CSE 487/587
Data Intensive Computing
Lecture 25: IO Technologies

Vipin Chaudhary

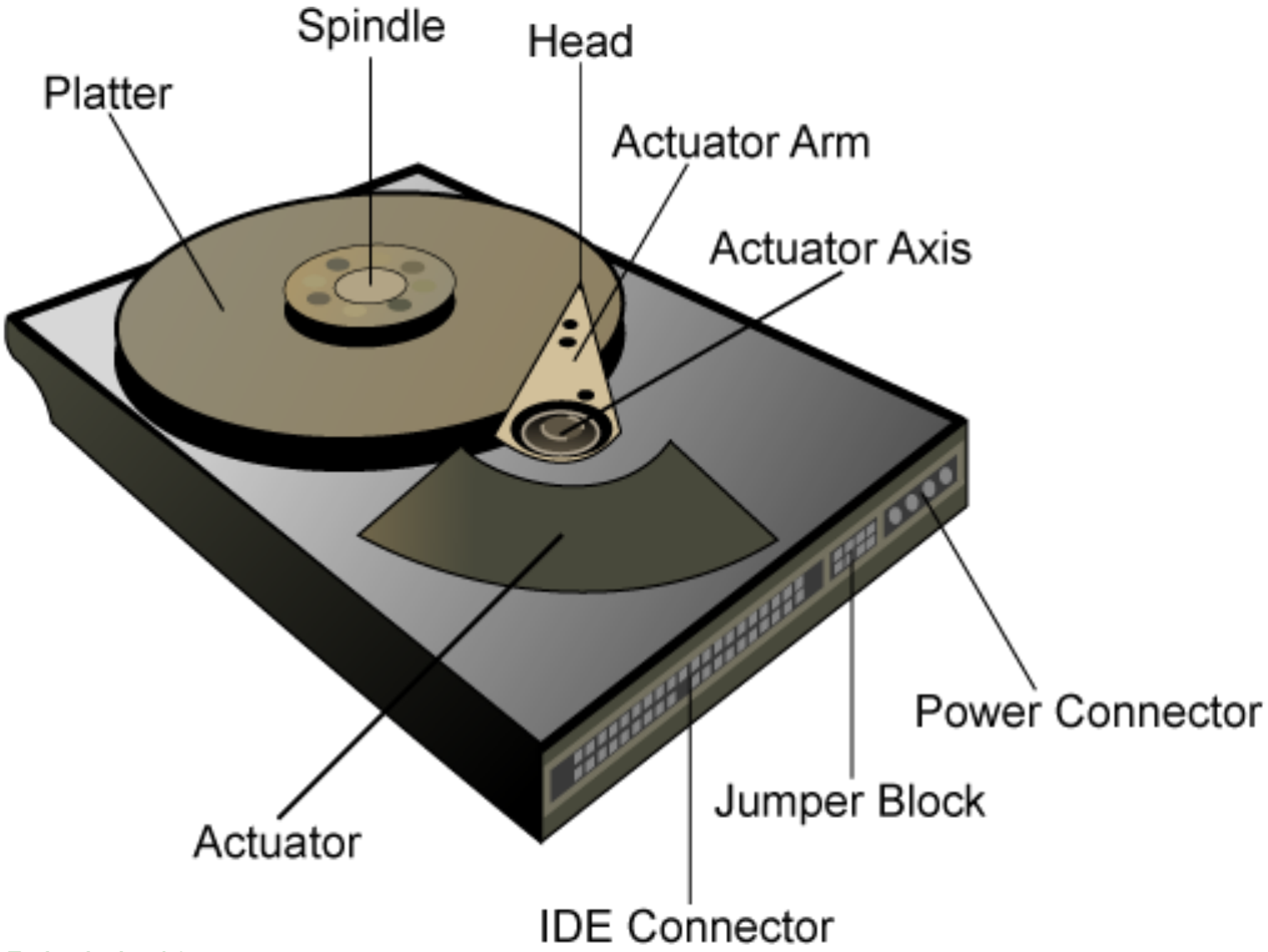
vipin@buffalo.edu

716.645.4740
305 Davis Hall

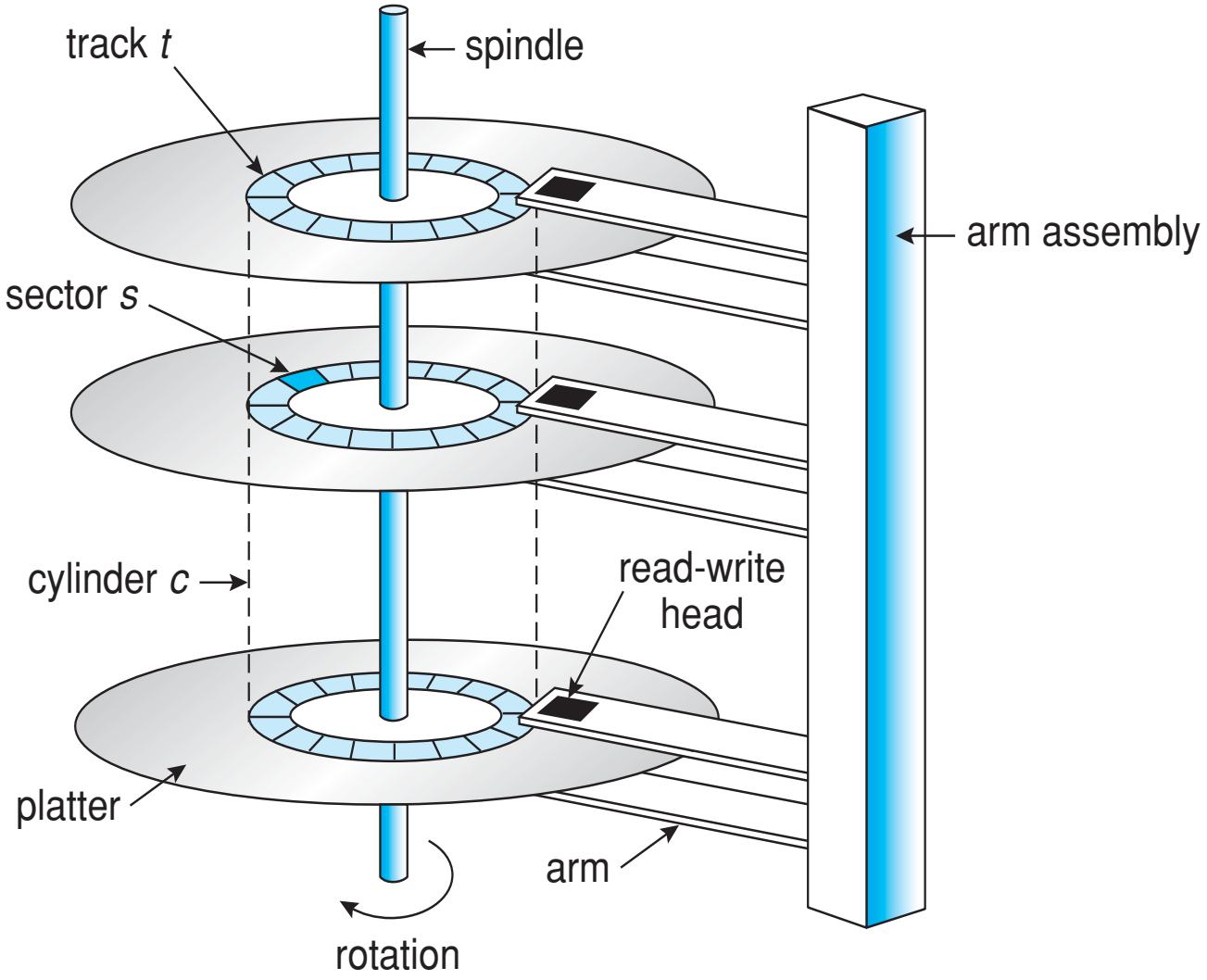
Overview of Lecture Series

- IO Technolgies
- Flash, SSD

Hard Drive Hardware



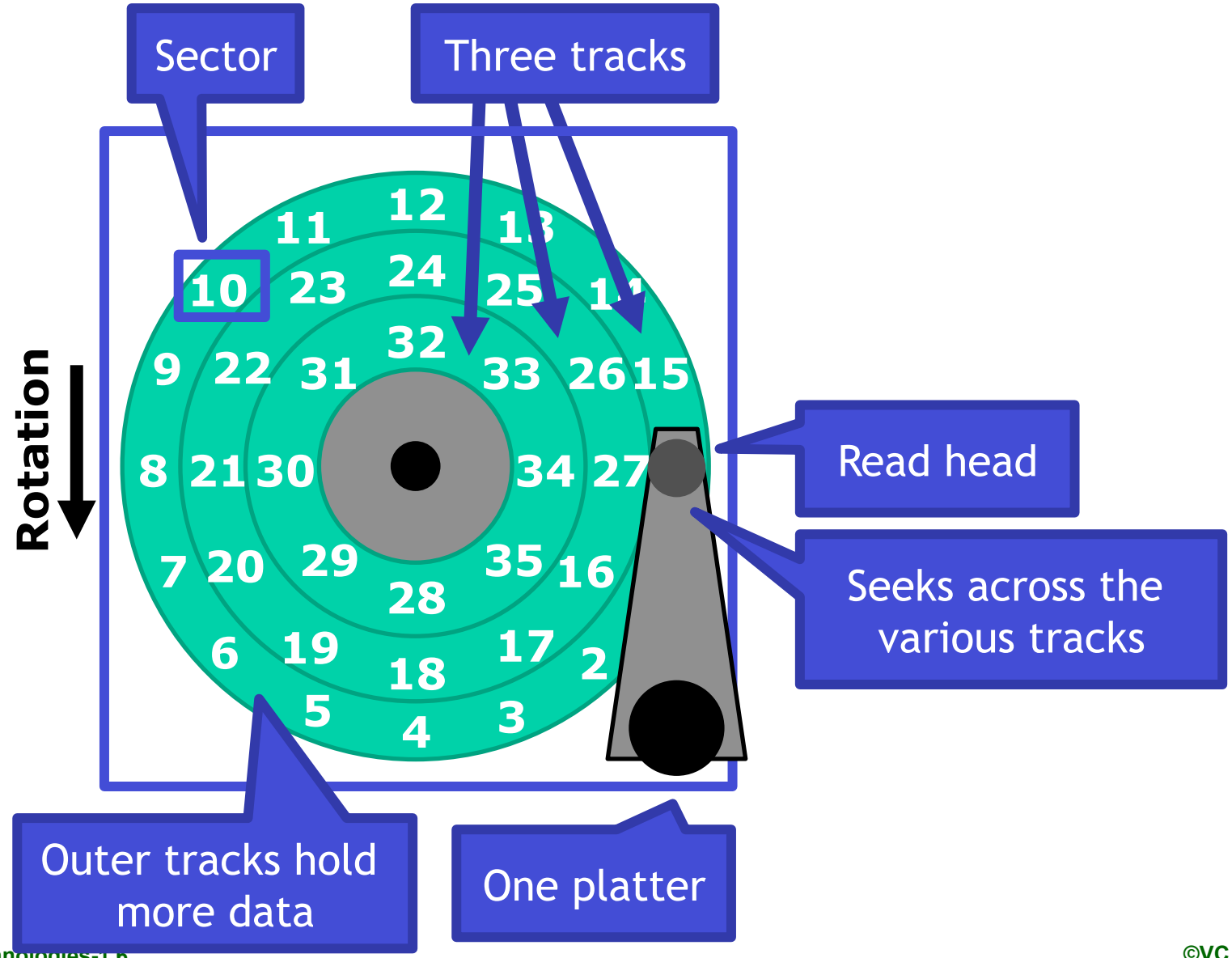
A Multi-Platter Disk



Addressing and Geometry

- Externally, hard drives expose a large number of sectors (blocks)
  - Typically 512 or 4096 bytes
  - Individual sector writes are atomic
  - Multiple sectors writes may be interrupted (torn write)
- Drive geometry
  - Sectors arranged into tracks
  - A cylinder is a particular track on multiple platters
  - Tracks arranged in concentric circles on platters
  - A disk may have multiple, double-sided platters
- Drive motor spins the platters at a constant rate
  - Measured in revolutions per minute (RPM)

