CISC 3320

Mass Storage: NVM and Others

Hui Chen

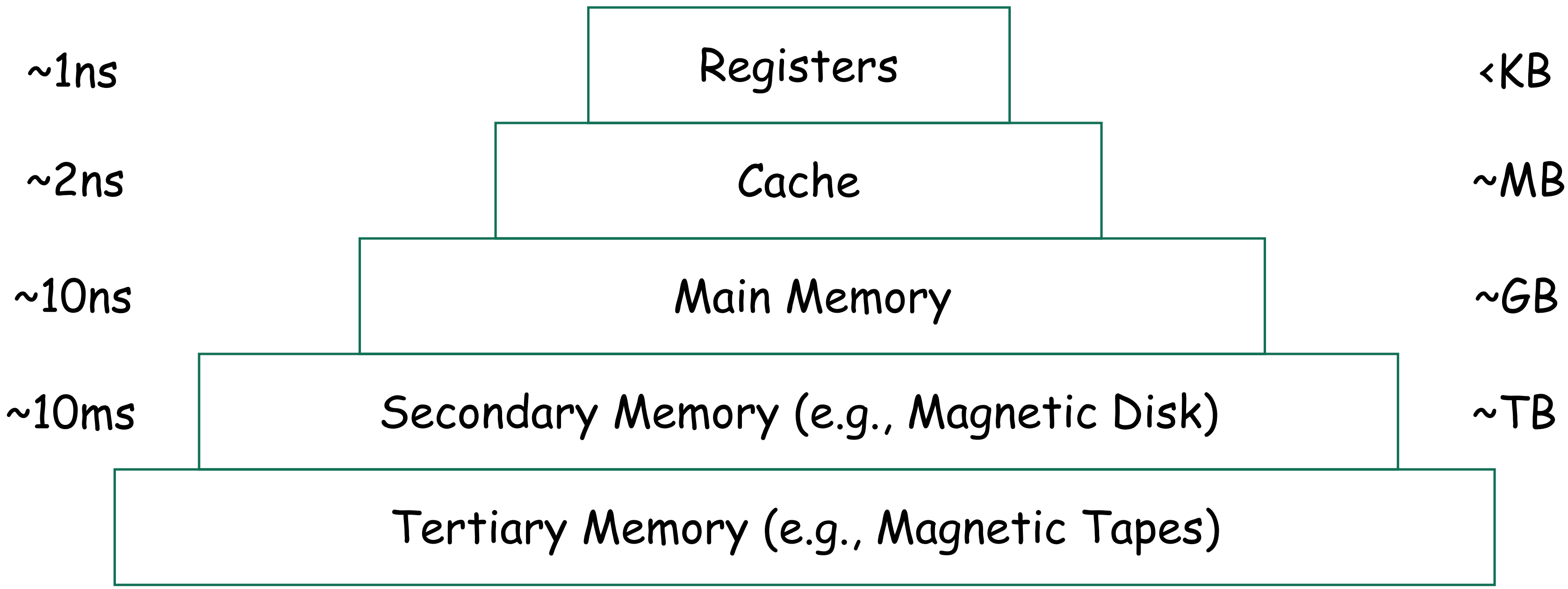
Department of Computer & Information Science

CUNY Brooklyn College

Acknowledgement

- These slides are a revision of the slides provided by the authors of the textbook via the publisher of the textbook

Memory Hierarchy



Outline

- NVM Scheduling
- Volatile memory and tape drives

- Error Detection and Correction
- Storage Device Management
- Swap-Space Management
- Storage Attachment
- RAID Structure

Overview of Mass Storage Devices

- Bulk of secondary storage for modern computers are
  - Hard disk drives (HDDs)
  - Nonvolatile memory (NVM) devices
- Frequently used as a mass storage device
  - Volatile memory
- Once used as a secondary storage
  - Magnetic tapes

Nonvolatile Memory Devices

NVM Devices Characteristics

- Have different forms
  - If disk-drive like, then called **solid-state disks** (**SSD**s)
  - Other forms include **USB drives** (thumb drive, flash drive), DRAM disk replacements, surface-mounted on motherboards, and main storage in devices like smartphones
- Can be more reliable than HDDs
- Maybe have shorter life span – need careful management
- Less capacity
- But much faster
- Buses can be too slow -> connect directly to PCI (e.g., NVM Express, i.e., NVMe)
- No moving parts, so no seek time or rotational latency

