



COS 318: Operating Systems

Storage Devices

Jaswinder Pal Singh
Computer Science Department
Princeton University

<http://www.cs.princeton.edu/courses/archive/fall13/cos318/>



Today's Topics

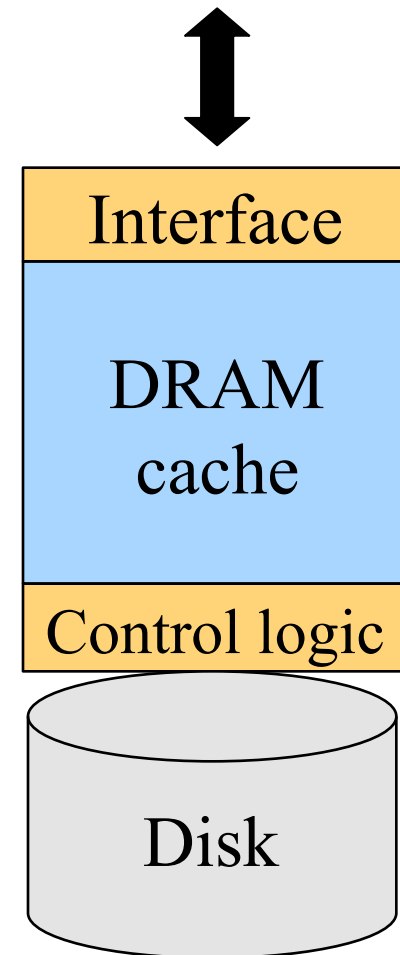
- ◆ Magnetic disks
- ◆ Magnetic disk performance
- ◆ Disk arrays
- ◆ Flash memory



A Typical Magnetic Disk Controller

- ◆ External interfaces (phys + cmds)
 - IDE/ATA, **SATA(1.0, 2.0, 3.0)**
 - SCSI (1, 2, 3),
Ultra-(160, 320, 640) SCSI
 - Fibre channel
- ◆ Cache
 - Cache and buffer data
- ◆ Control logic
 - Read/write operations
 - Disk arm and head positioning
 - Cache replacement
 - Failure detection and recovery

External connection



Disk Caching

◆ Method

- Use DRAM to cache recently accessed blocks
 - Typically a disk has 64-128 MB blocks
 - Some of the RAM space stores “firmware” (an embedded OS)
- Blocks are replaced usually in an LRU order + “tracks”

◆ Pros

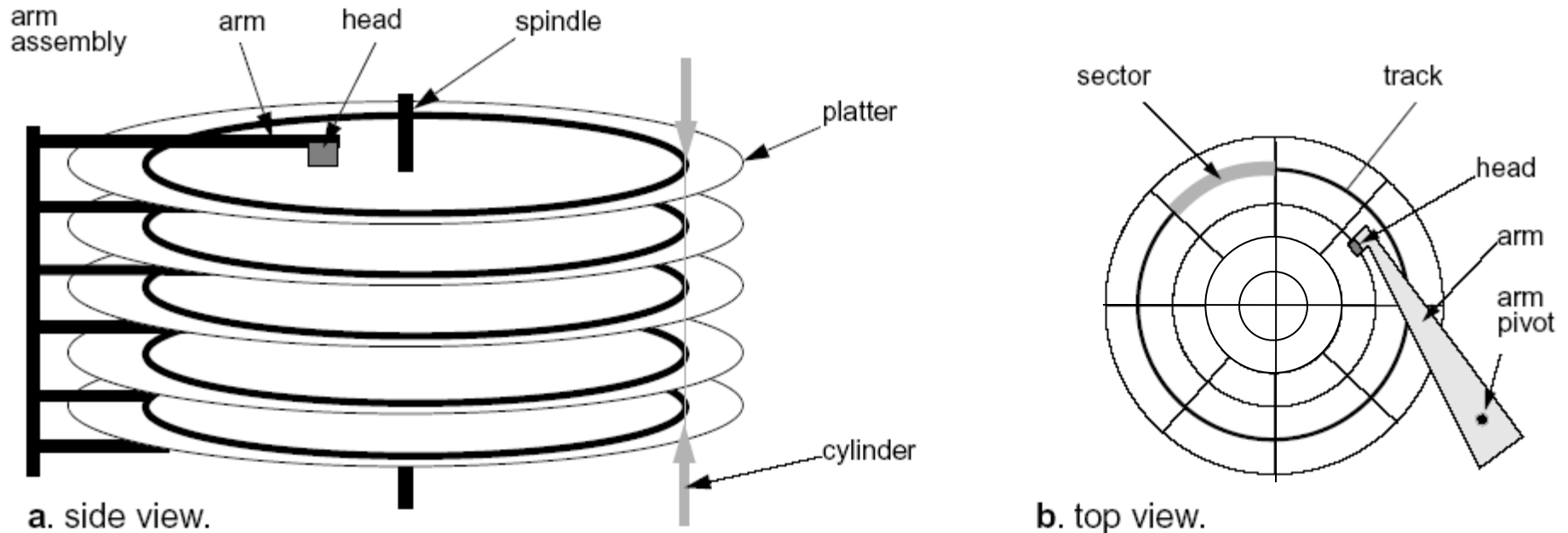
- Good for reads if accesses have locality

◆ Cons

- Doesn't really help with writes since they need to be reliable



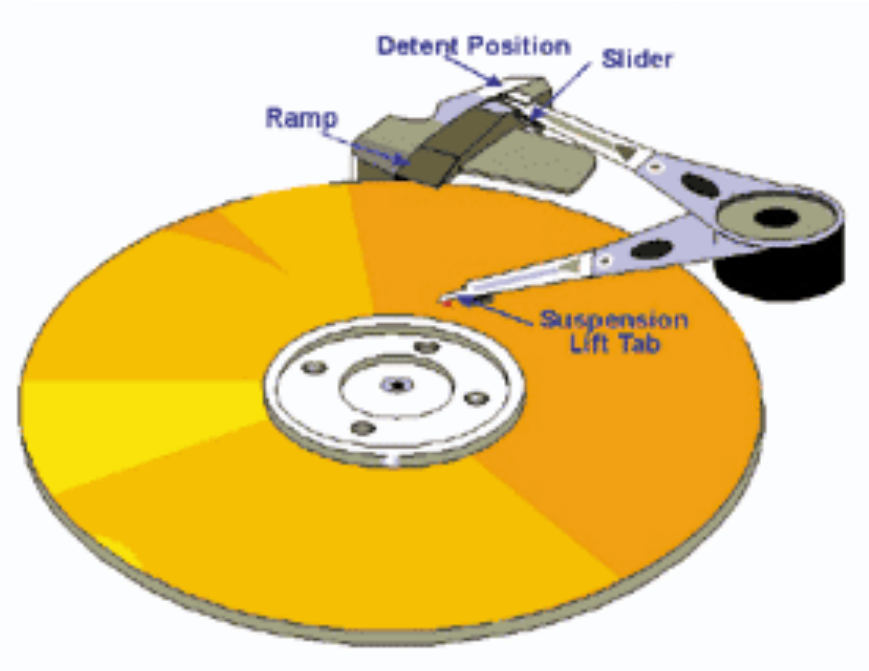
Mechanical Component of A Disk Drive



- ◆ Tracks
 - Concentric rings around platter surface, bits laid out serially along each track
- ◆ Sector
 - Arc of track, holding some min # of bytes, variable # sectors/track
- Cylinder
 - A track viewed across all platters
- ◆ Zone
 - Group of cylinders whose tracks have same number of sectors. 1000-5000 cylinders per zone, 1 spare per zone