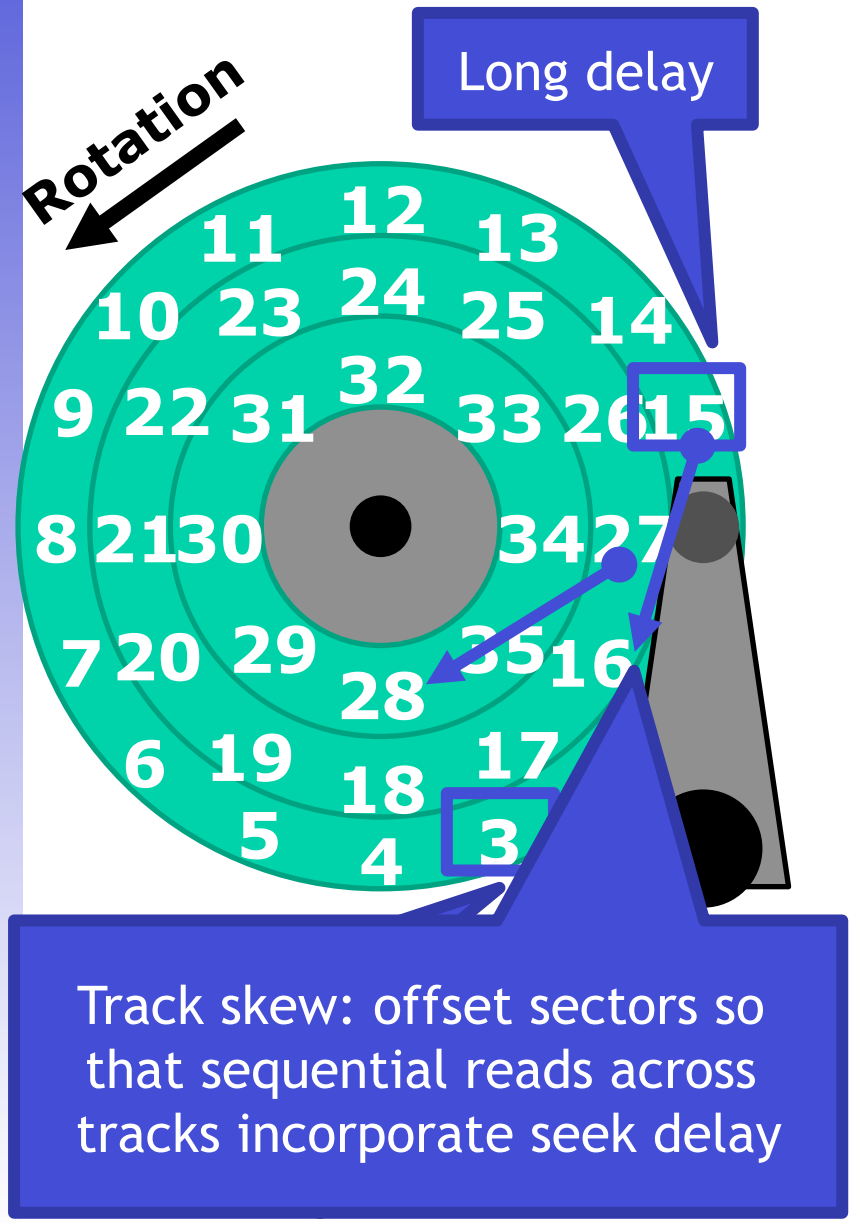
Geometry Example



Common Disk Interfaces

- ST-506 → ATA → IDE → SATA
  - Ancient standard
  - Commands (read/write) and addresses in cylinder/head/sector format placed in device registers
  - Recent versions support Logical Block Addresses (LBA)
- SCSI (Small Computer Systems Interface)
  - Packet based, like TCP/IP
  - Device translates LBA to internal format (e.g. c/h/s)
  - Transport independent
    - USB drives, CD/DVD/Bluray, Firewire
    - iSCSI is SCSI over TCP/IP and Ethernet

Types of Delay With Disks



Three types of delay

1. Rotational Delay
   - Time to rotate the desired sector to the read head
   - Related to RPM
2. Seek delay
   - Time to move the read head to a different track
3. Transfer time
   - Time to read or write bytes

How To Calculate Transfer Time

	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15000	7200
Avg. Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

Transfer time

$$T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$$

Assume we are transferring 4096 bytes

Cheetah

$$T_{I/O} = 4\ ms + 1 / (15000\ RPM / 60\ s/M / 1000\ ms/s) / 2 + (4096\ B / 125\ MB/s * 1000\ ms/s / 2^{20}\ MB/B)$$

$$T_{I/O} = 4\ ms + 2ms + 0.03125\ ms \approx 6\ ms$$

Barracuda

$$T_{I/O} = 9\ ms + 1 / (7200\ RPM / 60\ s/M / 1000\ ms/s) / 2 + (4096\ B / 105\ MB/s * 1000\ ms/s / 2^{20}\ MB/B)$$

$$T_{I/O} = 9\ ms + 4.17\ ms + 0.0372\ ms \approx 13.2\ ms$$

Sequential vs. Random Access

Rate of I/O

$R_{I/O}$ = transfer_size / $T_{I/O}$

Access Type	Transfer Size		Cheetah 15K.5	Barracuda
Random	4096 B	$T_{I/O}$	6 ms	13.2 ms
		$R_{I/O}$	0.66 MB/s	0.31 MB/s
Sequential	100 MB	$T_{I/O}$	800 ms	950 ms
		$R_{I/O}$	125 MB/s	105 MB/s
Max Transfer Rate			125 MB/s	105MB/s

Random I/O results in very poor disk performance!

Caching

- Many disks incorporate caches (track buffer)
  - Small amount of RAM (8, 16, or 32 MB)
- Read caching
  - Reduces read delays due to seeking and rotation
- Write caching
  - Write back cache: drive reports that writes are complete after they have been cached
    - Possibly dangerous feature. Why?
  - Write through cache: drive reports that writes are complete after they have been written to disk
- Today, some disks include flash memory for persistent caching (hybrid drives)

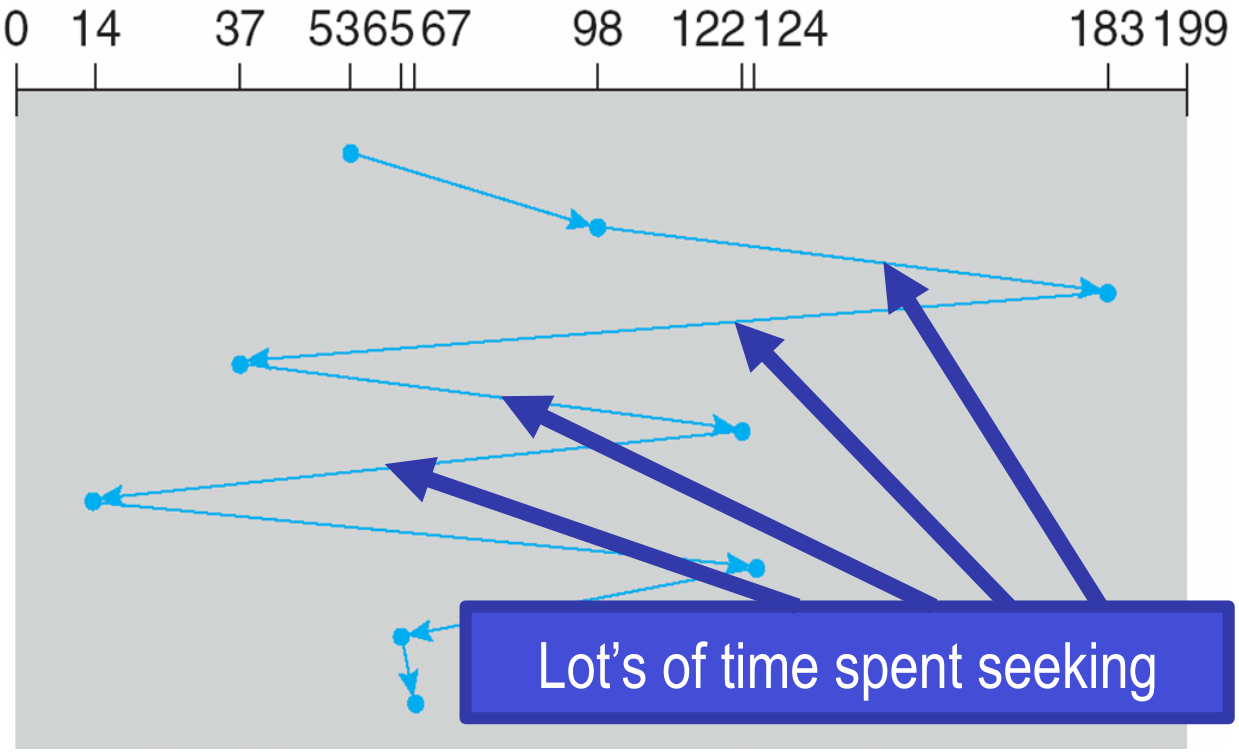
Disk Scheduling

- Caching helps improve disk performance
- But it can't make up for poor random access times
- Key idea: if there are a queue of requests to the disk, they can be reordered to improve performance
  - First come, first serve (FCFS)
  - Shortest seek time first (SSTF)
  - SCAN, otherwise know as the elevator algorithm
  - C-SCAN, C-LOOK, etc.

