COS 318: Operating Systems

Storage Devices

Computer Science Department
Princeton University

(http://www.cs.princeton.edu/courses/cos318/)

Where Are We?

- Covered:
  - Management of CPU & concurrency
  - Management of main memory & virtual memory
- Currently --- "Management of I/O devices"
  - Last lecture: Interacting with I/O devices, device drivers
  - This lecture: storage devices
- Then, file systems
  - File system structure
  - Naming and directories
  - Efficiency and performance
  - Reliability and protection

Storage Devices

- Magnetic disks
- Disk arrays
- Flash memory

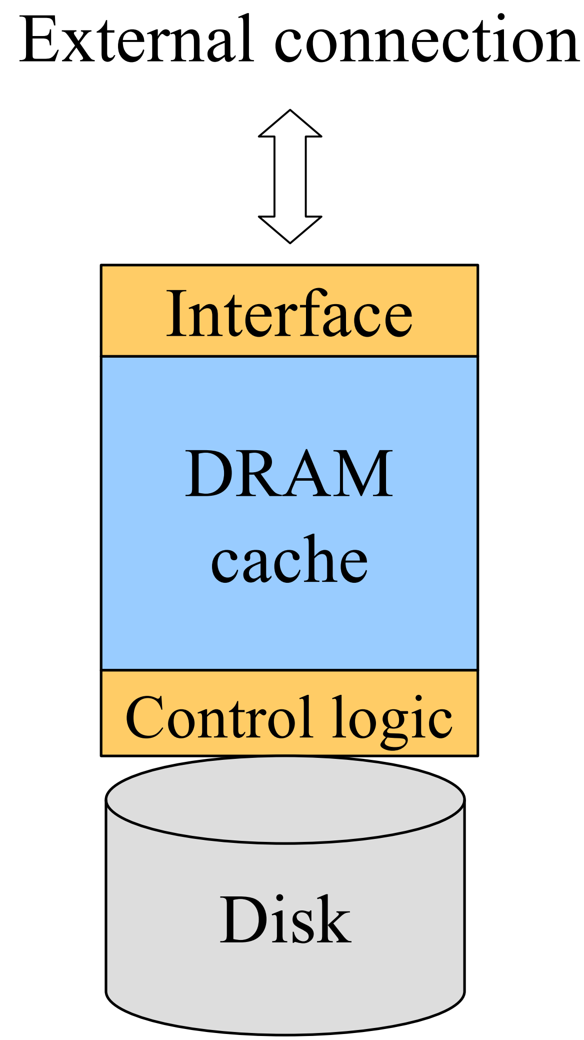
- The devices provide
  - Storage that (usually) survives across machine crashes
  - Block level (random) access
  - Large capacity at low cost
  - Relatively slow performance
    - Magnetic disk read takes 10-20M processor instructions
- Users typically access via file system, which provides a very different interface and translates to blocks

Storage devices

- Magnetic disks
  - Storage that rarely becomes corrupted
  - Large capacity at low cost
  - Block level random access
  - Slow performance for random access
  - Better performance for streaming access
- Flash memory
  - Storage that rarely becomes corrupted
  - Capacity at intermediate cost (50x disk)
  - Block level random access
  - Good performance for reads; worse for random writes

A Typical Magnetic Disk Controller

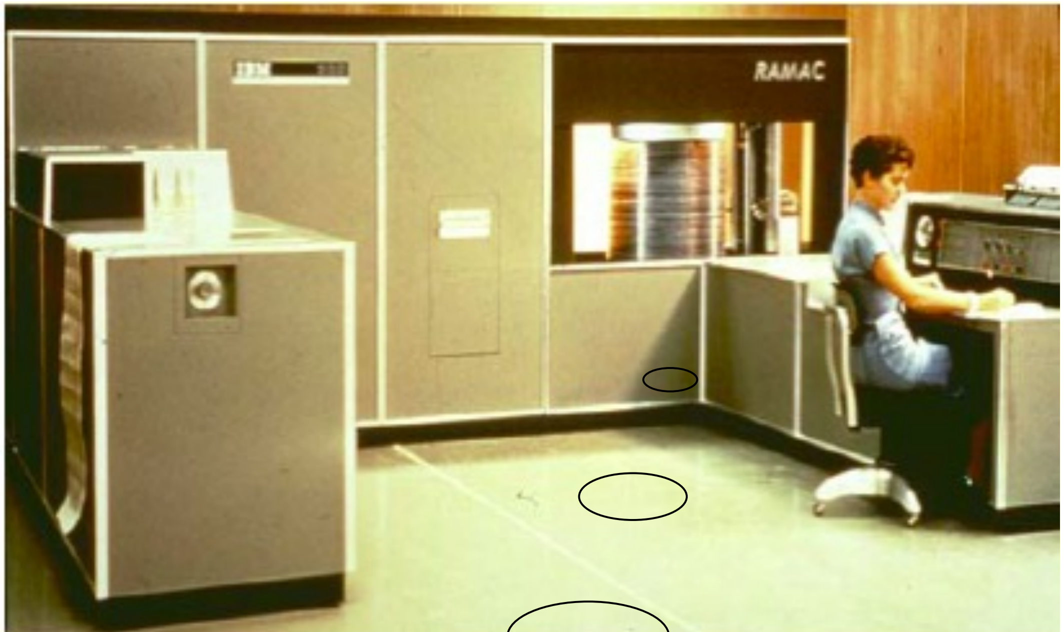
- External interfaces
  - IDE/ATA, **SATA(1.0, 2.0, 3.0)**
  - SCSI, SCSI-2, Ultra-(160, 320, 640) SCSI
  - Fibre channel
- Cache
  - Buffer data between disk and interface
- Control logic
  - Read/write operations (incl. disk head positioning, etc.)
  - Cache replacement
  - Failure detection and recovery

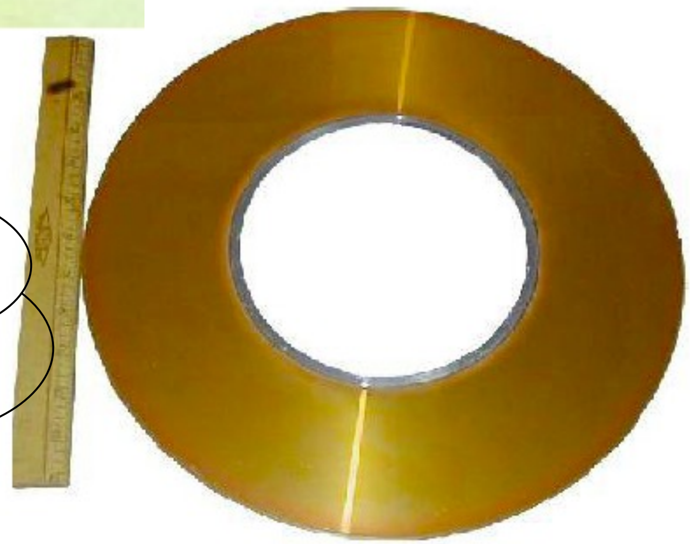


Caching in a Disk Controller

- Method
  - Disk controller has DRAM to cache recently accessed blocks
    - e.g. Hitachi disk has 16MB
    - Some of the RAM space stores "firmware" (an embedded OS)
  - Blocks are replaced usually in an LRU order + "tracks"
  - Disk and Flash devices have CPU in them
- Pros
  - Good for reads if accesses have locality
- Cons
  - Expensive
  - Doesn't really help with writes since they need to be reliable