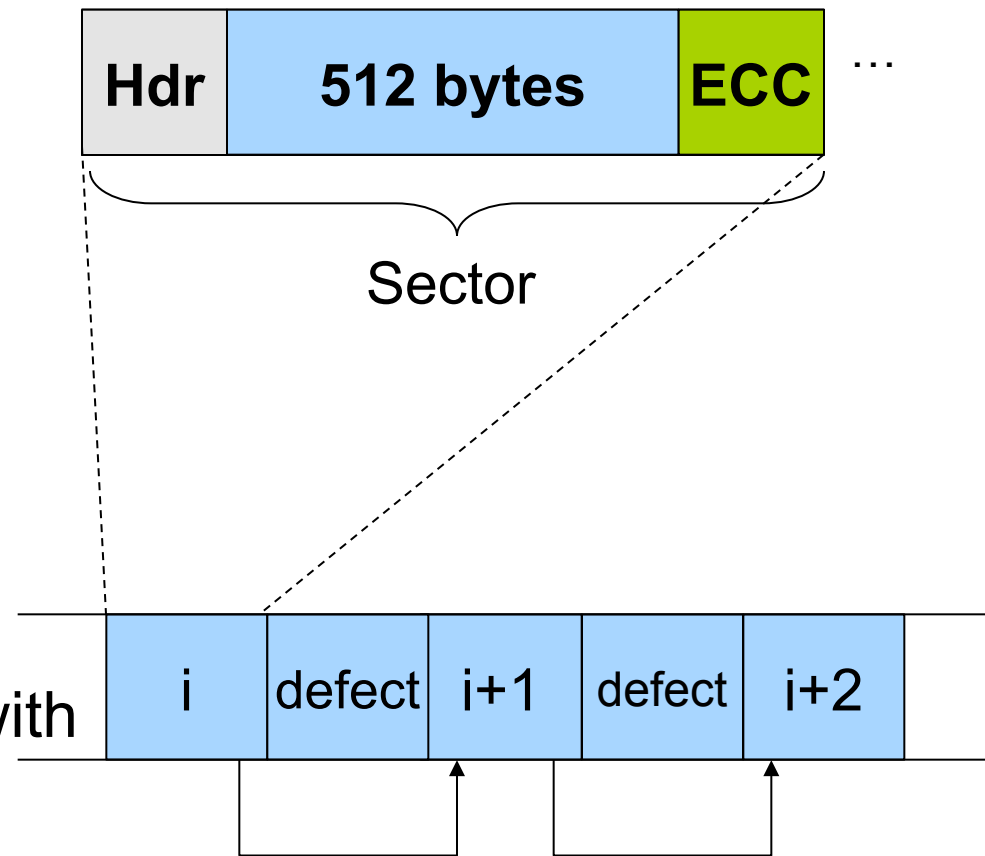
Disk Arm and Head

- ◆ Platter
 - ◆ Spins at about 100 times/s
- ◆ Disk arm
 - Carries disk heads, moves head between edges of platter
- ◆ Disk head
 - Mounted on an actuator
 - Read/write magnetic charges on disk surface
- ◆ Read/write operation
 - Read/write with (track, sector)
 - Seek the right cylinder (tracks)
 - Wait until the sector comes
 - Perform read/write



Disk Sectors

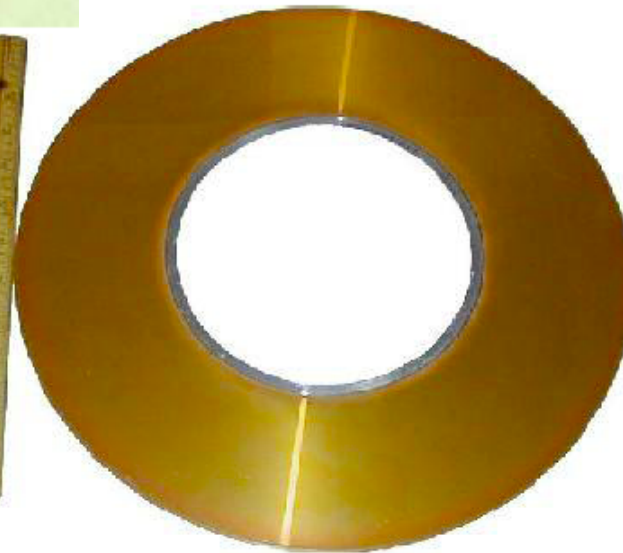
- ◆ Where do they come from?
 - Formatting process
 - Logical # mapped to physical (track, sector)
- ◆ What is a sector?
 - Header (ID, defect flag, ...)
 - Real space (e.g. 512 bytes)
 - Trailer (ECC code)
- ◆ What about errors?
 - Detect errors in a sector
 - Correct them with ECC
 - If not recoverable, replace it with a spare
 - Skip bad sectors in the future