FCFS Scheduling

- Most basic scheduler, serve requests in order

- Head starts at block 53
- Queue: 98, 183, 37, 122, 14, 124, 65, 67

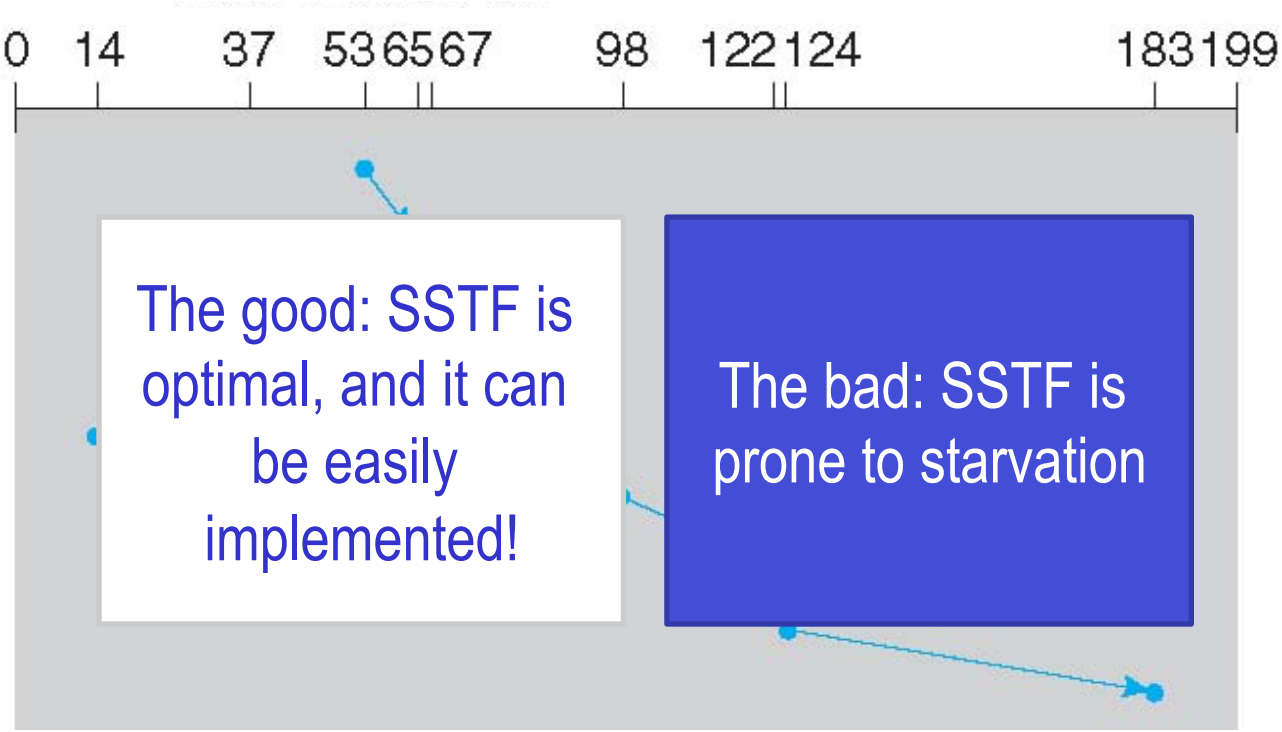


- Total movement: 640 cylinders

SSTF Scheduling

- Idea: minimize seek time by always selecting the block with the shortest seek time

- Head starts at block 53
- Queue: 98, 183, 37, 122, 14, 124, 65, 67



- Total movement: 236 cylinders

SCAN Example

- Head sweeps across the disk servicing requests in order

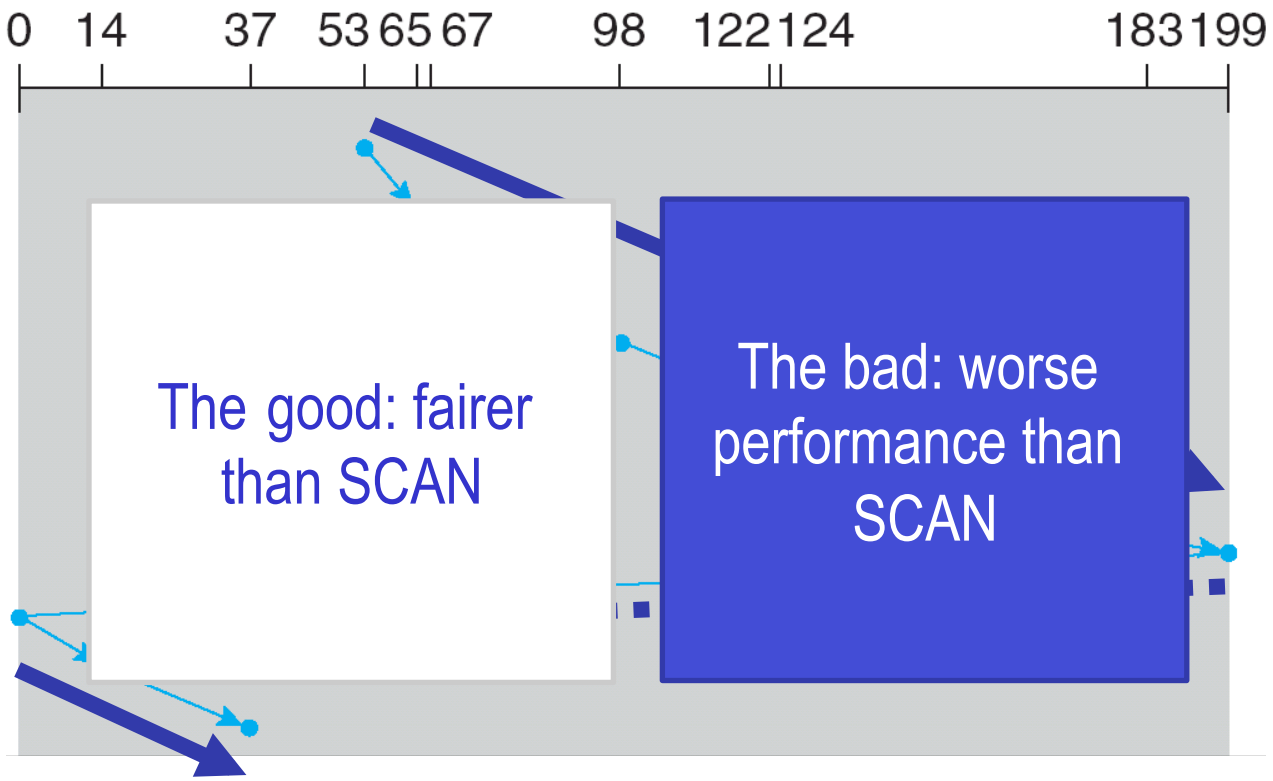
- Head starts at block 53
- Queue: 98, 183, 37, 122, 14, 124, 65, 67