Disks Were Large





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

Storage Form Factors Are Changing

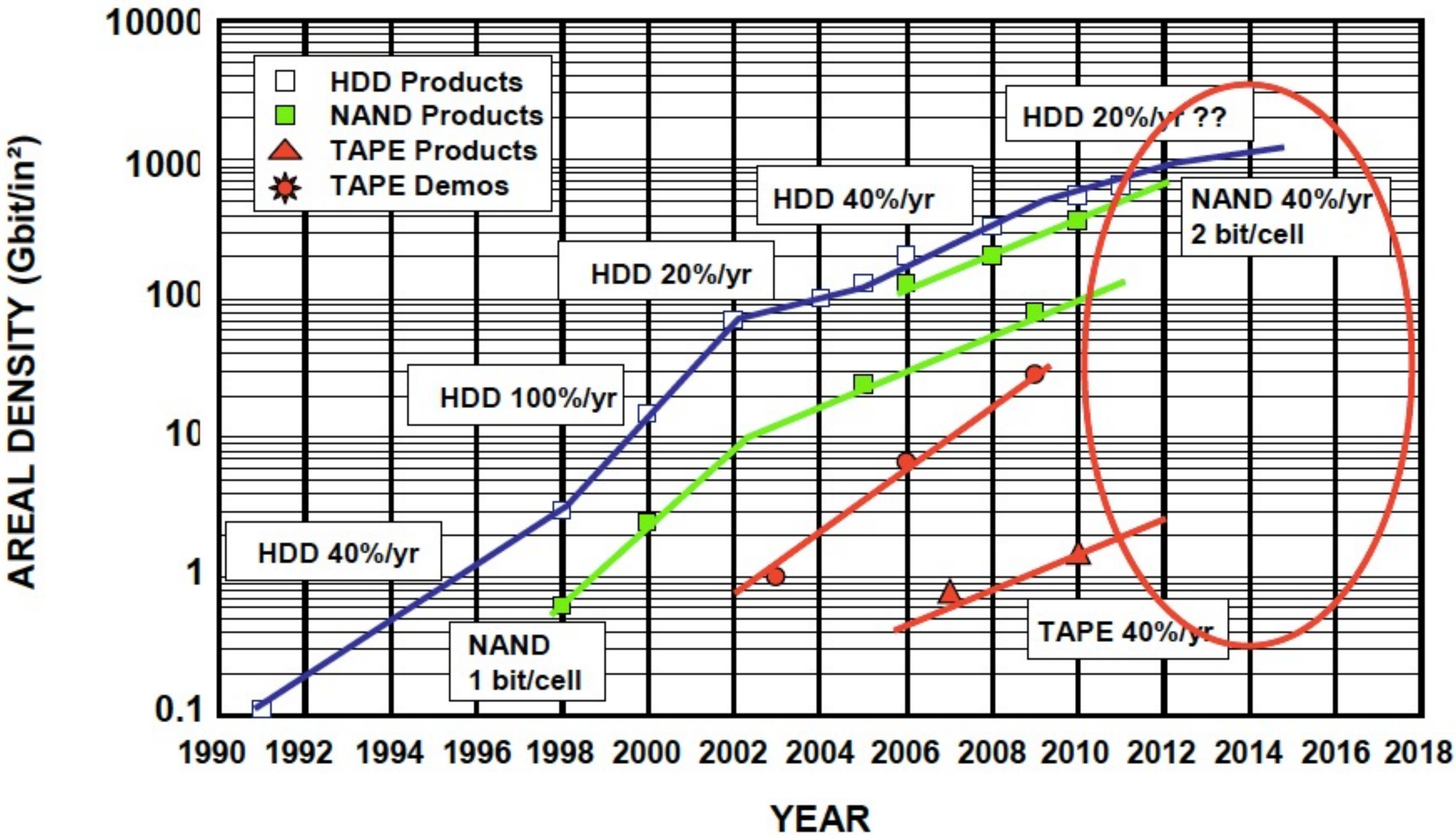
Form factor:
.5-1” × 4” × 5.7”
Storage:
0.5-6TB

Form factor:
.4-.7” × 2.7” × 3.9”
Storage:
0.5-2TB

Form factor: 24mm × 32mm × 2.1mm
Storage: 1-2TB

Form factor: PCI card
Storage: 0.5-10TB

Areal Density vs. Moore's Law



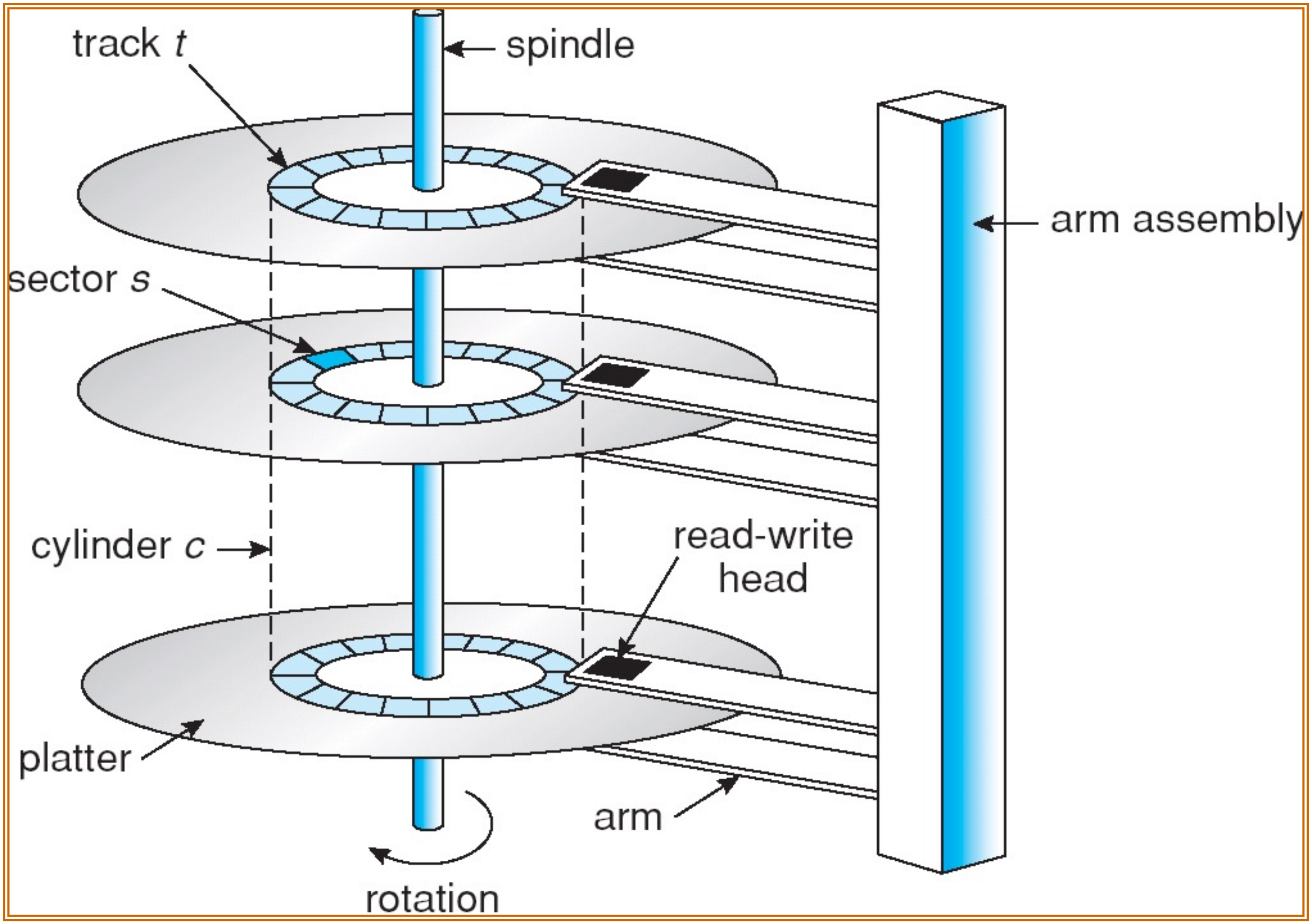
(Fontana, Decad, Hetzler, 2012)