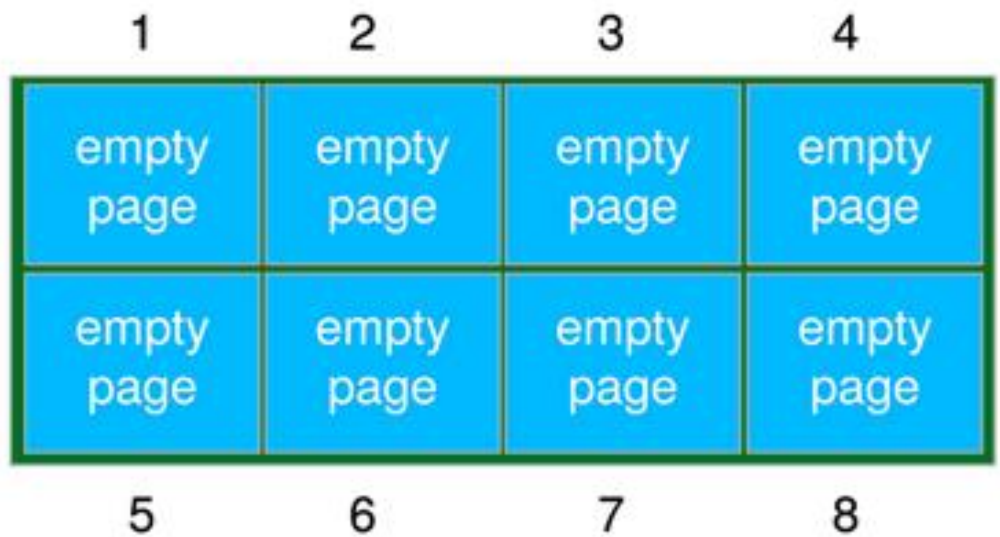
Issues with NVM Devices

- Have characteristics that present challenges
- Read and written in “page” increments (akin to sector) but can’t overwrite in place
- To overwrite, must first be erased, and erases happen in larger ”block” increments
- Can only be erased a limited number of times before worn out (~ 100,000 times)
- Life span measured in **drive writes per day** (**DWPD**)
    - A 1TB NAND drive with rating of 5DWPD is expected to have 5TB per day written within warrantee period without failing