- Total movement: 236 cylinders

C-SCAN Example

- Like SCAN, but only service requests in one direction

- Head starts at block 53
- Queue: 98, 183, 37, 122, 14, 124, 65, 67

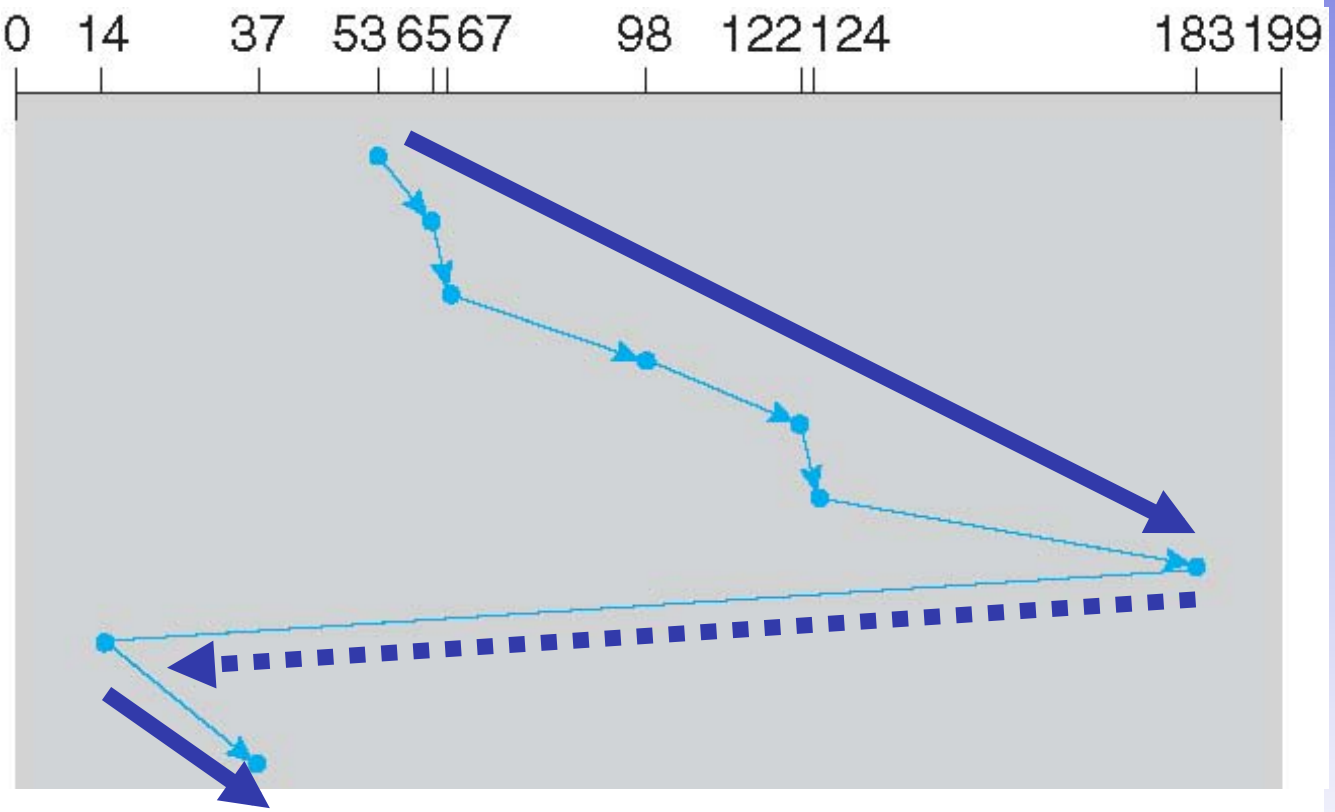


- Total movement: 382 cylinders

C-LOOK Example

- Peek at the upcoming addresses in the queue
- Head only goes as far as the last request

- Head starts at block 53
- Queue: 98, 183, 37, 122, 14, 124, 65, 67



- Total movement: 322 cylinders

Implementing Disk Scheduling

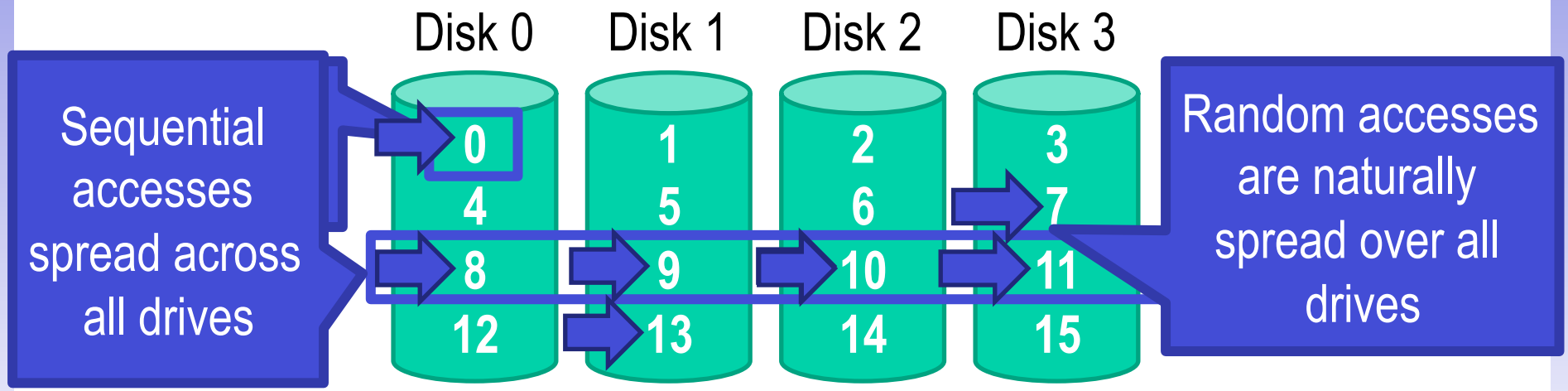
- Scheduling problems that take place in the kernel
  - Process scheduling
  - Page swapping
- Where should disk scheduling be implemented?
  - OS scheduling
    - OS can implement SSTF or LOOK by ordering the queue by LBA
    - However, the OS cannot account for rotation delay
  - On-disk scheduling
    - Disk knows the exact position of the head and platters
    - Can implement more advanced schedulers (SPTF)
    - But, requires specialized hardware and drivers

Command Queuing

- Feature where a disk stores of queue of pending read/write requests
  - Called Native Command Queuing (NCQ) in SATA
- Disk may reorder items in the queue to improve performance
  - E.g. batch operations to close sectors/tracks
- Supported by SCSI and modern SATA drives
- Tagged command queuing: allows the host to place constraints on command re-ordering

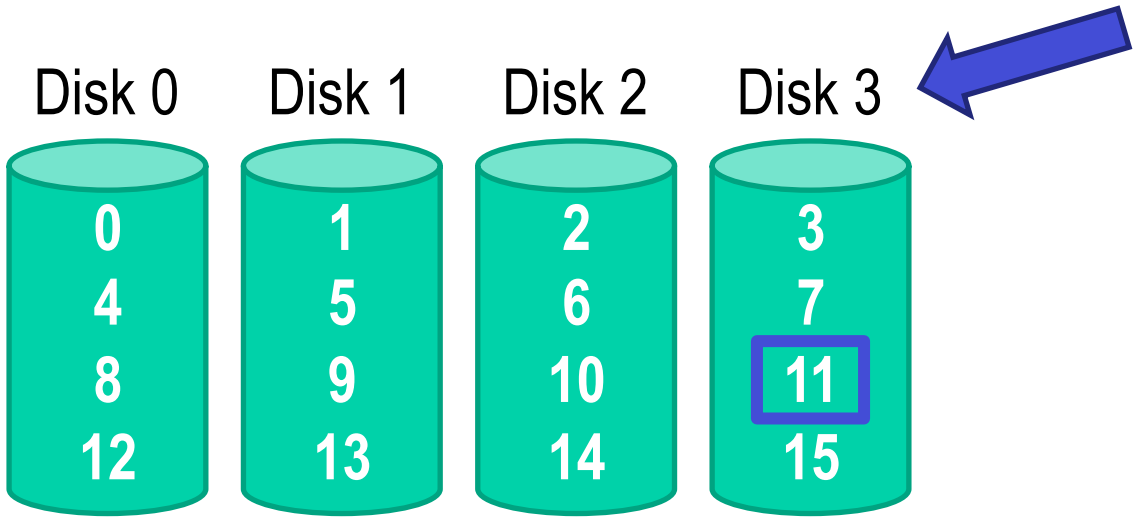
RAID 0: Striping

- Key idea: present an array of disks as a single large disk
- Maximize parallelism by striping data cross all *N* disks