50 Years (Mark Kryder at SNW 2006)

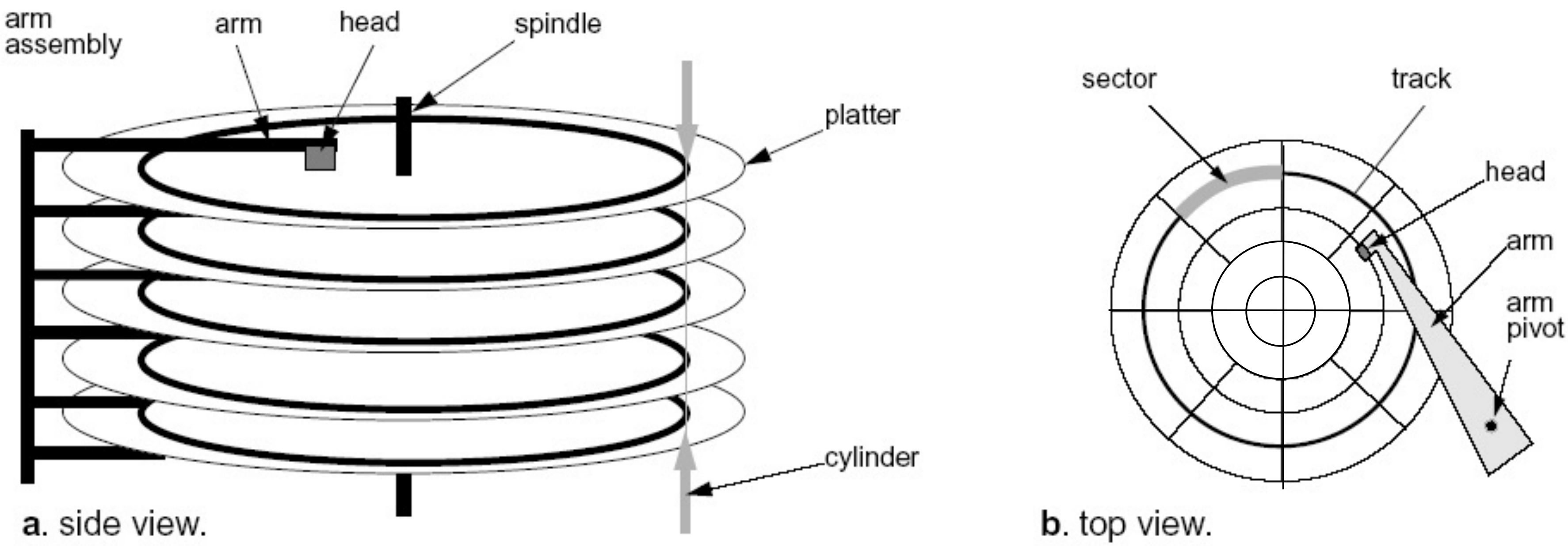
	IBM RAMAC (1956)	Seagate Momentus (2006)	Difference
Capacity	5MB	160GB	32,000
Areal Density	2K bits/in^2	130 Gbits/in^2	65,000,000
Disks	50 @ 24" diameter	2 @ 2.5" diameter	1 / 2,300
Price/MB	$1,000	$0.01	1 / 100,000
Spindle Speed	1,200 RPM	5,400 RPM	5
Seek Time	600 ms	10 ms	1 / 60
Data Rate	10 KB/s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

Magnetic disk

Moving-head Disk Mechanism



Tracks, Cylinders, Sectors



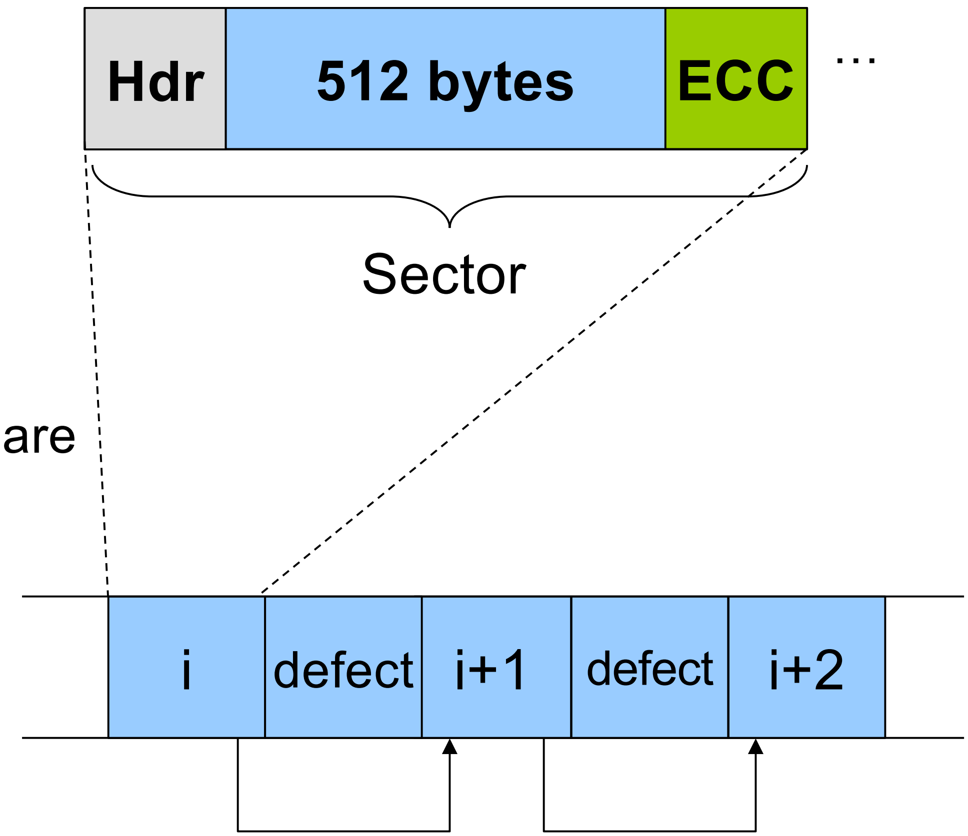
- Tracks
  - Concentric rings around disk surface, bits laid out serially along each track
- Cylinder
  - A track of the platter, 1000-5000 cylinders per zone, 1 spare per zone
- Sector
  - Arc of track holding some min # of bytes, variable # sectors/track