Disks Were Large



First Disk:
IBM 305 RAMAC (1956)
5MB capacity
50 platters, each 24" diam



Storage Form Factors Are Changing



Form factor:
.5-1" · 4" · 5.7"
Storage:
0.5-6TB



Form factor:
.4-.7" · 2.7" · 3.9"
Storage:
0.5-2TB



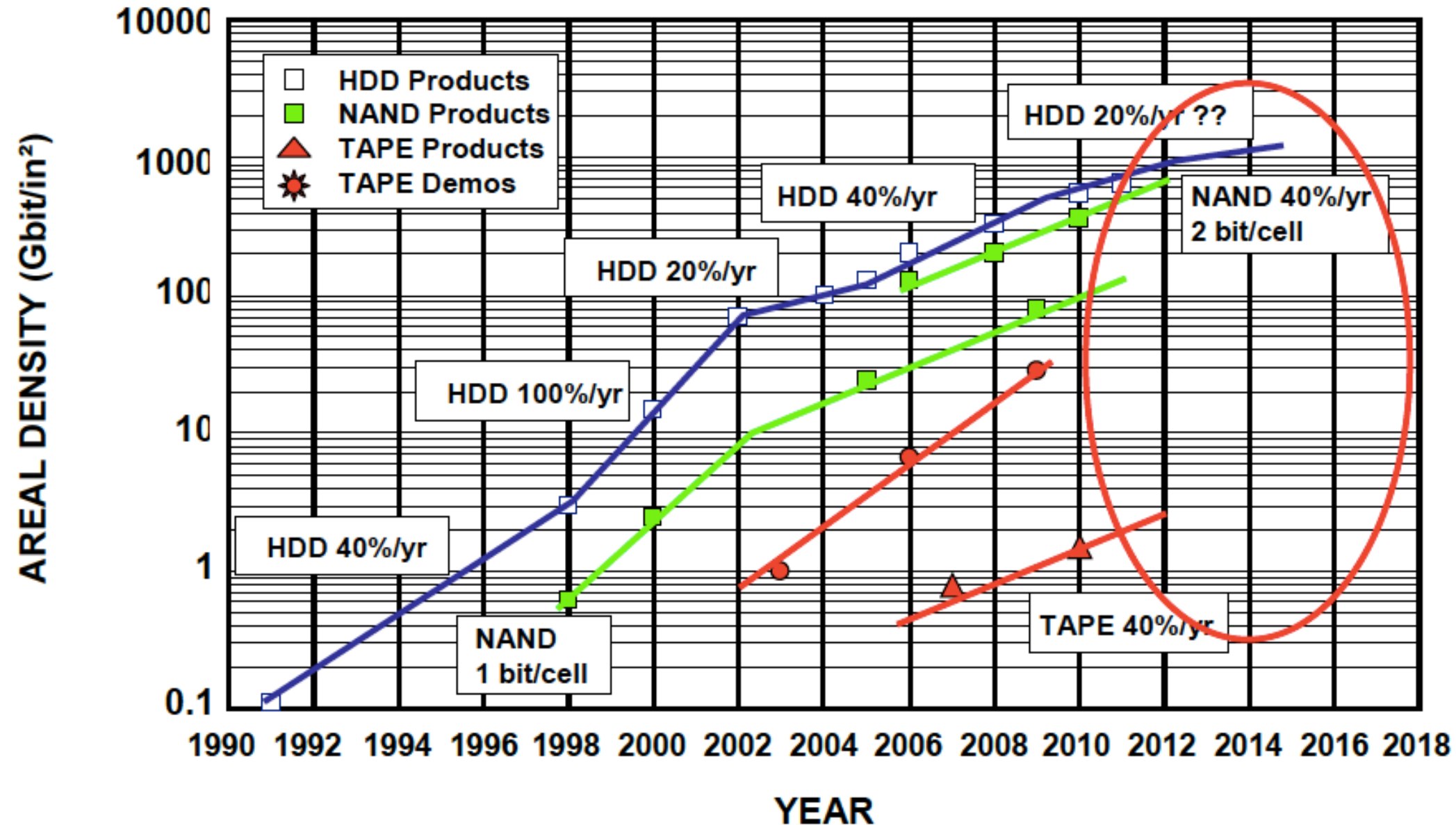
Form factor: 24mm · 32mm · 2.1mm
Storage: 1-1TB



Form factor: PCI card
Storage: 0.5-10TB



Areal Density vs. Moore's Law



50 Years (Mark Kryder at SNW 2006)

	IBM RAMAC (1956)	Seagate Momentus (2006)	Difference
Capacity	5MB	160GB	32,000
Areal Density	2K bits/in ²	130 Gbits/in ²	65,000,000
Disks	50 @ 24" diameter	2 @ 2.5" diameter	1 / 2,300
Price/MB	\$1,000	\$0.01	1 / 100,000
Spindle Speed	1,200 RPM	5,400 RPM	5
Seek Time	600 ms	10 ms	1 / 60
Data Rate	10 KB/s	44 MB/s	4,400
Power	5000 W	2 W	1 / 2,500
Weight	~ 1 ton	4 oz	1 / 9,000



Sample Disk Specs (from Seagate)

	Enterprise Performance	Desktop HDD
Capacity		
Formatted capacity (GB)	600	4096
Discs / heads	3 / 6	4 / 8
Sector size (bytes)	512	512
Performance		
External interface	SATA	SATA
Spindle speed (RPM)	15,000	7,200
Average latency (msec)	2.0	4.16
Seek time, read/write (ms)	3.5/3.9	8.5/9.5
Track-to-track read/write (ms)	0.2-0.4	0.8/1.0
Transfer rate (MB/sec)	138-258	146
Cache size (MB)	128	64
Power		
Average / Idle / Sleep	8.5 / 6 / NA	7.5 / 5 / 0.75
Reliability		
Recoverable read errors	1 per 10 ¹² bits read	1 per 10 ¹⁰ bits read
Non-recoverable read errors	1 per 10¹⁶ bits read	1 per 10 ¹⁴ bits read
MTBF	1.2M hr, at 100% duty cycle	700K hr, at 10-20%



Disk Performance

- ◆ Seek
 - Position heads over cylinder, typically 3.5-9.5 ms
- ◆ Rotational delay
 - Wait for a sector to rotate underneath the heads
 - Typically 2 – 4 ms (7,200 – 15,000RPM, 4 - 8 ms per rotation)
- ◆ Transfer bytes
 - Transfer bandwidth is typically 70 -250 Mbytes/sec
- ◆ Example:
 - Performance of transfer 1 Kbytes of Desktop HDD, assuming BW = 100MB/sec, seek = 5ms, rotation = 4ms
 - Total time = 5ms + 4ms + 0.01ms = 9.01ms
 - What is the effective bandwidth?



More on Performance

- ◆ What transfer size can get 90% of the disk bandwidth?
 - Assume Disk BW = 100MB/sec, avg rotation = 4ms, avg seek = 5ms
 - $\text{size} / (\text{size}/\text{BW} + \text{rotation} + \text{seek}) = \text{BW} * 90\%$
 - $\text{size} = \text{BW} * (\text{rotation} + \text{seek}) * 0.9 / (1 - 0.9)$
 $= 100\text{MB} * 0.009 * 0.9 / 0.1 = 8.1\text{MB}$