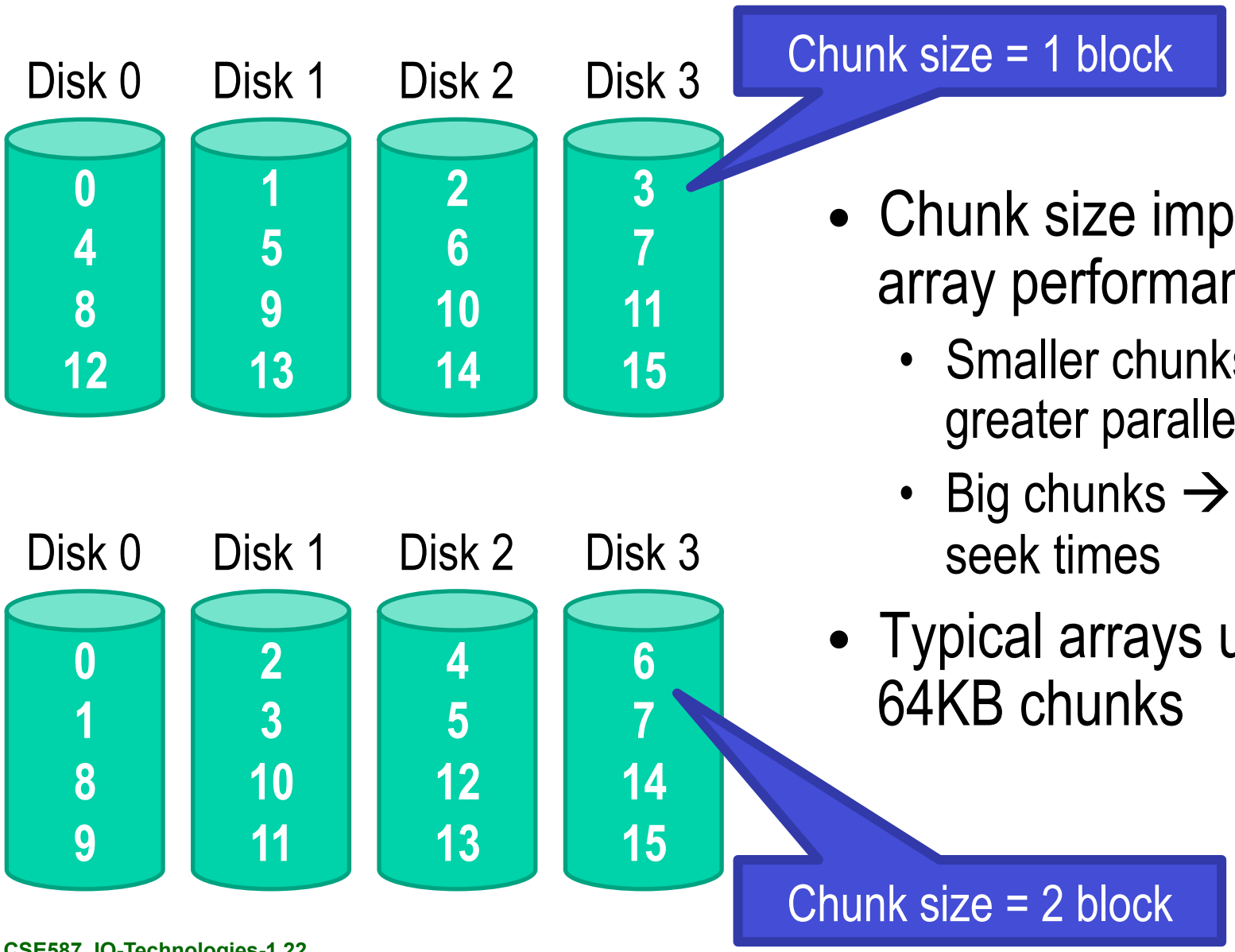


Addressing Blocks

- How do you access specific data blocks?
  - Disk = logical_block_number % number_of_disks
  - Offset = logical_block_number / number_of_disks
- Example: read block 11
  - 11 % 4 = Disk 3
  - 11 / 4 = Physical Block 2 (starting from 0)



Chunk Sizing



- Chunk size impacts array performance
  - Smaller chunks → greater parallelism
  - Big chunks → reduced seek times
- Typical arrays use 64KB chunks

Measuring RAID Performance (1)

- As usual, we focus on sequential and random workloads
- Assume disks in the array have **sequential** access time *S*
  - 10 MB transfer
  - *S* = *transfer_size* / *time_to_access*
  - 10 MB / (7 ms + 3 ms + 10 MB / 50 MB/s) = 47.62 MB/s

Average seek time	7 ms
Average rotational delay	3 ms
Transfer rate	50 MB/s

Measuring RAID Performance (2)

- As usual, we focus on sequential and random workloads
- Assume disks in the array have **random** access time *R*
  - 10 KB transfer
  - *R* = *transfer_size* / *time_to_access*
  - 10 KB / (7 ms + 3 ms + 10 KB / 50 MB/s) = 0.98 MB/s

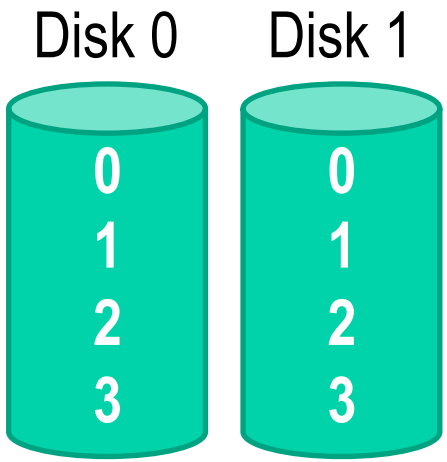
Average seek time	7 ms
Average rotational delay	3 ms
Transfer rate	50 MB/s

Analysis of RAID 0

- Capacity: *N*
  - All space on all drives can be filled with data
- Reliability: 0
  - If any drive fails, data is permanently lost
- Sequential read and write: *N* * *S*
  - Full parallelization across drives
- Random read and write: *N* * *R*
  - Full parallelization across all drives