Disk Tracks

- ~1 micron wide
  - Wavelength of light is ~0.5 micron
  - Resolution of human eye is 50 microns
  - 100K tracks on a typical 2.5” disk
- Tracks separated by unused guard regions
  - Reduces likelihood of corrupting nearby tracks during write
- Track length varies across disk
  - Outer tracks have more sectors per track, higher bandwidth
  - Disk organized into “zones” of tracks, each with same no. of sectors per track
  - Only outer half of disk radius is typically used

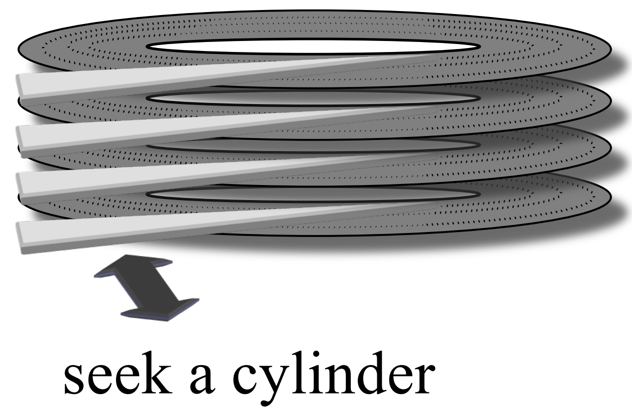
Disk Sectors

- What is a sector?
  - Header (ID, defect flag, …)
  - Real space (e.g. 512 bytes)
  - Trailer (ECC code)
- Skewed from one track to next
  - Accommodate head movement for sequential operations
- Logically addressed (usually)
- Have sophisticated ECC
  - If not recoverable, replace with a spare
- Sector sparing
  - When bad sector, remap it to spare sectors on same surface
  - Skip bad sectors in the future
- Slip sparing
  - When bad sector, remap all sectors to preserve sequential behavior



How Data are Read/Written

- Disk surface
  - Coated with magnetic material
- Disk arm
  - A disk arm carries disk heads
- Disk head
  - Mounted on an actuator
  - Read/write on disk surface
- Read/write operation
  - Disk controller gets read/write with (track, sector)
  - Seek the right cylinder (tracks)
  - Wait until the sector comes under the disk head
  - Perform read/write



Disk Performance

- Disk latency = seek + rotation + transfer (time)
- Seek time
  - Position heads over cylinder, typically 1-20 ms
- Rotation time
  - Wait for a sector to rotate underneath the heads
  - Disk rotation time is yypically 4-15 ms
  - On average, need to wait half a rotation
- Transfer time
  - 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?

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	STA	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 10^{16} bits read**	1 per 10^{14} bits read

Question

- How long to complete 500 random disk reads, in FIFO order?

Question

- How long to complete 500 random disk reads, in FIFO order?
  - Seek: average 10.5 msec
  - Rotation: average 4.15 msec
  - Transfer: 5-10 usec
- 500 * (10.5 + 4.15 + 0.01)/1000 = 7.3 seconds

Question

- How long to complete 500 sequential disk reads?
  - Seek Time: 10.5 ms (to reach first sector)
  - Rotation Time: 4.15 ms (to reach first sector)
  - Transfer Time: (outer track)

    500 sectors * 512 bytes / 128MB/sec = 2ms

Total: 10.5 + 4.15 + 2 = 16.7 ms

Disk Performance

- Seek and rotational times dominate the cost of small accesses
  - Disk transfer bandwidth iswasted
  - Need algorithms to reduce seek time
- Let's look at some disk scheduling algorithms

More on Performance

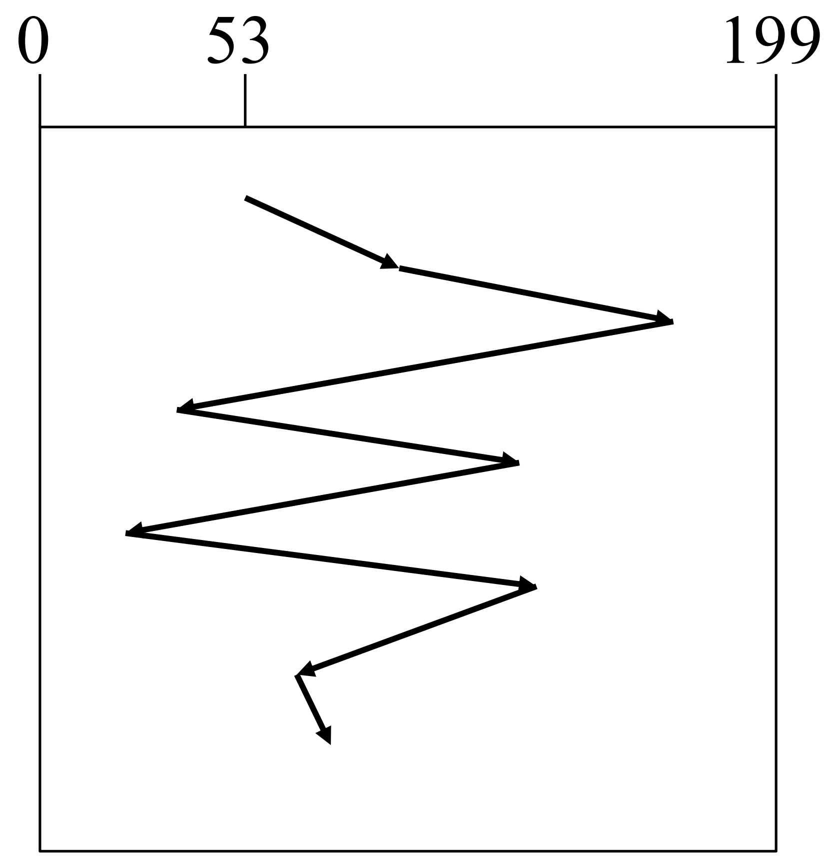
- What transfer size can get 90% of the disk bandwidth?
  - Assume Disk BW = 100MB/sec, avg rotation = 4ms, avg seek = 5ms
  - size / (size/BW + rotation + seek) = BW * 90%
  - size = BW * (rotation + seek) * 0.9 / (1 – 0.9)
    = 100MB * 0.009 * 0.9 / 0.1 = 8.1MB