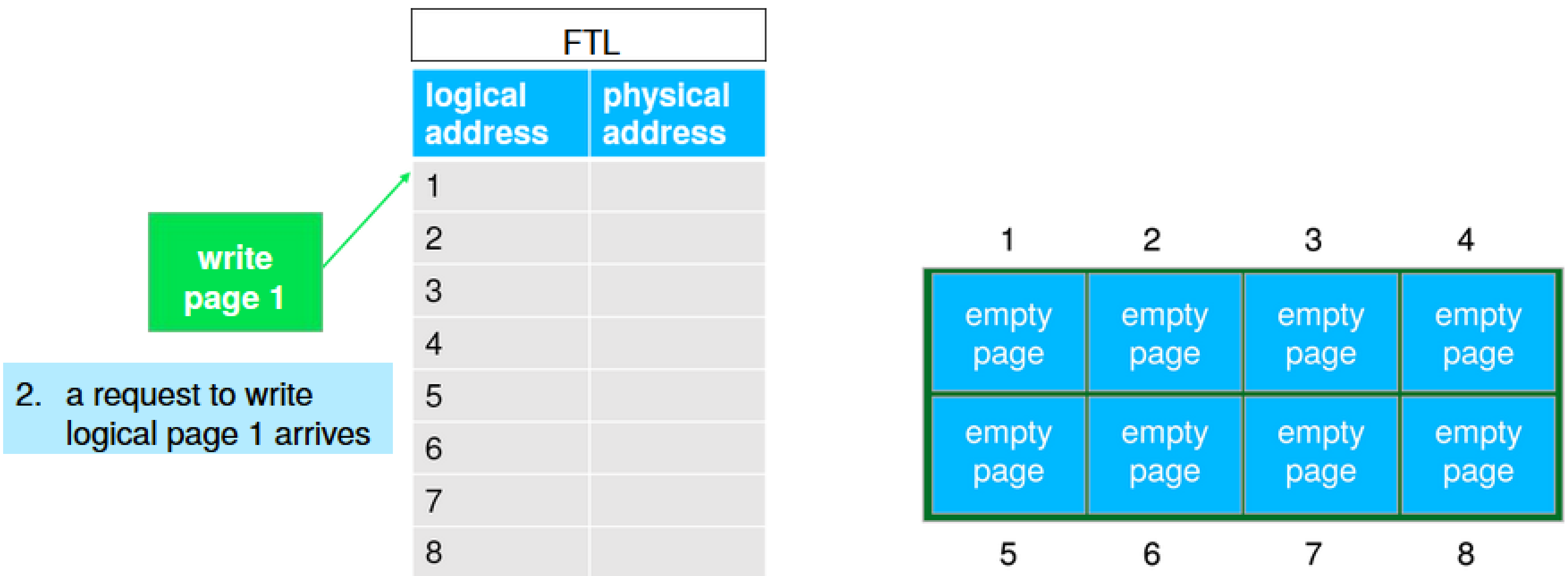
NAND Flash Controller Algorithms

- With no overwrite, pages end up with mix of valid and invalid data
- To track which logical blocks are valid, controller maintains **flash translation layer** (**FTL**) table
- Also implements **garbage collection** to free invalid page space
- Allocates **overprovisioning** to provide working space for GC
- Each cell has lifespan, so **wear leveling** needed to write equally to all cells

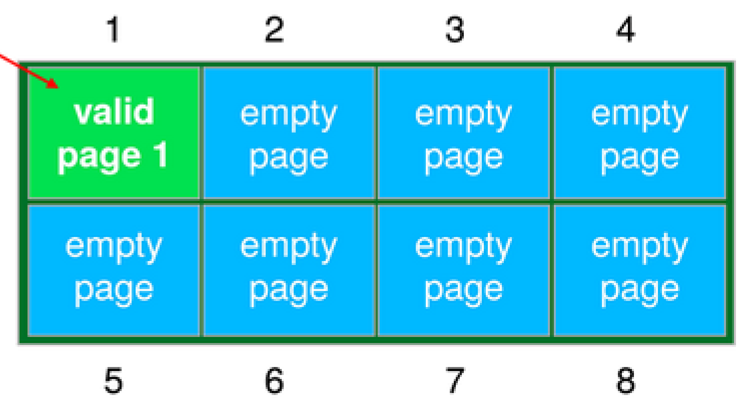
FTL	
logical address	**physical address**
1	
2	
3	
4	
5	
6	
7	
8	



1. assume eight pages in one block of NAND flash storage, not yet written to


write page 1
2. a request to write logical page 1 arrives
FTL
logical address
physical address
1
2
3
4
5
6
7
8
1
2
3
4
empty page
empty page
empty page
empty page
empty page
empty page
empty page
empty page
5
6
7
8

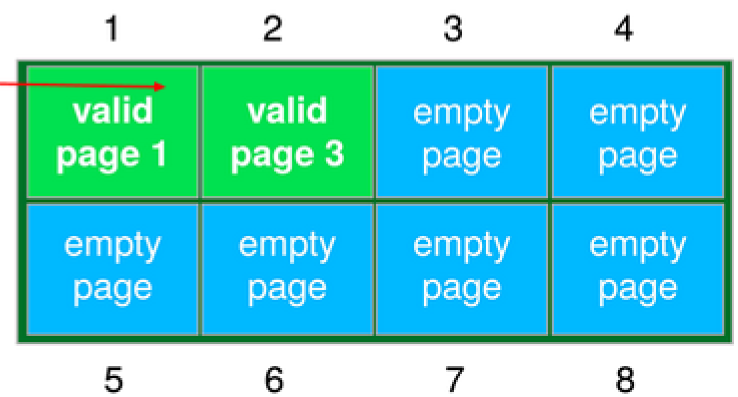
FTL	
logical address	physical address
1	1
2	
3	
4	
5	
6	
7	
8	