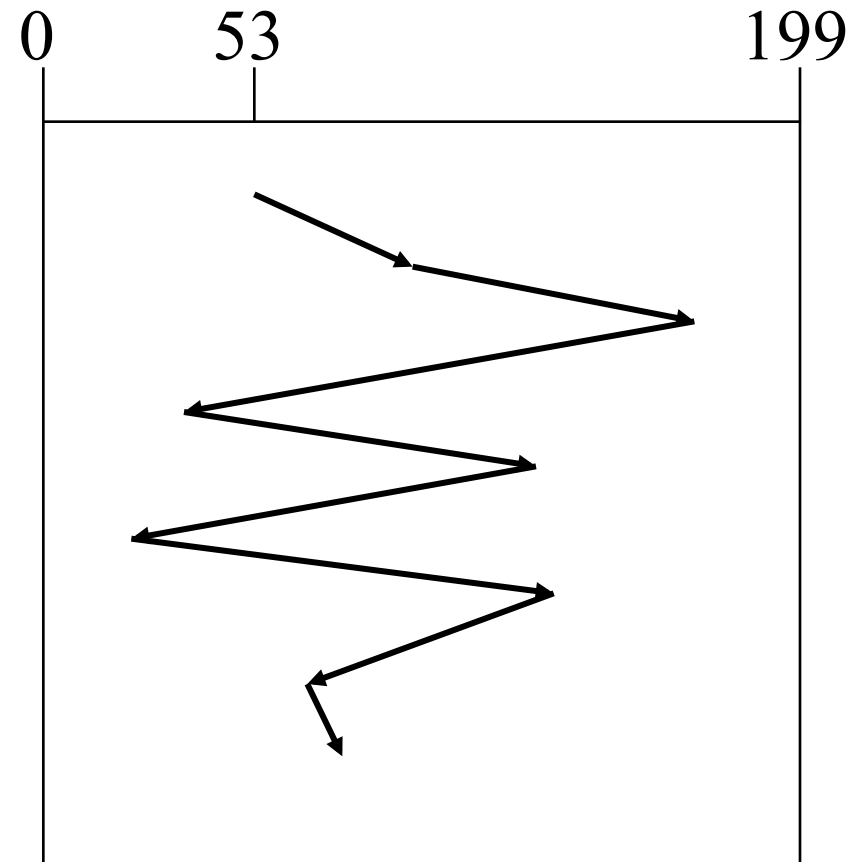
Block Size (Kbytes)	% of Disk Transfer Bandwidth
9Kbytes	1%
100Kbytes	10%
0.9Mbytes	50%
8.1Mbytes	90%

- ◆ Seek and rotational times dominate the cost of small accesses
 - Disk transfer bandwidth are wasted
 - Need algorithms to reduce seek time



FIFO (FCFS) order

- ◆ Method
 - First come first serve
- ◆ Pros
 - Fairness among requests
 - In the order applications expect
- ◆ Cons
 - Arrival may be on random spots on the disk (long seeks)
 - Wild swing can happen

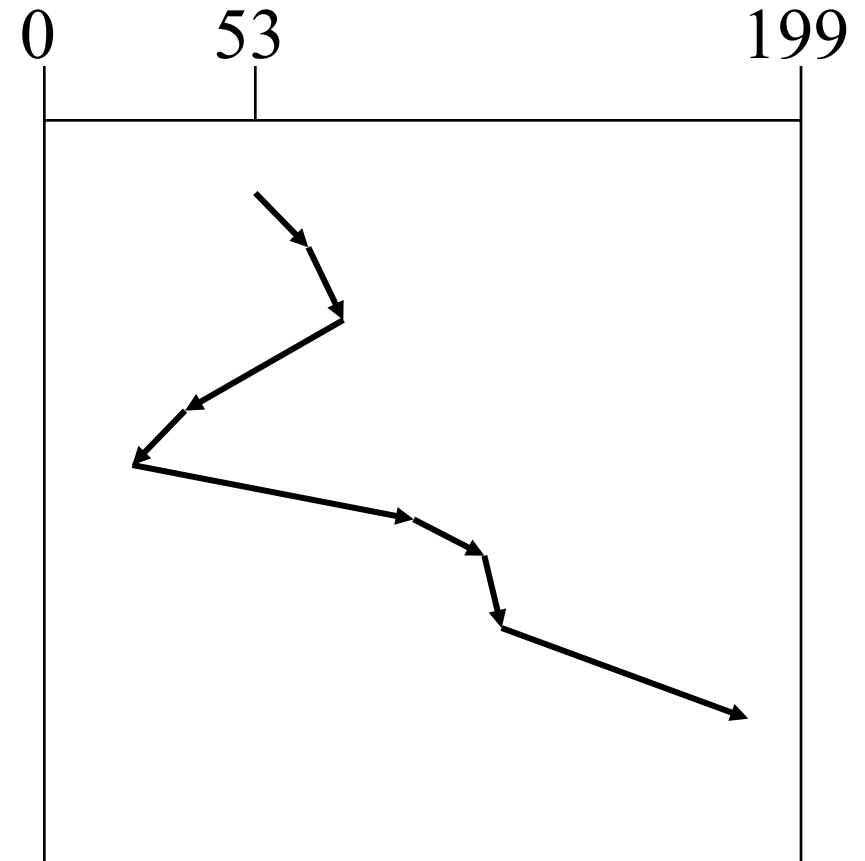


98, 183, 37, 122, 14, 124, 65, 67



SSTF (Shortest Seek Time First)

- ◆ Method
 - Pick the one closest on disk
 - Rotational delay is in calculation
- ◆ Pros
 - Try to minimize seek time
- ◆ Cons
 - Starvation
- ◆ Question
 - Is SSTF optimal?
 - Can we avoid the starvation?

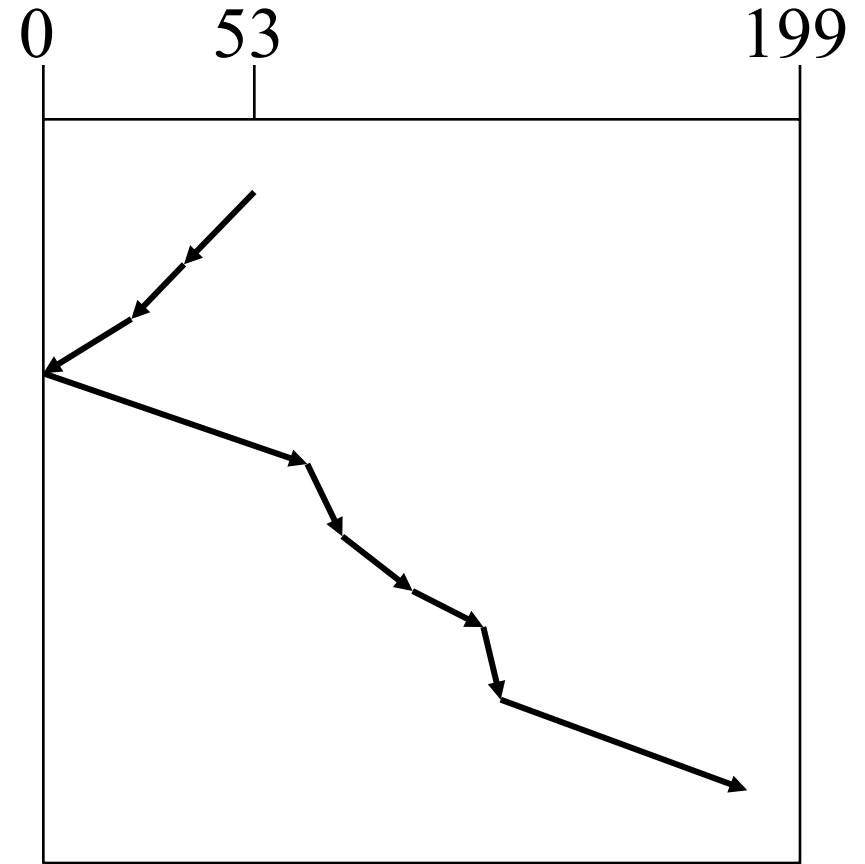


98, 183, 37, 122, 14, 124, 65, 67
(65, 67, 37, 14, 98, 122, 124, 183)