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

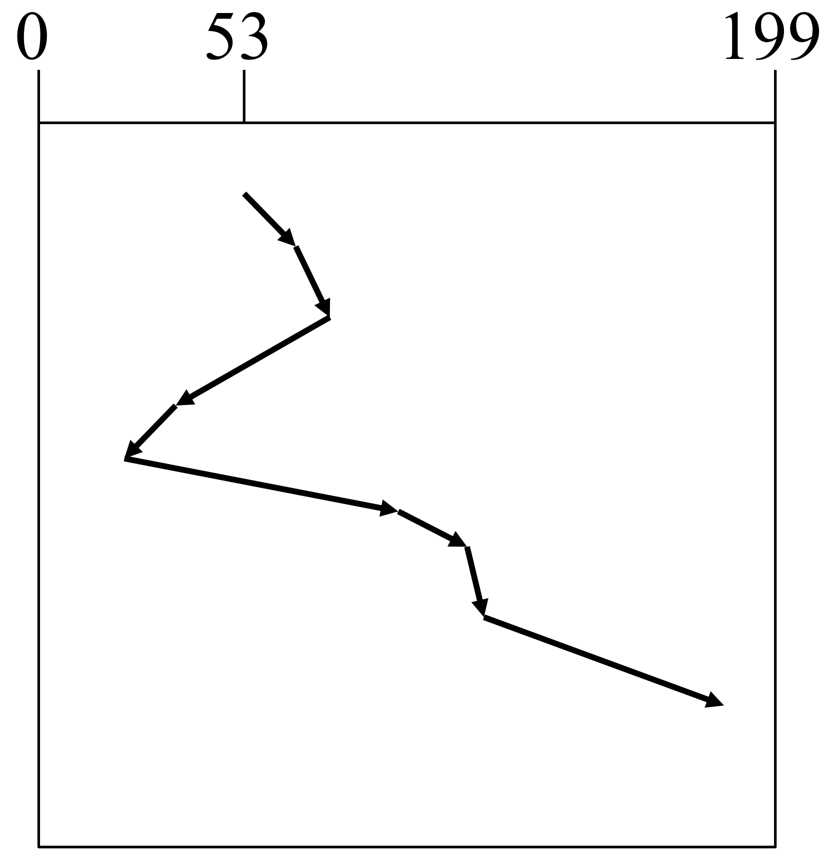
- Method
  - First come first serve
- Pros
  - Fairness among requests
  - In the order applications expect
- Cons
  - Arrival may be on random spots on the disk (long seeks)
  - Wild swings can happen
  - Low throughput, esp with small transfers



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

SSTF (Shortest Seek Time First)

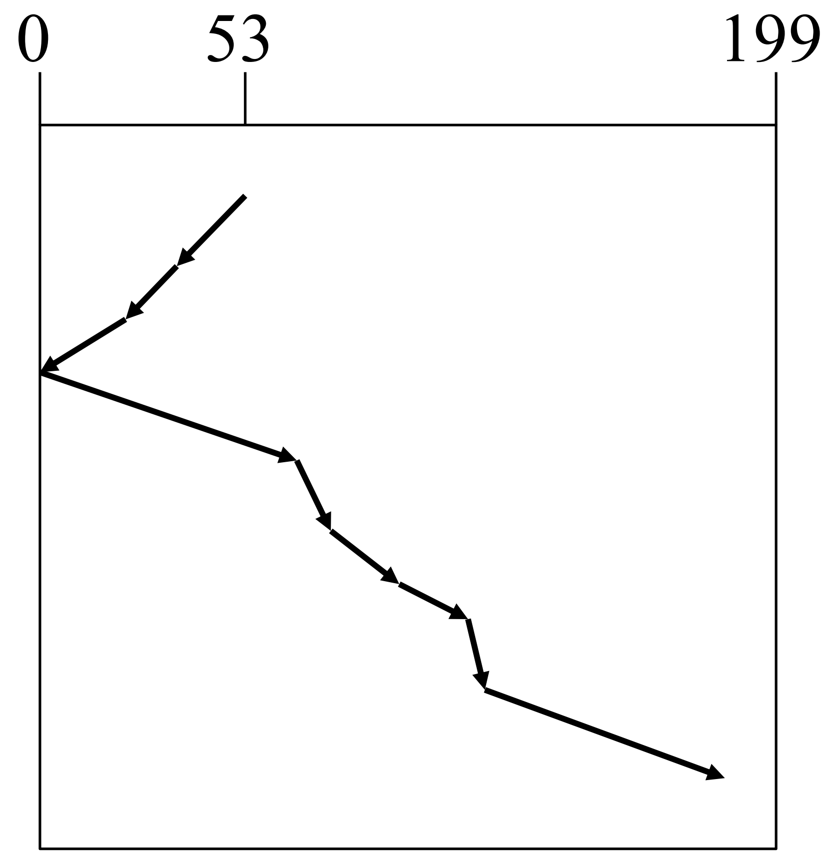
- Method
  - Pick the one closest on disk
  - Can include rotational delay in calculation
- Pros
  - Try to minimize seek (and rotation) time
- Cons
  - Starvation
- Question
  - Is SSTF optimal?
  - Can we avoid the starvation?



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

Elevator (SCAN)

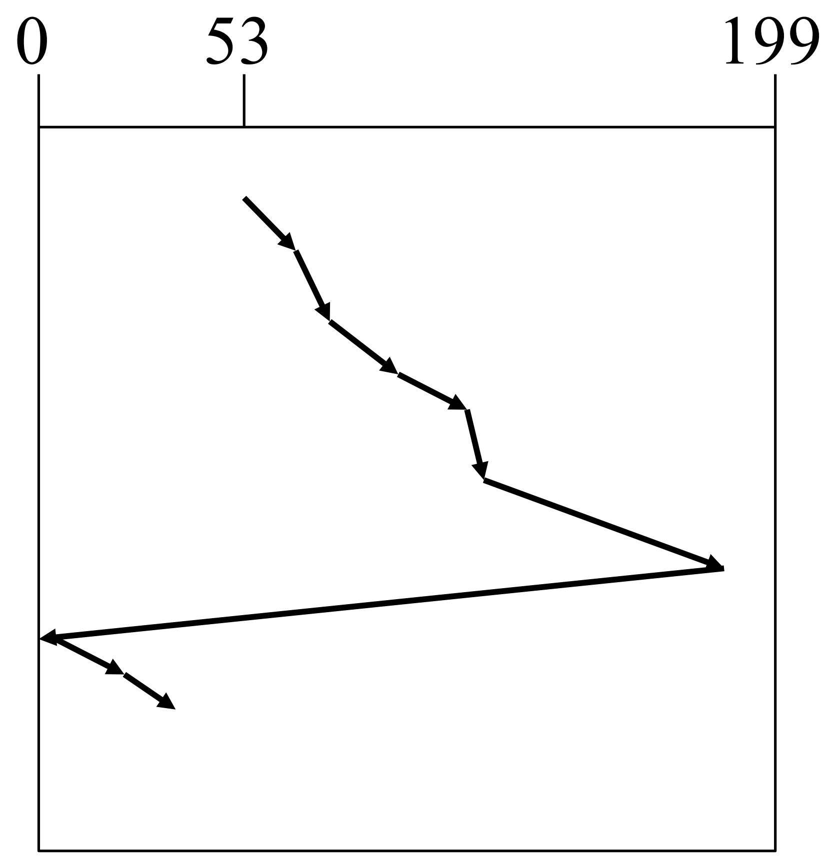
- Method
  - Take the closest request in the direction of travel
  - Real implementations do not go to the end (called LOOK)
- Pros
  - Bounded time for each request
- Cons
  - Request at the other end will take a while