3. physical page 1 chosen, logical page 1 written to flash physical page 1

write page 3

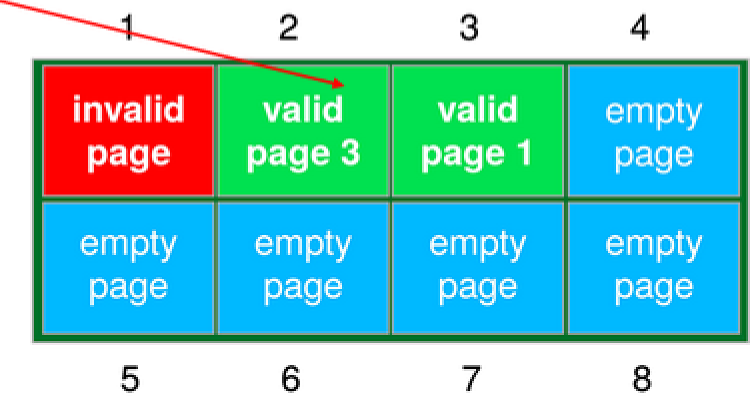
FTL	
logical address	physical address
1	1
2	
3	2
4	
5	
6	
7	
8	



4. logical page 3 write request, physical page 2 chosen and written to

write page 1

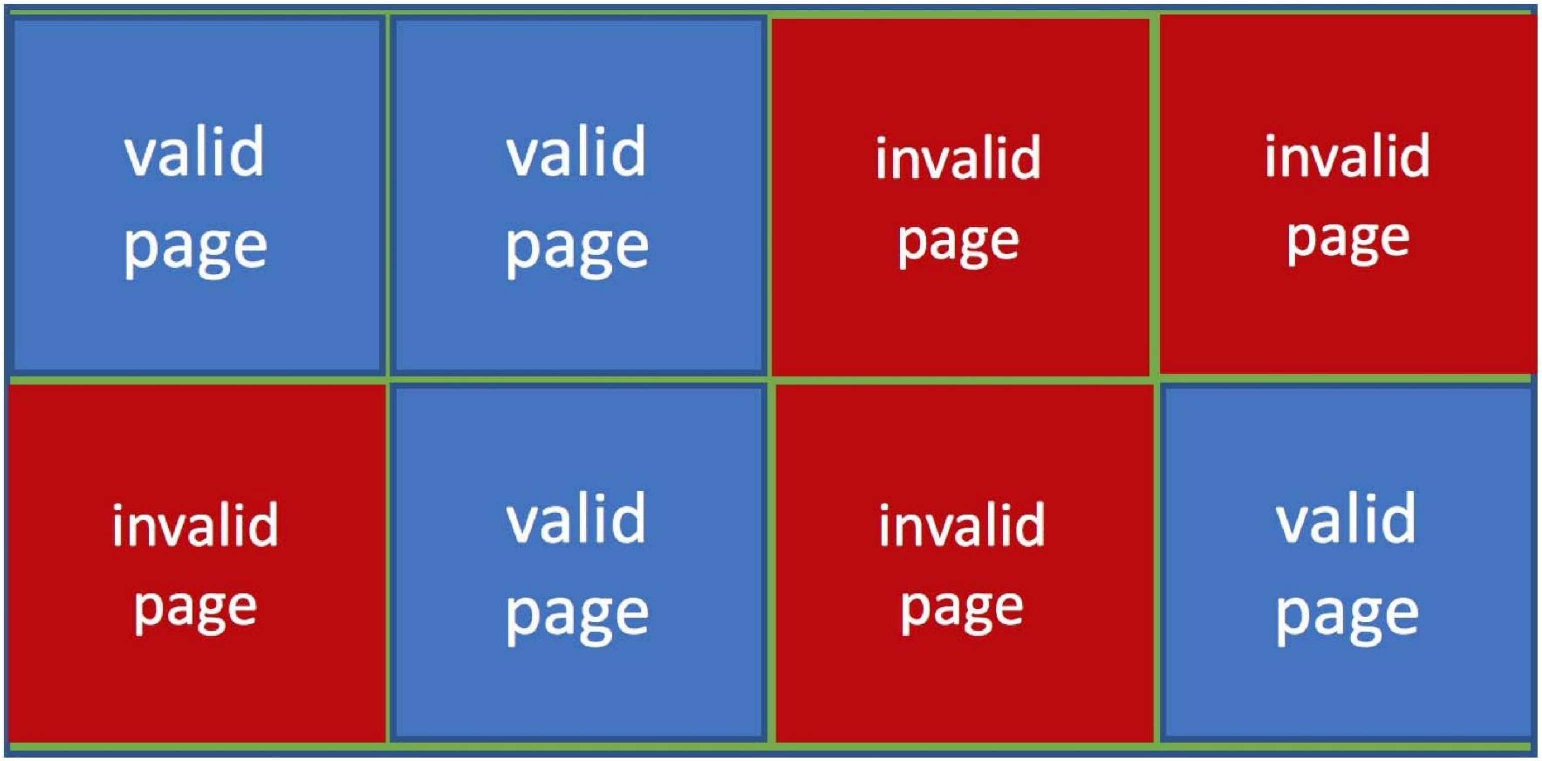
FTL	
logical address	**physical address**
1	3
2	
3	2
4	
5	
6	
7	
8	