Elevator (SCAN)

- ◆ Method
 - Take the closest request in the direction of travel
 - Real implementations do not go to the end (called LOOK)
- ◆ Pros
 - Bounded time for each request
- ◆ Cons
 - Request at the other end will take a while

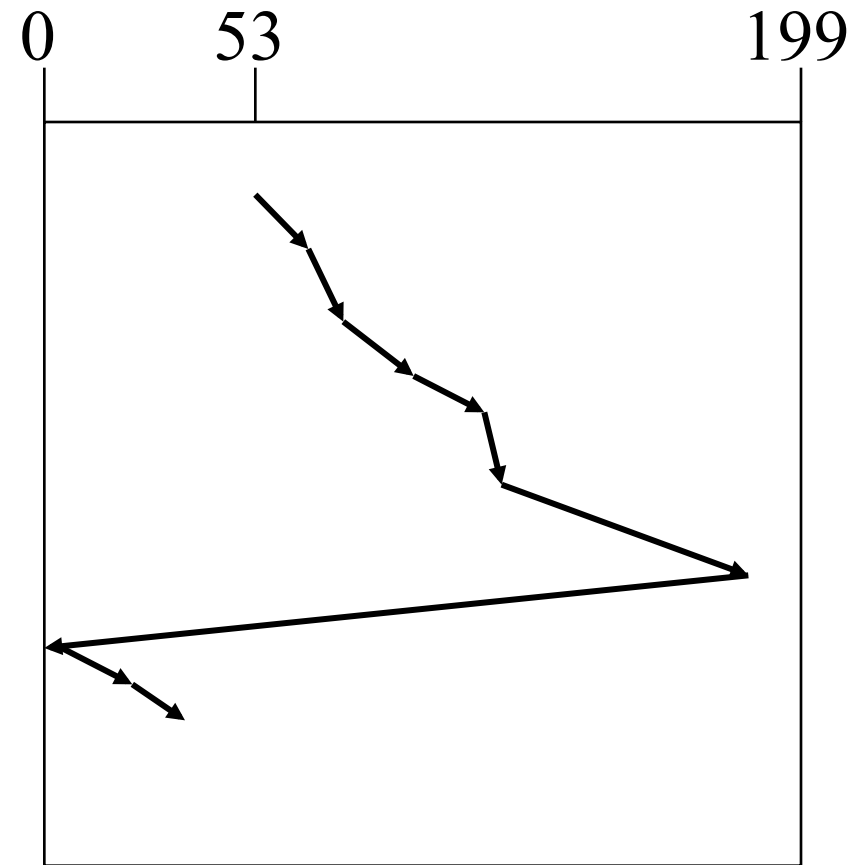


98, 183, 37, 122, 14, 124, 65, 67
(37, 14, 65, 67, 98, 122, 124, 183)



C-SCAN (Circular SCAN)

- ◆ Method
 - Like SCAN
 - But, wrap around
 - Real implementation doesn't go to the end (C-LOOK)
- ◆ Pros
 - Uniform service time
- ◆ Cons
 - Do nothing on the return



98, 183, 37, 122, 14, 124, 65, 67
(65, 67, 98, 122, 124, 183, 14, 37)



Discussions

- ◆ Which is your favorite?
 - FIFO
 - SSTF
 - SCAN
 - C-SCAN
- ◆ Disk I/O request buffering
 - Where would you buffer requests?
 - How long would you buffer requests?
- ◆ More advanced issues
 - Can the scheduling algorithm minimize both seek and rotational delays?



RAID (Redundant Array of Independent Disks)

◆ Main idea

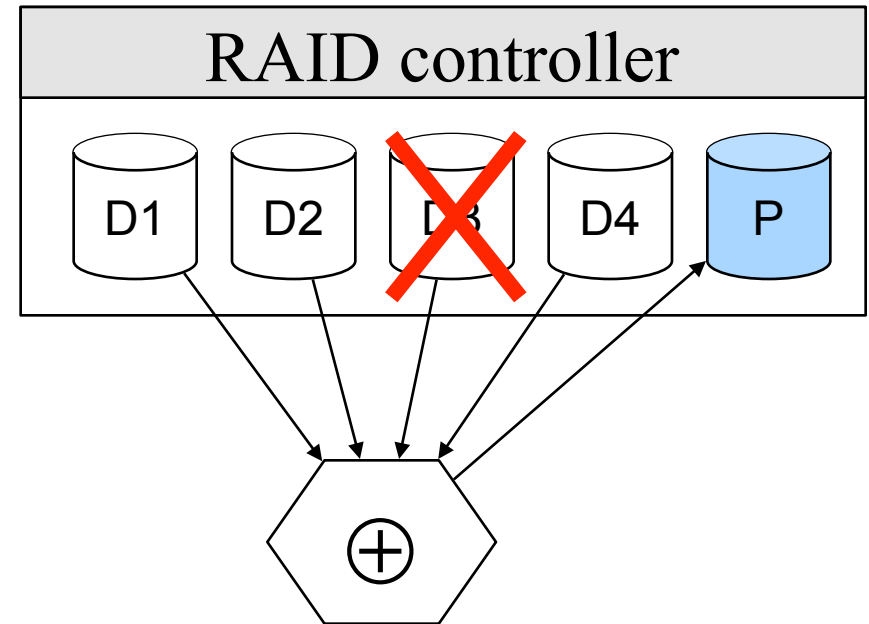
- Compute XORs and store parity on disk P
- Upon any failure, one can recover the block from using P and other disks

◆ Pros

- Reliability
- High bandwidth?