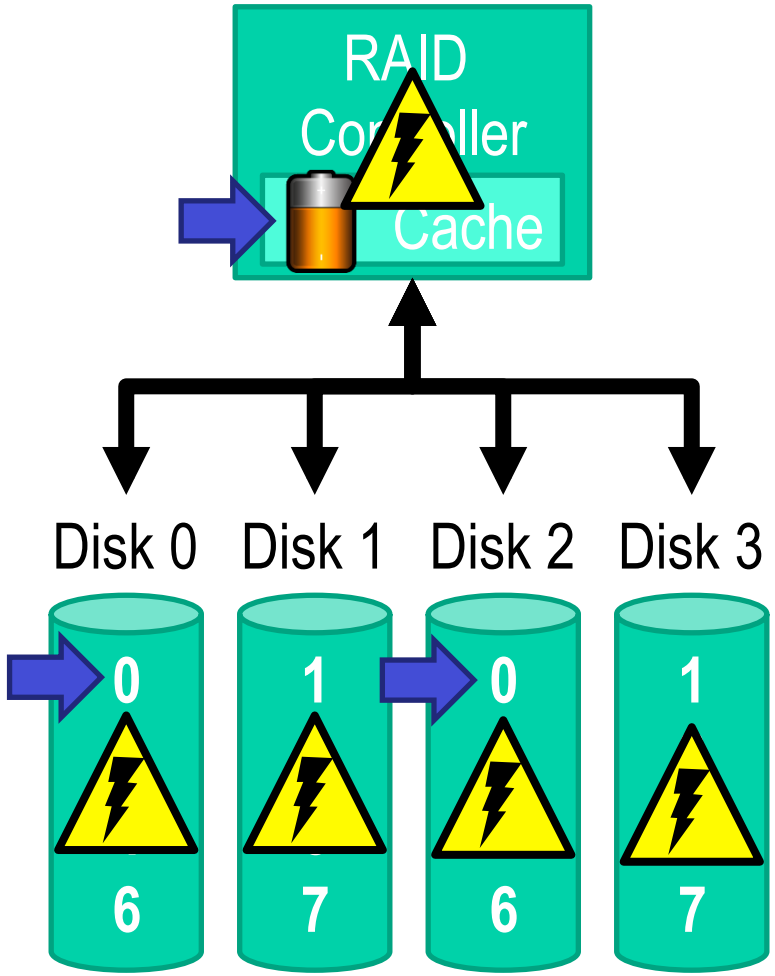
RAID 1: Mirroring

- RAID 0 offers high performance, but zero error recovery
- Key idea: make two copies of all data



The Consistent Update Problem

- Mirrored writes should be atomic
  - All copies are written, or none are written
- However, this is difficult to guarantee
  - Example: power failure
- Many RAID controllers include a write-ahead log
  - Battery backed, non-volatile storage of pending writes



Comparison of RAID Levels

- *N* – number of drives
- *S* – sequential access speed
- *R* – random access speed
- *D* – latency to access a single disk

		RAID 0	RAID 1	RAID 4	RAID 5
	Capacity	***N***	*N / 2*	*N – 1*	*N – 1*
	Reliability	*0*	***1* (maybe *N / 2*)**	*1*	*1*
Throughput	Sequential Read	***N * S***	*(N / 2) * S*	*(N – 1) * S*	*(N – 1) * S*
Throughput	Sequential Write	***N * S***	*(N / 2) * S*	*(N – 1) * S*	*(N – 1) * S*
Throughput	Random Read	***N * R***	***N * R***	*(N – 1) * R*	***N * R***
Throughput	Random Write	***N * R***	*(N / 2) * R*	*R / 2*	*(N / 4) * R*
Latency	Read	***D***	***D***	***D***	***D***
Latency	Write	***D***	***D***	*2 * D*	*2 * D*

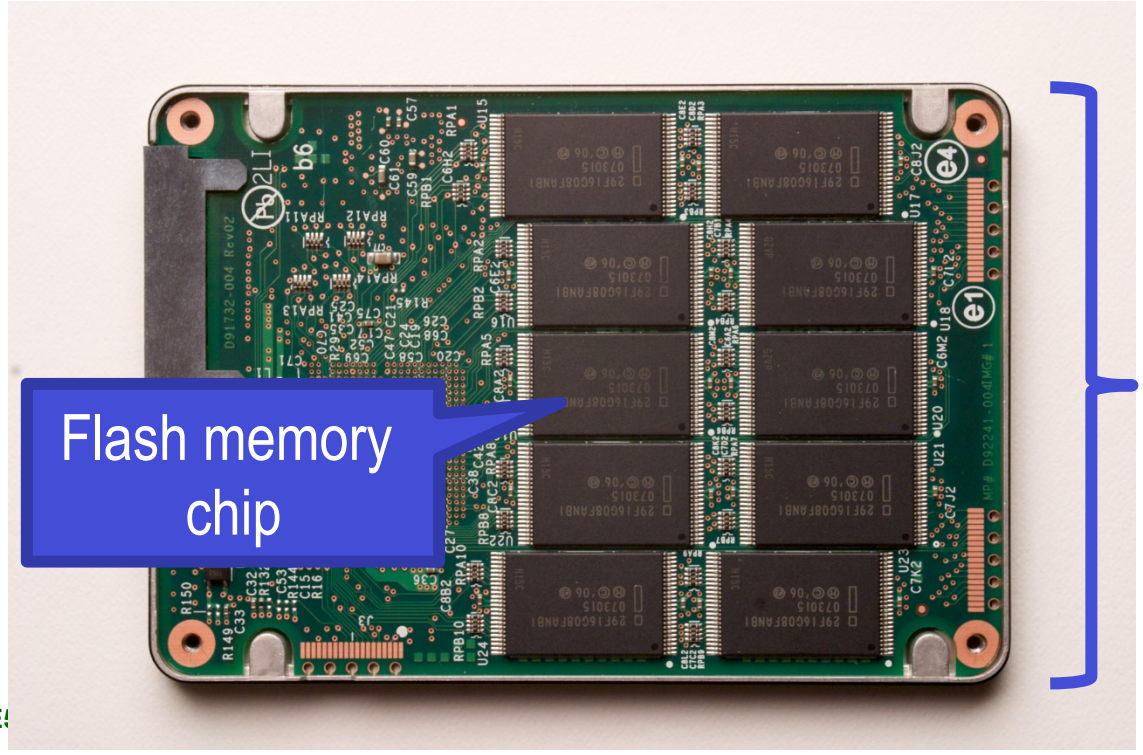
- Hard Drives
- RAID
- SSD

Beyond Spinning Disks

- Hard drives have been around since 1956
  - The cheapest way to store large amounts of data
  - Sizes are still increasing rapidly
- However, hard drives are typically the slowest component in most computers
  - CPU and RAM operate at GHz
  - PCI-X and Ethernet are GB/s
- Hard drives are not suitable for mobile devices
  - Fragile mechanical components can break
  - The disk motor is extremely power hungry