5. update to logical page 1, physical page 3 chosen, original page 1 contents marked invalid

   if only part of page 1 had been updated, part of physical page 1 would have been marked invalid and data would have been garbage collected

NAND block with Valid and Invalid pages: Example



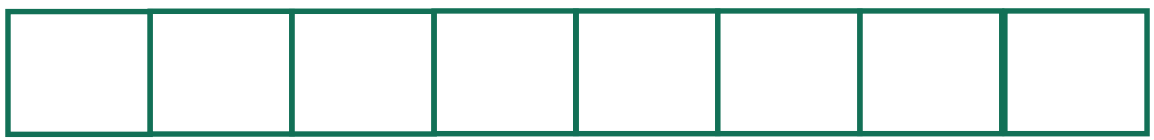
NAND block with valid and invalid pages

Write Amplification

- NVM writes are slow (when compared to reads)
  - As time goes on, a write of n bytes data requires efforts equivalent to a write of m bytes and m >> n
- Because we cannot overwrite pages in NVM, and must erase a whole block before writing to an invalid page

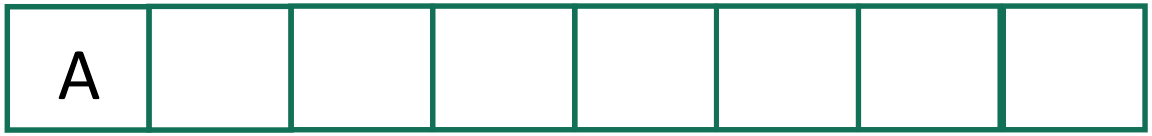
Write Amplification: Example

- Consider a 8 page block, each 4KB

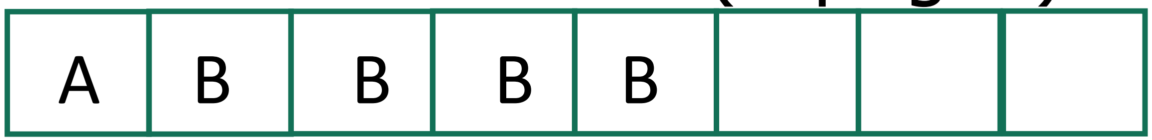


- and the following I/O operation sequence
  - Write file A of 4 KB (1 pages)
  - Write file B of 16 KB (4 pages)
  - Erase file A
  - Write file C of 16 KB (4 pages)

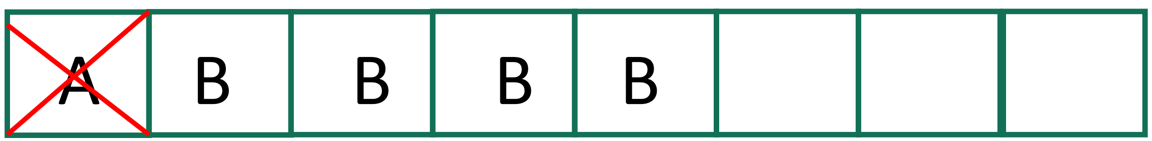
- Write file A of 4 KB (1 pages)



- Write file B of 16 KB (4 pages)