◆ Cons

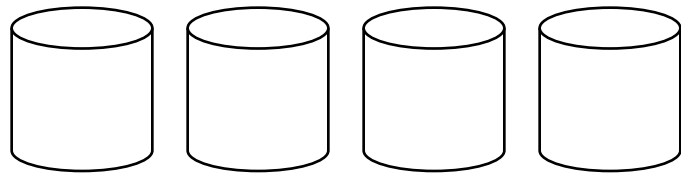
- Cost
- The controller is complex



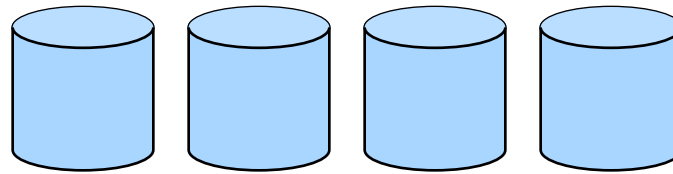
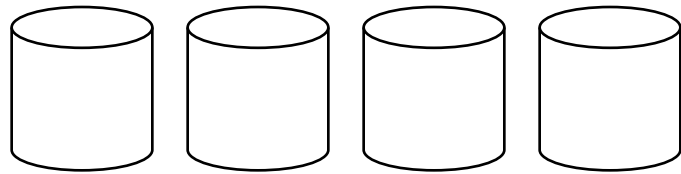
$$P = D1 \oplus D2 \oplus D3 \oplus D4$$

$$D3 = D1 \oplus D2 \oplus P \oplus D4$$

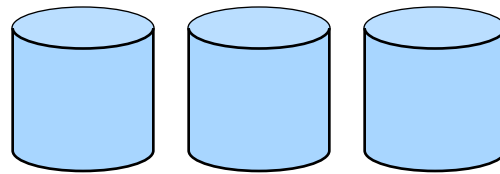
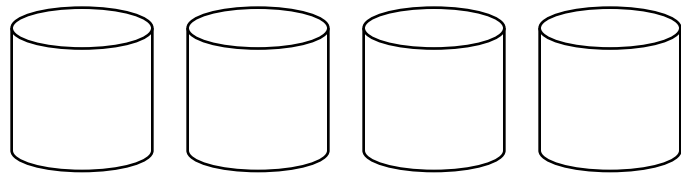
Synopsis of RAID Levels



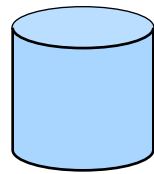
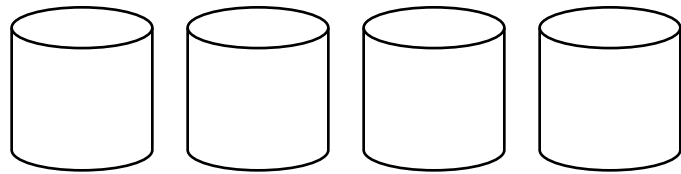
RAID Level 0: Non redundant



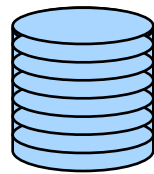
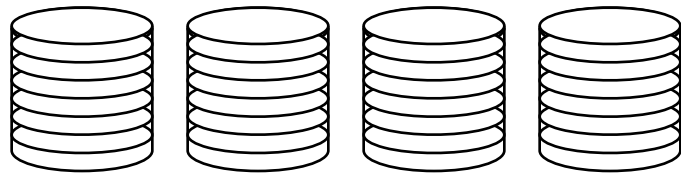
RAID Level 1:
Mirroring



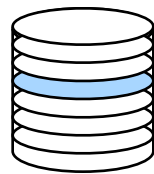
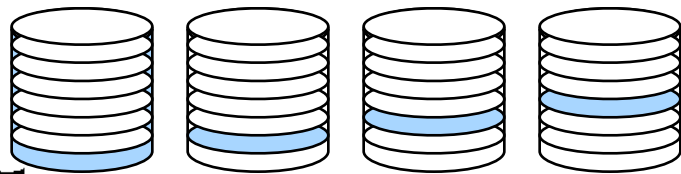
RAID Level 2:
Byte-interleaved, ECC



RAID Level 3:
Byte-interleaved, parity



RAID Level 4:
Block-interleaved, parity

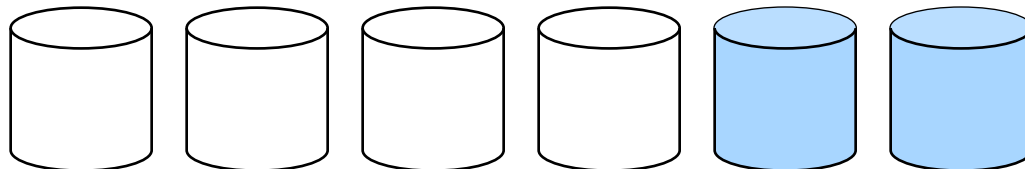
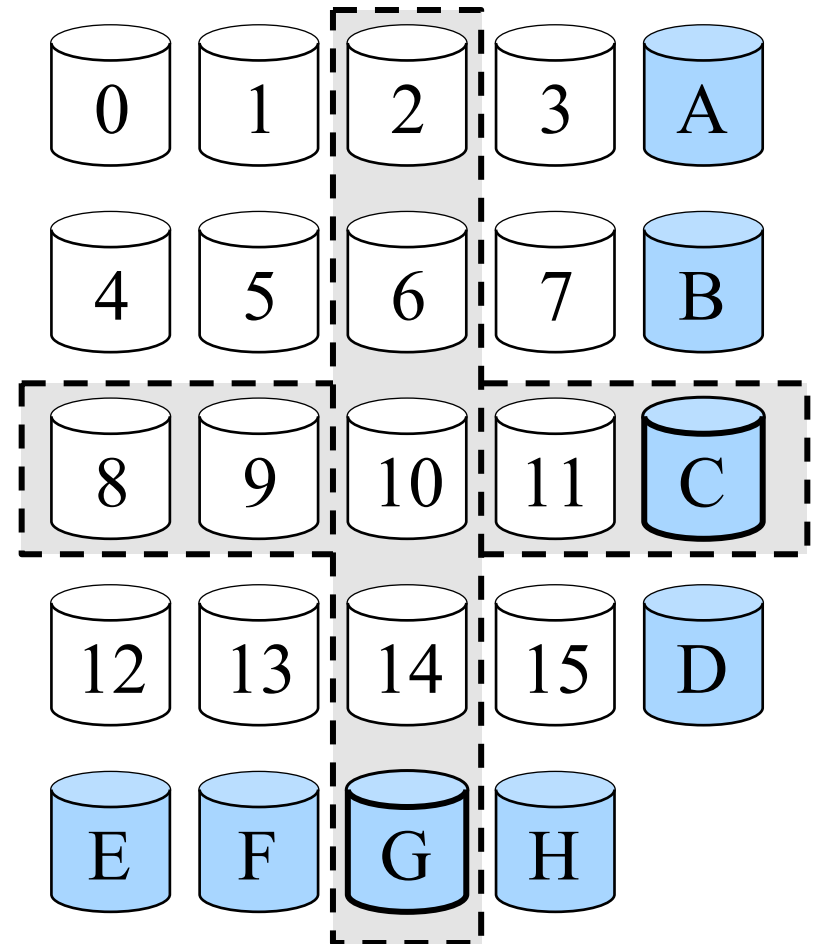