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 bound regardless of where on disk
- Cons
  - Do nothing on the return, so the bound can be larger than in Elevator



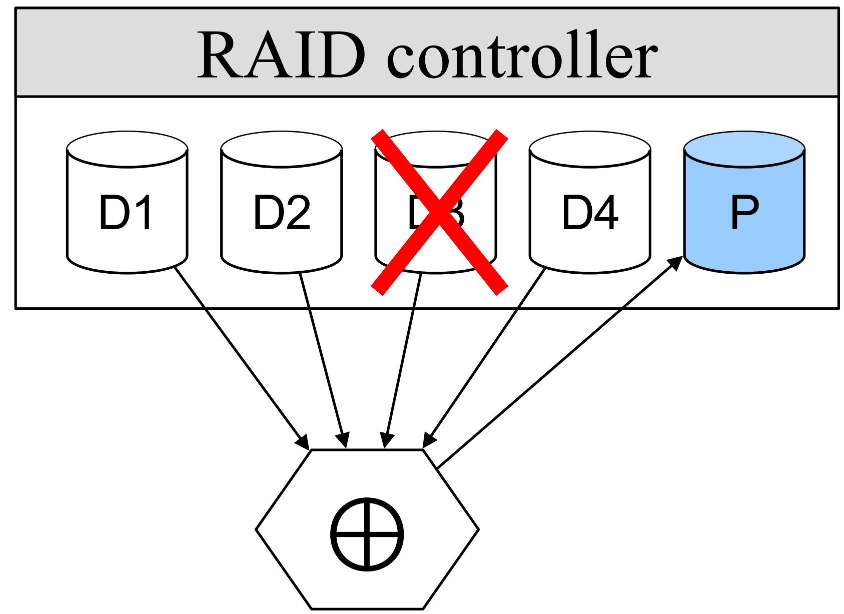
98, 183, 37, 122, 14, 124, 65, 67
(65, 67, 98, 122, 124, 183, 14, 37)

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)

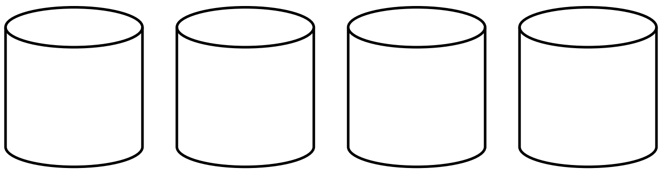
- Main ideas
  - Parallel access
  - Redundancy of data
    - E.g. Compute XORs and store parity on disk P
    - Upon any failure, one can recover the block from using P and other disks
- Pros
  - Reliability
  - High bandwidth?
- Cons
  - Cost
  - The controller is complex



