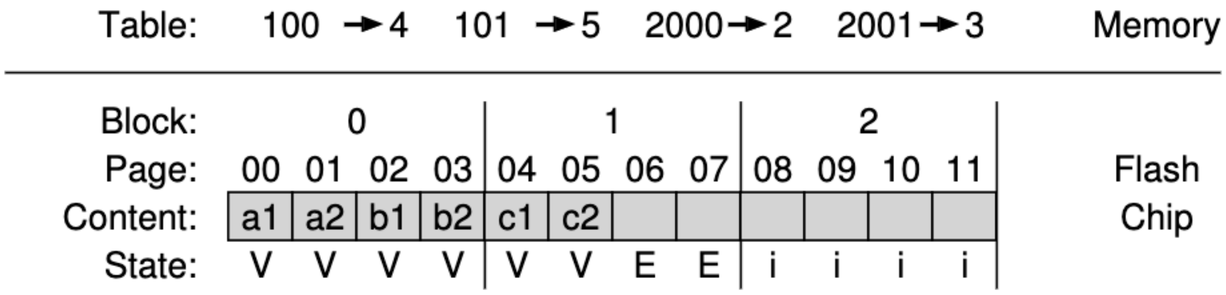
Avoids expensive read-modify-write behavior

Better wear levelling: writes get spread across pages,
even if there is spatial locality in writes at logical level

Challenges? Garbage!

GARBAGE COLLECTION





GARBAGE COLLECTION

Steps:

Read all pages in physical block

Write out the alive entries to the end of the log

Erase block (freeing it for later use)

How does SSD know about rm?

Table: 100 → 4   101 → 5   2000 → 2   2001 → 3   Memory

Block:	0				1				2				
Page:	00	01	02	03	04	05	06	07	08	09	10	11	Flash
Content:	a1	a2	b1	b2	c1	c2							Chip
State:	V	V	V	V	V	V	E	E	i	i	i	i	

Table: 100 → 4   101 → 5   2000 → 6   2001 → 7   Memory

Block:	0				1				2				
Page:	00	01	02	03	04	05	06	07	08	09	10	11	Flash
Content:					c1	c2	b1	b2					Chip
State:	E	E	E	E	V	V	V	V	i	i	i	i	

OVERHEADS

Garbage collection requires extra read+write traffic

Overprovisioning makes GC less painful

- SSD exposes logical space that is smaller than the physical space

- By keeping extra, "hidden" pages around, the SSD tries to defer GC to a background task (thus removing GC from critical path of a write)

Occasionally shuffle live (i.e., non-garbage) blocks that never get overwritten

- Enforces wear levelling

INTERFACE CHANGES

Complex software in SSD firmware requiring powerful CPU and RAM

~50% of the SSD price is not related to storage medium (NAND chips)

TRIM: Mark region on SSD as unused. Used by FS in every OS today after ____?

ZNS (Zoned Namespace) SSDs: Big Sequential writes only. Why is it better?

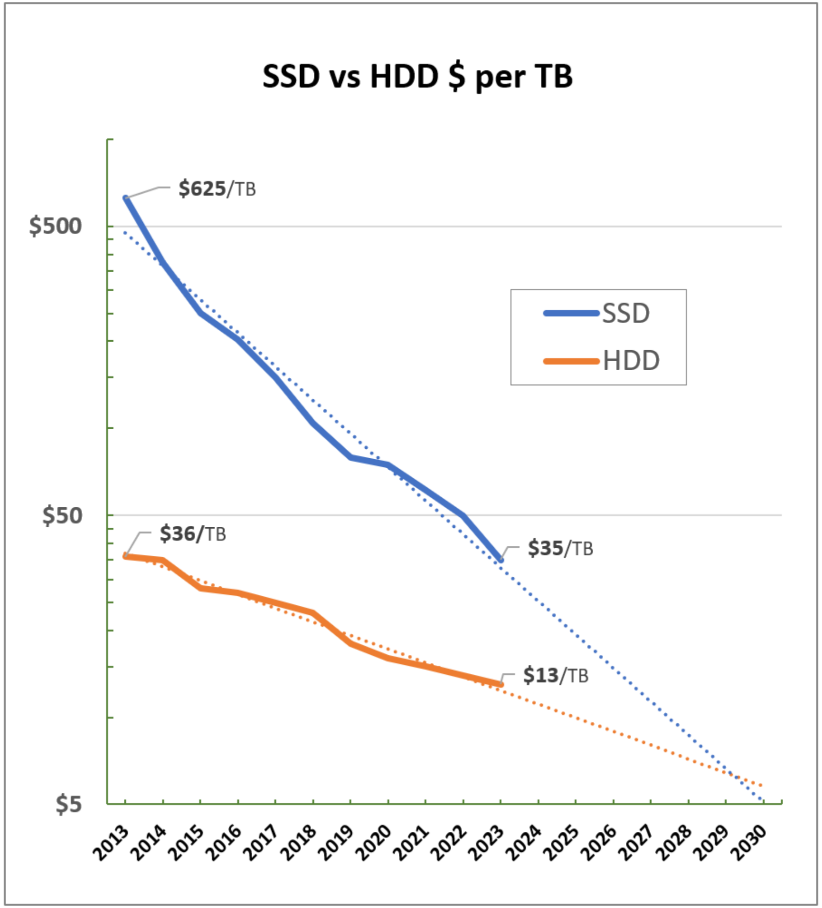
OVERALL PERFORMANCE

Device	Random Reads (MB/s)	Random Writes (MB/s)	Sequential Reads (MB/s)	Sequential Writes (MB/s)
Samsung 840 Pro SSD	103	287	421	384
Seagate 600 SSD	84	252	424	374
Intel SSD 335 SSD	39	222	344	354
Seagate Savvio 15K.3 HDD	2	2	223	223

Samsung 990 Pro (PCIe 4, 30us)	2,277	1,855	3,190	4,857
Crucial T705 (PCIe 5, 30us)	2,374	1,836	7,702	8,576

DDR5-4800 2x32GB (80ns) Read: 74,518 Write: 71,872

COST?



Not just about the drive price!

Power, Reliability, Physical Space, Cooling…

SSDs allow massive NAND arrays with a single FTL. Makes it extremely cheap.

HDDs cannot be modified to have for example multiple spindles with just single head etc.

Pure Storage provides only NAND-based storage solutions today.

NEXT STEPS

Next class: Distributed Systems!