RAID Level 5:
Block-interleaved, distributed parity

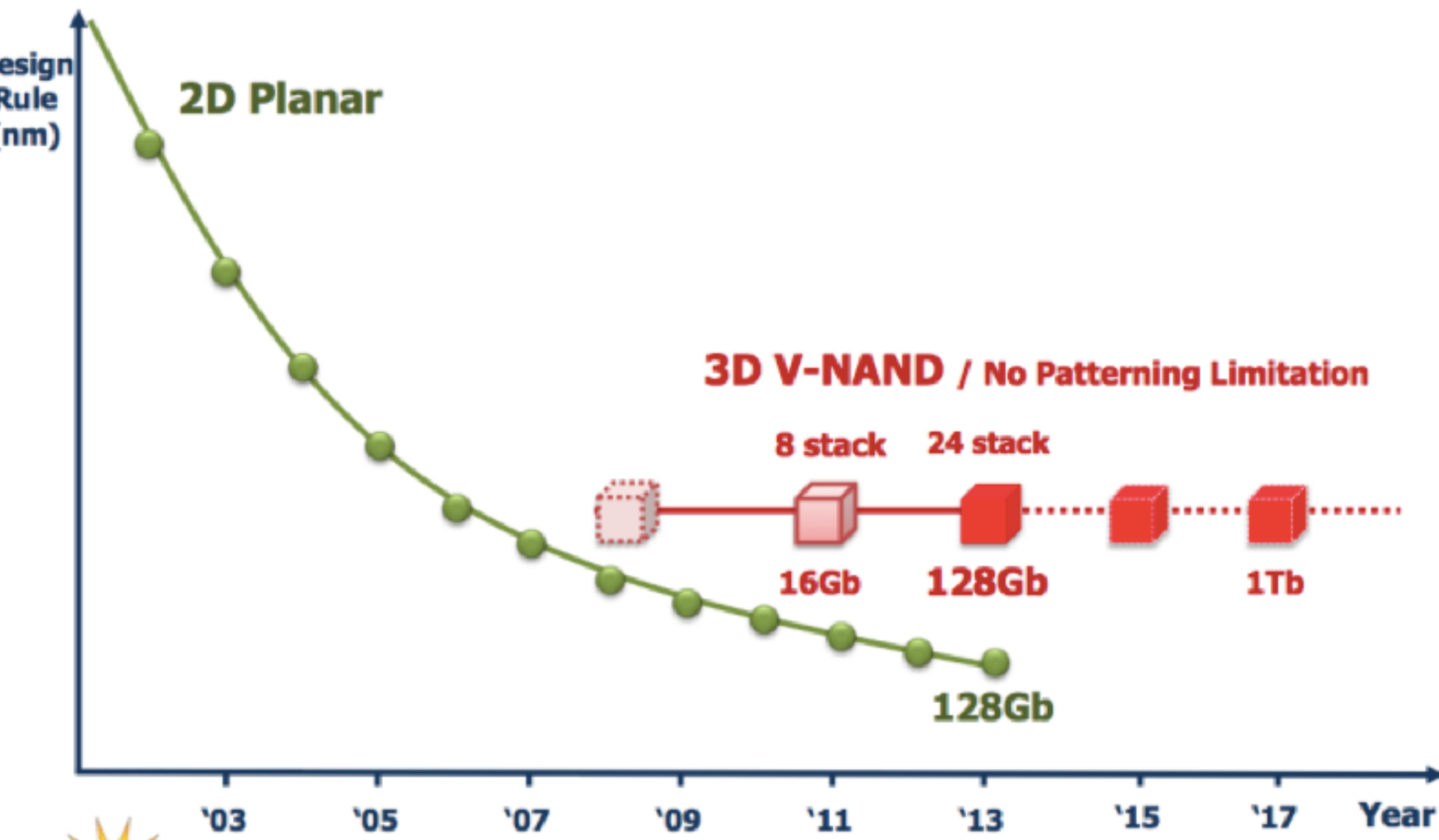


RAID Level 6 and Beyond

- ◆ Goals
 - Less computation and fewer updates per random write
 - Small amount of extra disk space
- ◆ Extended Hamming code
 - Remember Hamming code?
- ◆ Specialized Eraser Codes
 - IBM Even-Odd, NetApp RAID-DP, ...
- ◆ Beyond RAID-6
 - Reed-Solomon codes, using MOD 4 equations
 - Can be generalized to deal with k (>2) disk failures