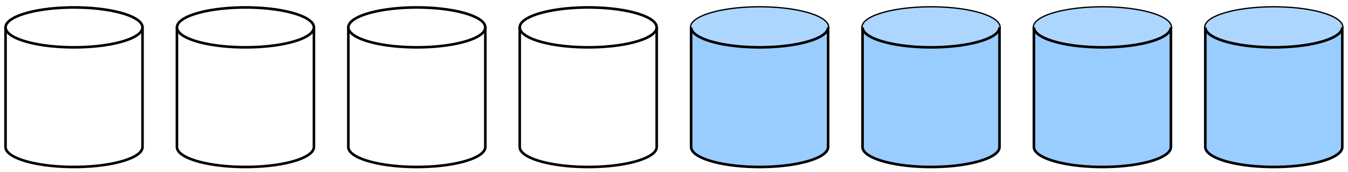
$P = D1 \oplus D2 \oplus D3 \oplus D4$

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

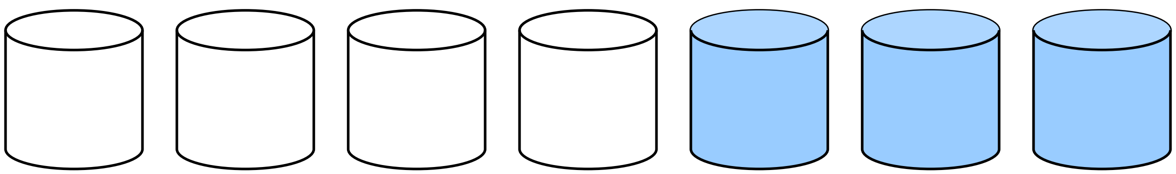
Synopsis of RAID Levels



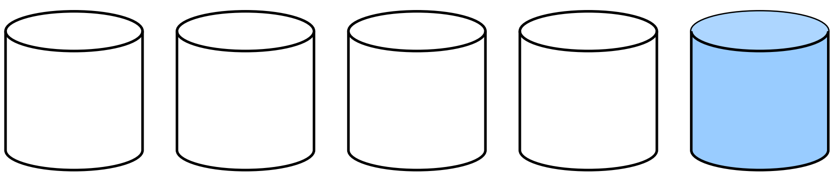
RAID Level 0: Non redundant



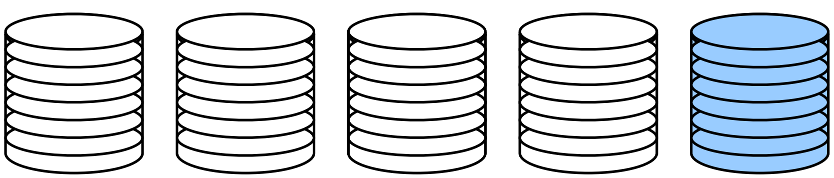
RAID Level 1:
Mirroring



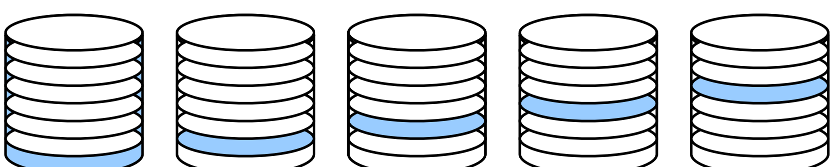
RAID Level 2:
Byte-interleaved, ECC



RAID Level 3:
Byte-interleaved, parity



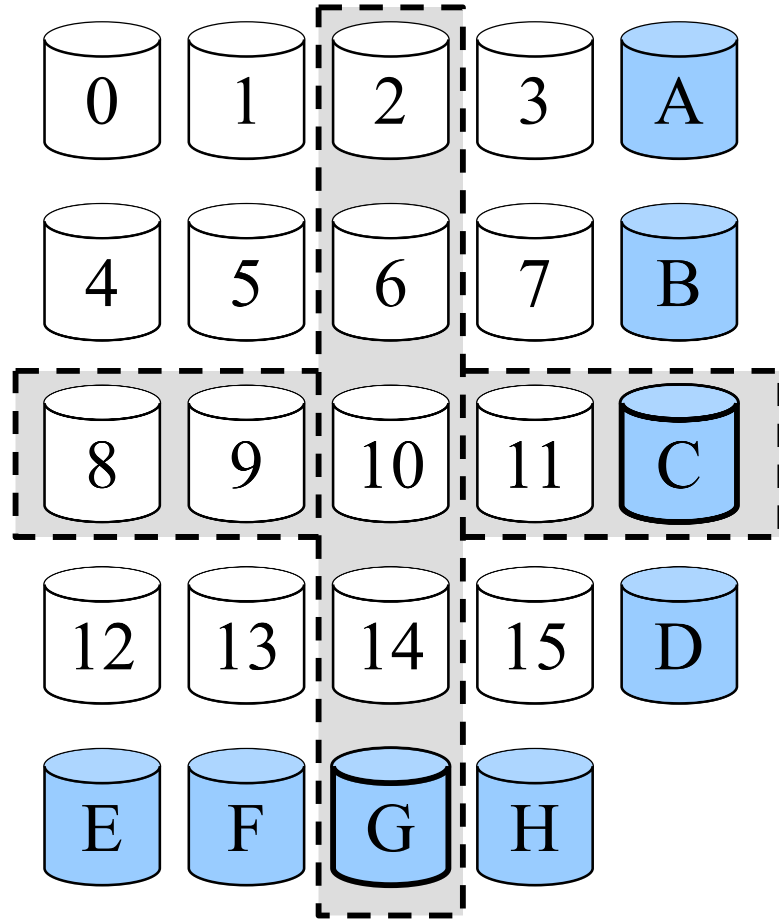
RAID Level 4:
Block-interleaved, parity

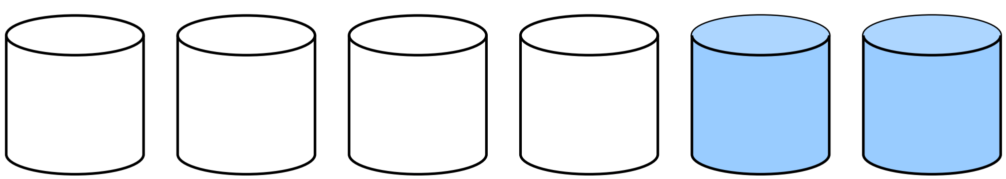


RAID Level 5:
Block-interleaved, distributed parity

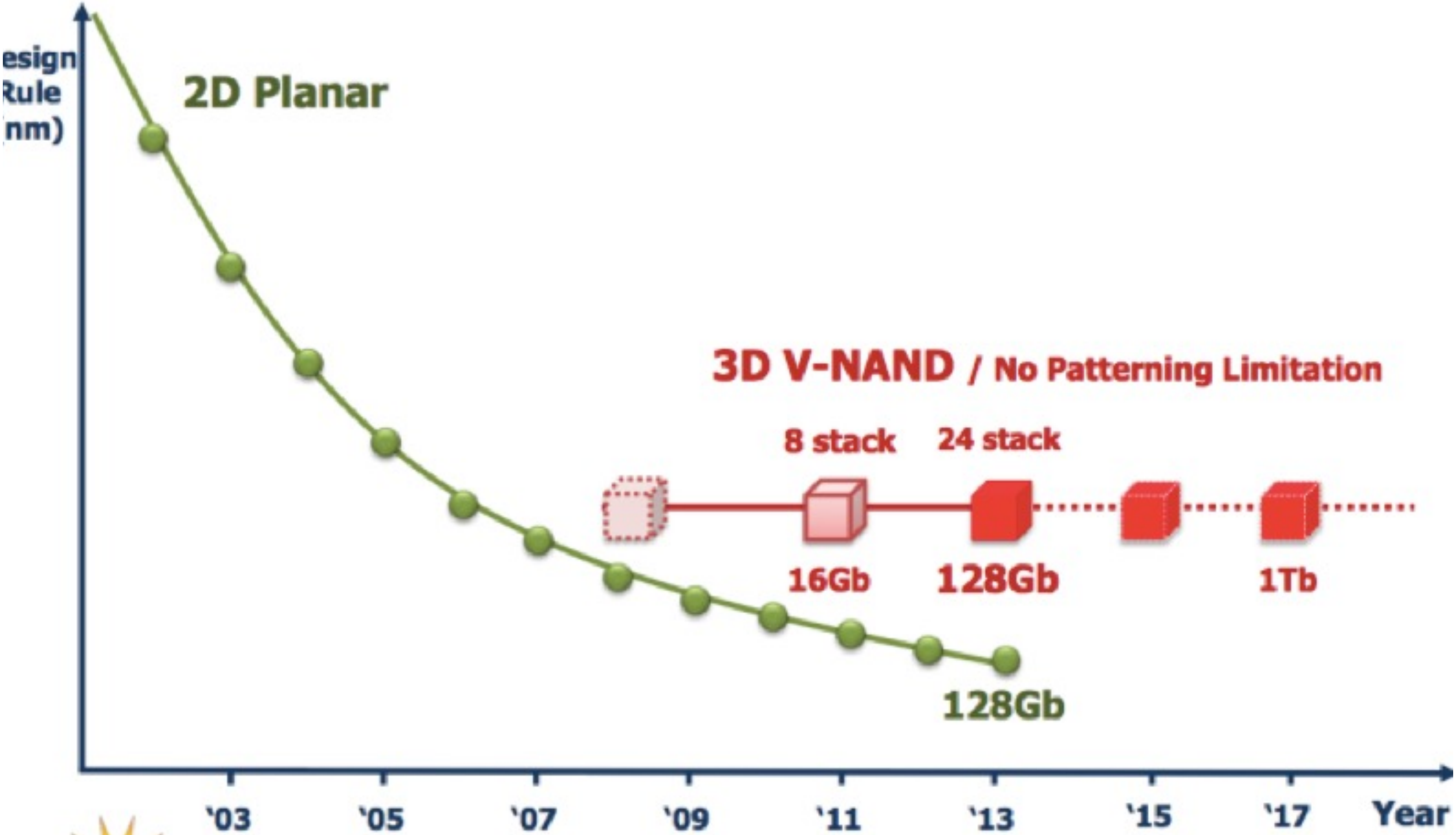
RAID Level 6 and Beyond

- Goals
  - Less computation and fewer updates per random write
  - Small amount of extra disk space
- Extended Hamming code
- 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