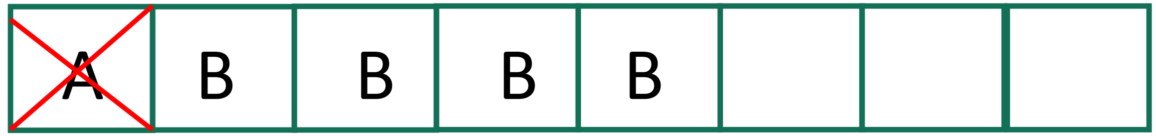


- Erase file A



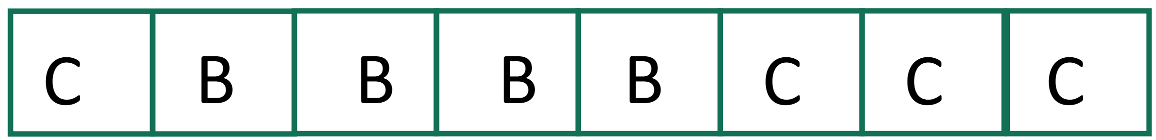
- Write file C of 16 KB (4 pages)
  - ???

On NVM



- Write file C of 16 KB (4 pages)
  - First load whole block to primary memory
    - Read 4+ pages
    - Modify the in-memory block

In memory



  - Erase the whole block on the NVM
    - Erase the whole block (8 pages)
  - Write the block back to the NVM (8 pages)

Write Amplification: Example: Pages Written

- To write
  - A, B, and C: 1 + 4 + 4 = 9 pages
- We must
  - Write 1 + 4 + 8 = 13 pages
  - (Read 4 pages and erase 8 pages)
- As the NVM fills up and files get written / modified / deleted, writes are amplified

NVM and HDD Comparison

- NVM best at random and parallel I/O, HDD at sequential
  - NVM have much higher Input/Output operations per Second (IOPS) (hundreds of thousands vs hundreds)
    - have no moving parts, so no seek time or rotational latency
    - have much smaller random read latency,
    - are great at small writes, and
    - are great at parallel read/write
  - for which HDDs don't do well
- NVMs and HDDs can have similar throughput

Issues with NVMs

- No disk heads or rotational latency but still room for optimization
- The observed behavior of NVM devices
  - the time required to service reads is uniform, but write service time is not uniform.
    - write amplification (one write, causing garbage collection and many read/writes) can decrease the performance advantage of NVMs (when compared to HDDs)

NVM Scheduling

- No disk heads or rotational latency but still room for optimization, e.g.,
  - The controller keeps writing on the NVM until full before it attempts any rewrite/overwrite
  - The OS can, in the background, clean up block with invalid/empty pages so that they’re easily writable when needed
- Considering the observation
  - the time required to service reads is uniform, but write service time is not uniform.
  - Some SSD schedulers have exploited this property and merge only adjacent write requests, servicing all read requests in FCFS order.
- Example
  - In RHEL 7 NOOP (no scheduling) is used for NVMs but adjacent LBA requests are combined

Questions?

- NVMs and characteristics of NVMs
- Comparisons of NVMs and HDDs
- Issues with NVM
  - Can we optimize NVM?
    - NVM Scheduling?

Volatile Memory

- DRAM frequently used as mass-storage device
  - Not technically secondary storage because volatile, but can have file systems, be used like very fast secondary storage
- **RAM drives** (with many names, including RAM disks) present as raw block devices, commonly file system formatted

Why RAM Disks?

- Computers have buffering, caching via RAM, so why RAM drives?
  - Caches / buffers allocated / managed by programmer, operating system, hardware
  - RAM drives under user control
  - Found in all major operating systems
    - Linux `/dev/ram`, macOS `diskutil` to create them, Linux `/tmp` of file system type `tmpfs`
- Used as high speed temporary storage
  - Programs could share bulk date, quickly, by reading/writing to RAM drive

RAM Disk Example

- On Linux

```
mkdir ramdisk
mount -t tmpfs -o size=5m tmpfs ramdisk
```

Magnetic Tape

- Nonvolatile, hold large quantities of data
- Used as an early secondary-storage medium.
- Its access time is slow compared with that of main memory and drives.
- Serv
  now

Use of Magnetic Tape: Example

- https://www.youtube.com/watch?v=avP5d16wEp0#t=01m09s

Questions

- Characteristics, performances, and issues of HDD
- Characteristics, performances, and issues of NVM
- Characteristics, performances, and issues of RAM Disks
- Characteristics, performances, and issues of magnetic tapes