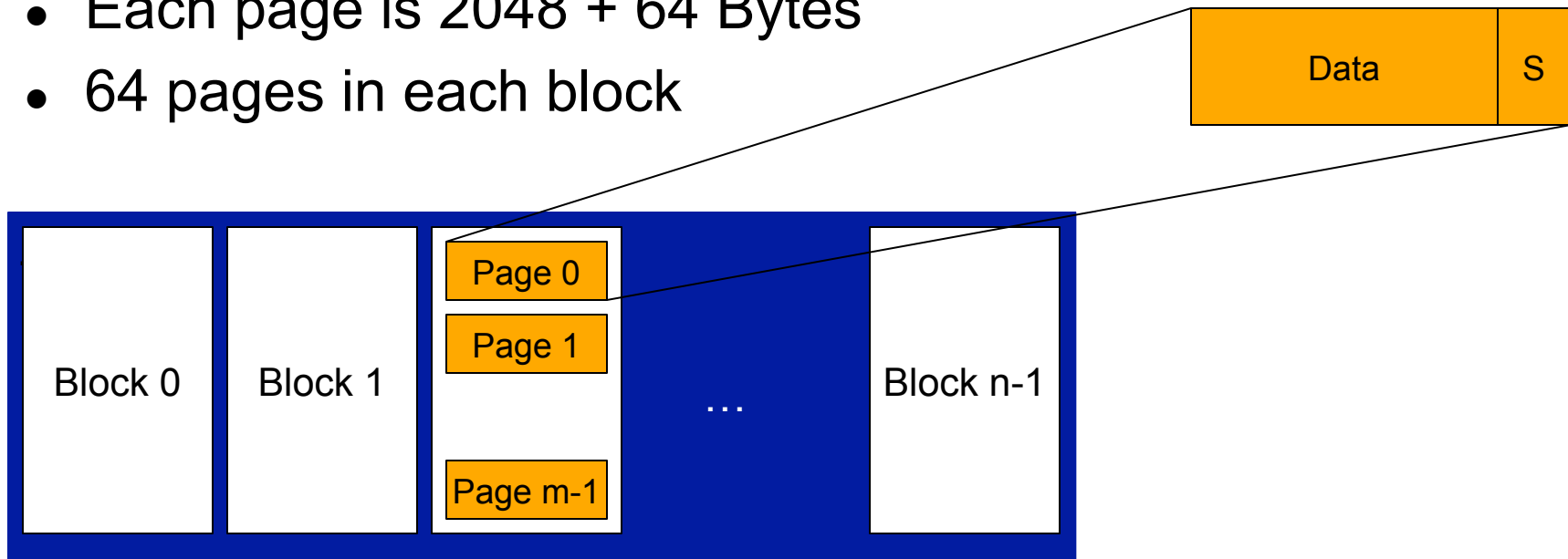


V-NAND Era for the Future



NAND Flash Memory

- ◆ High capacity
 - Single cell (more expensive, durable) vs. multiple cell
- ◆ Small block
 - Each page 512 + 16 Bytes
 - 32 pages in each block
- ◆ Large block
 - Each page is 2048 + 64 Bytes
 - 64 pages in each block



NAND Flash Memory Operations

- ◆ Speed
 - Read page: ~10-20 us
 - Write page: 20-200 us
 - Erase block: ~1-2 ms
- ◆ Limited performance
 - Can only write 0's, so erase (set all 1) then write
- ◆ Solution: Flash Translation Layer (FTL)
 - Map virtual page address to physical page address in flash controller
 - Keep erasing unused blocks
 - Remap to currently erased block to reduce latency

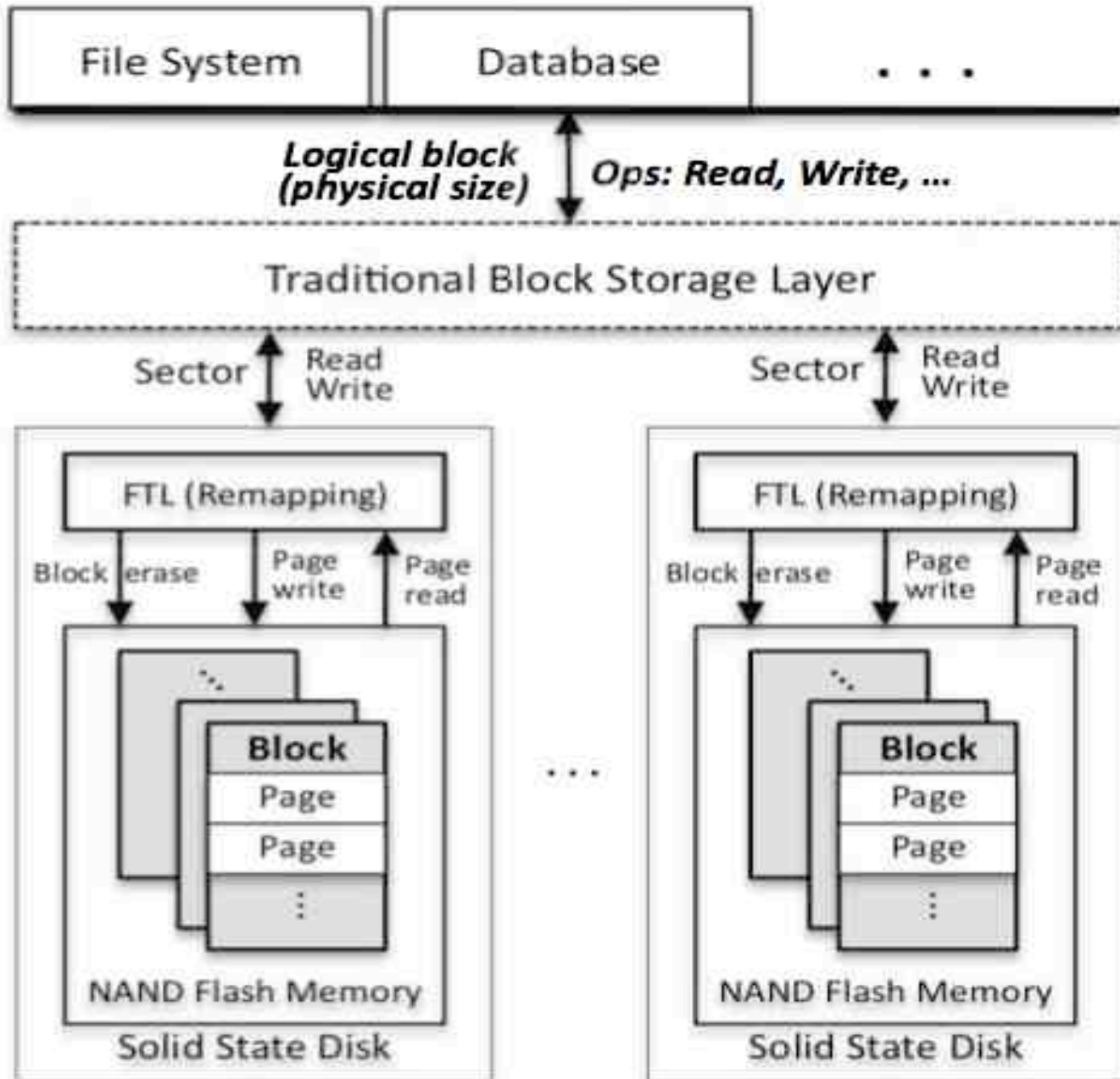


NAND Flash Lifetime

- ◆ Wear out limitations
 - ~50k to 100k writes / page (SLC – single level cell, expensive)
 - ~15k to 60k writes / page (MLC – multi-level; denser, cheaper)
- Wear Leveling:
 - Spread erases evenly across blocks, rather than using same block repeatedly
 - Question: Suppose write to cells evenly and 200,000 writes/sec, how long does it take to wear out 1,000M pages on SLC flash (50k/page)?
- ◆ Who does “wear leveling?”
 - Flash translation layer
 - File system design (later)



Flash Translation Layer



Example: Fusion I/O Flash Memory

- ◆ Flash Translation Layer (FTL) in device controller
 - Remapping
 - Wear-leveling
 - Write buffering
 - Log-structured file system (later)
- ◆ Performance
 - Fusion-IO Octal
 - 10TB
 - 6.7GB/s read
 - 3.9GB/s write
 - 45 μ s latency



Summary

- ◆ Disk is complex
- ◆ Disk real density has been on Moore's law curve
- ◆ Need large disk blocks to achieve good throughput
- ◆ System needs to perform disk scheduling
- ◆ RAID improves reliability and high throughput at a cost
- ◆ Flash memory has emerged at low and high ends