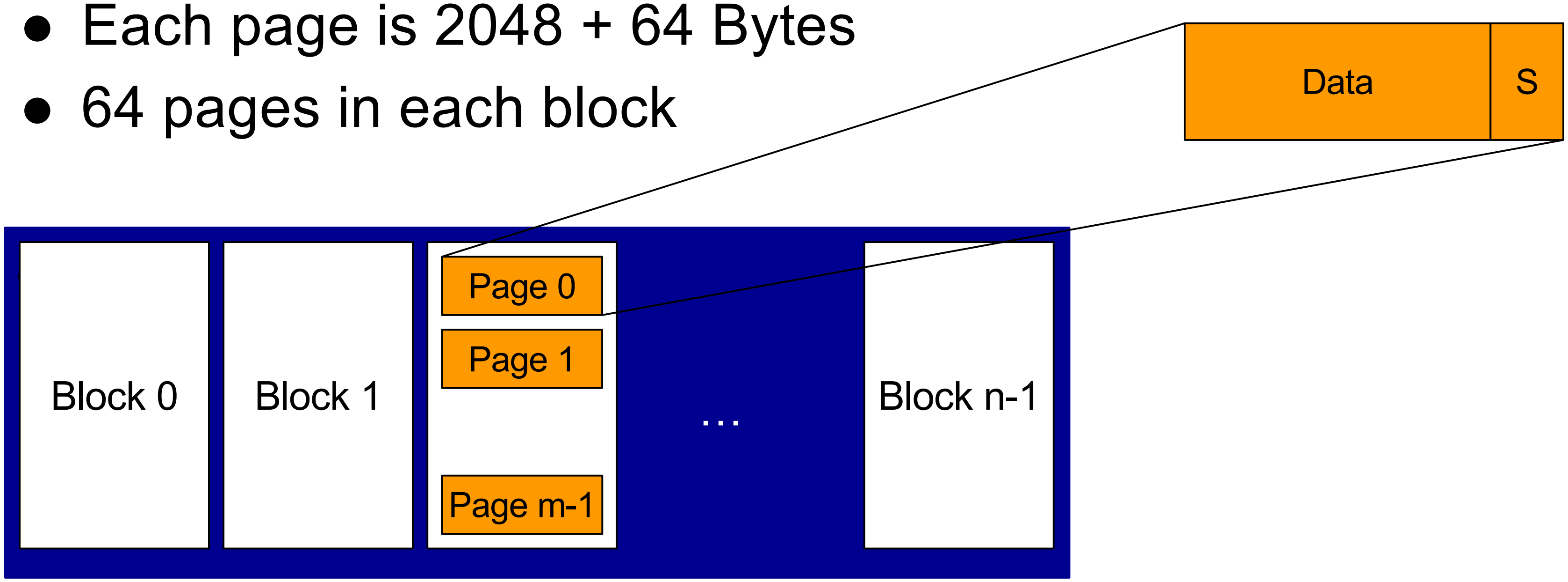


V-NAND Era for the Future



NAND Flash Memory

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



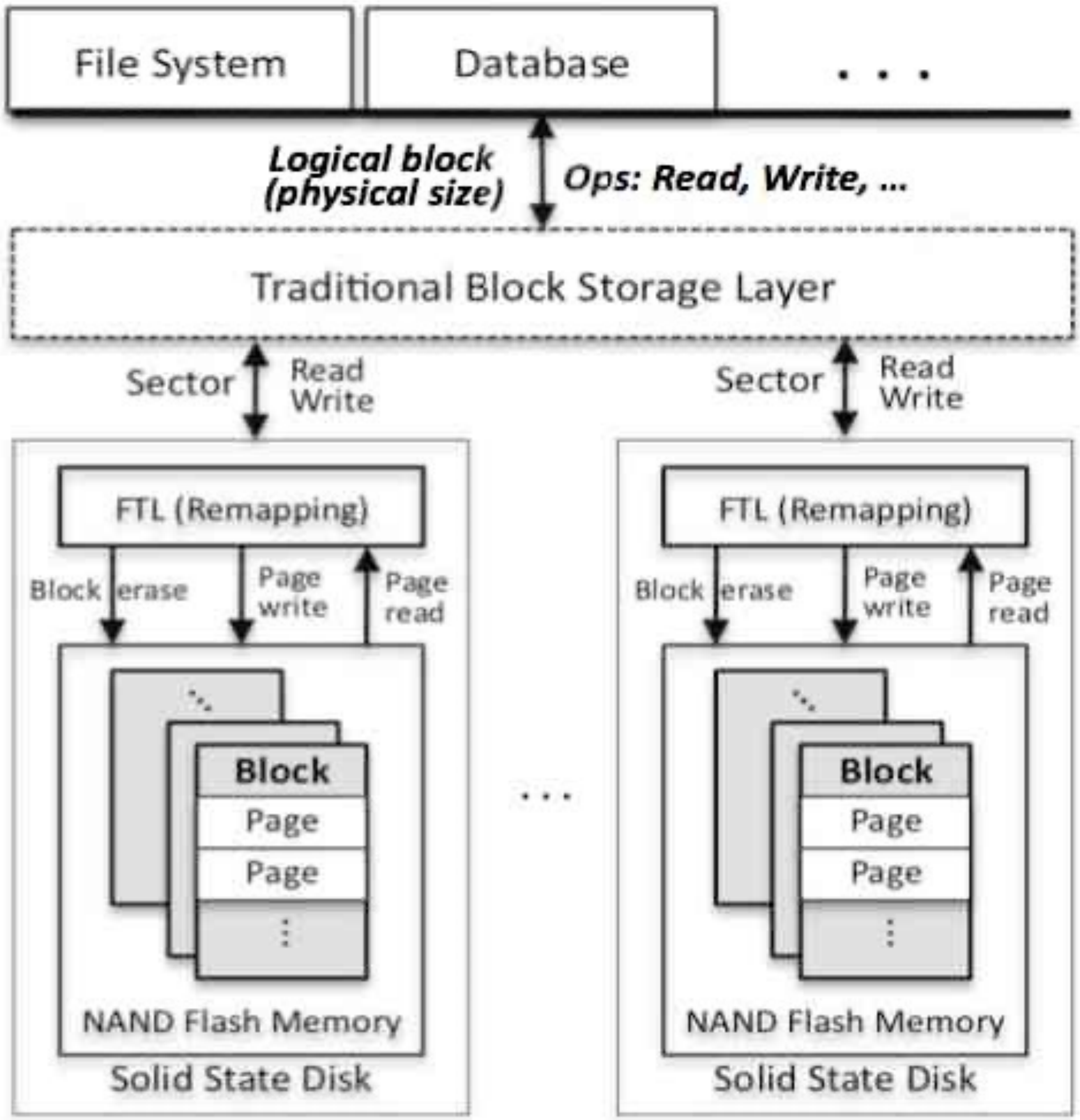
NAND Flash Memory Operations

- 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
  - Erasure blocks of 128-512KB are written into
- Solution: Flash Translation Layer (FTL)
  - Map virtual page to physical page address in flash controller
  - Keep erasing unused blocks
  - Garbage collect by copying live pages to new locations, and erasing large blocks
  - Remap to currently erased block to reduce latency

NAND Flash Lifetime

- Wear out limitations
  - ~50k to 100k writes / page (SLC – single level cell)
  - ~15k to 60k writes / page (MLC – multi-level cell)
- Wear Leveling:
  - Spread erases evenly across blocks, rather than using same block repeatedly
  - Remap pages that no longer work (like sector sparing on magnetic disks)
  - 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
  - ~10GB/s read
  - ~5GB/s write
  - ~25μs 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