Solid State Drives

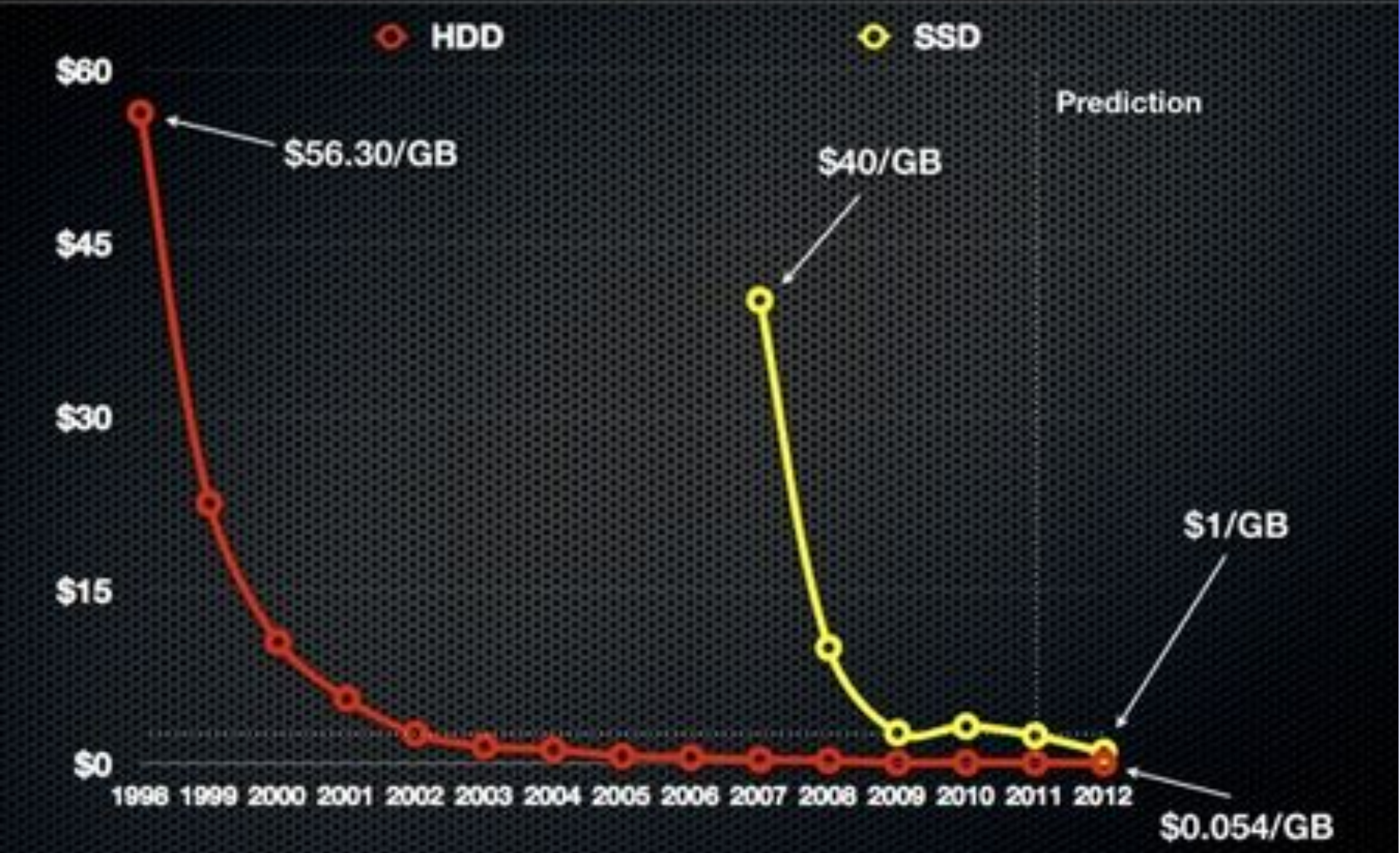
- NAND flash memory-based drives
  - High voltage is able to change the configuration of a floating-gate transistor
  - State of the transistor interpreted as binary data



Advantages of SSDs

- More resilient against physical damage
  - No sensitive read head or moving parts
  - Immune to changes in temperature
- Greatly reduced power consumption
  - No mechanical, moving parts
- Much faster than hard drives
  - >500 MB/s vs ~200 MB/s for hard drives
  - No penalty for random access
    - Each flash cell can be addressed directly
    - No need to rotate or seek
  - Extremely high throughput
    - Although each flash chip is slow, they are RAIDed

Average HDD and SSD prices in USD per gigabyte



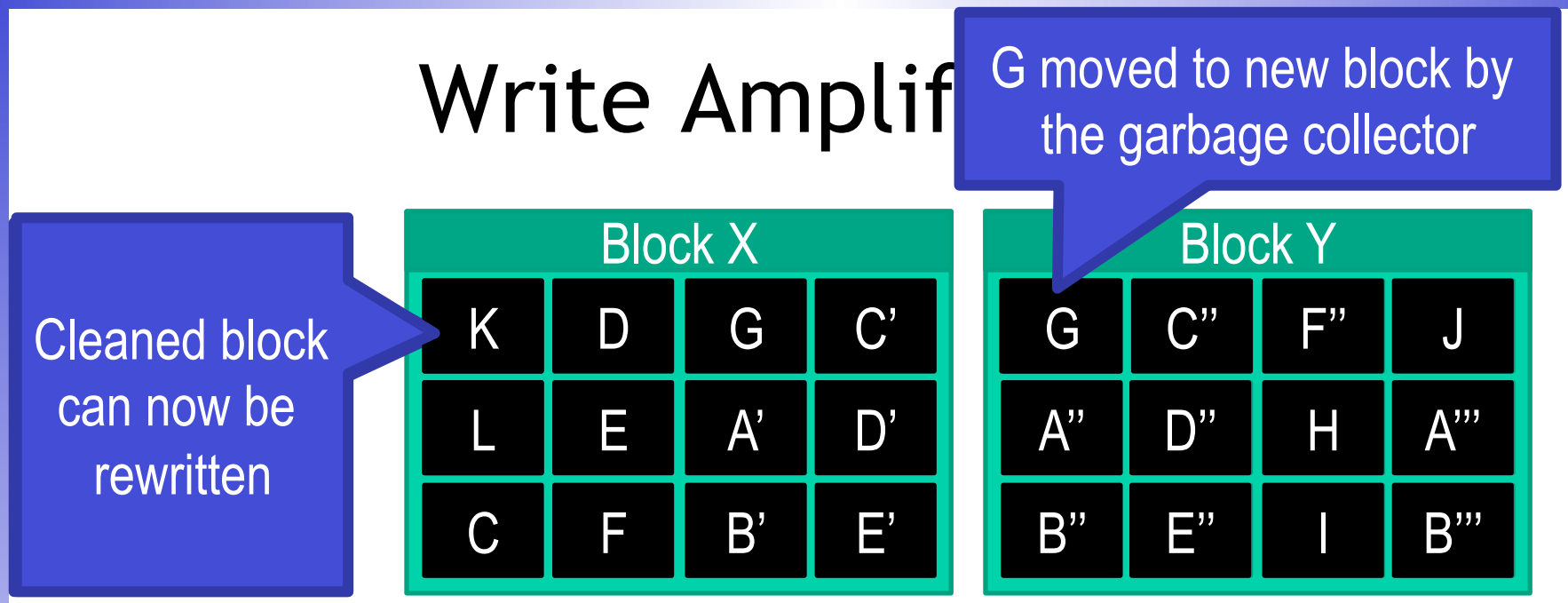
Data sources: Mkomo.com, Gartner, and Pingdom (December 2011)

www.pingdom.com

Challenges with Flash

- Flash memory is written in pages, but erased in blocks
  - Pages: 4 – 16 KB, Blocks: 128 – 256 KB
  - Thus, flash memory can become fragmented
  - Leads to the write amplification problem
- Flash memory can only be written a fixed number of times
  - Typically 3000 – 5000 cycles for MLC
  - SSDs use wear leveling to evenly distribute writes across all flash cells

Write Amplif



- Once all pages have been written, valid pages must be consolidated to free up space
- Write amplification: a write triggers garbage collection/ compaction
  - One or more blocks must be read, erased, and rewritten before the write can proceed

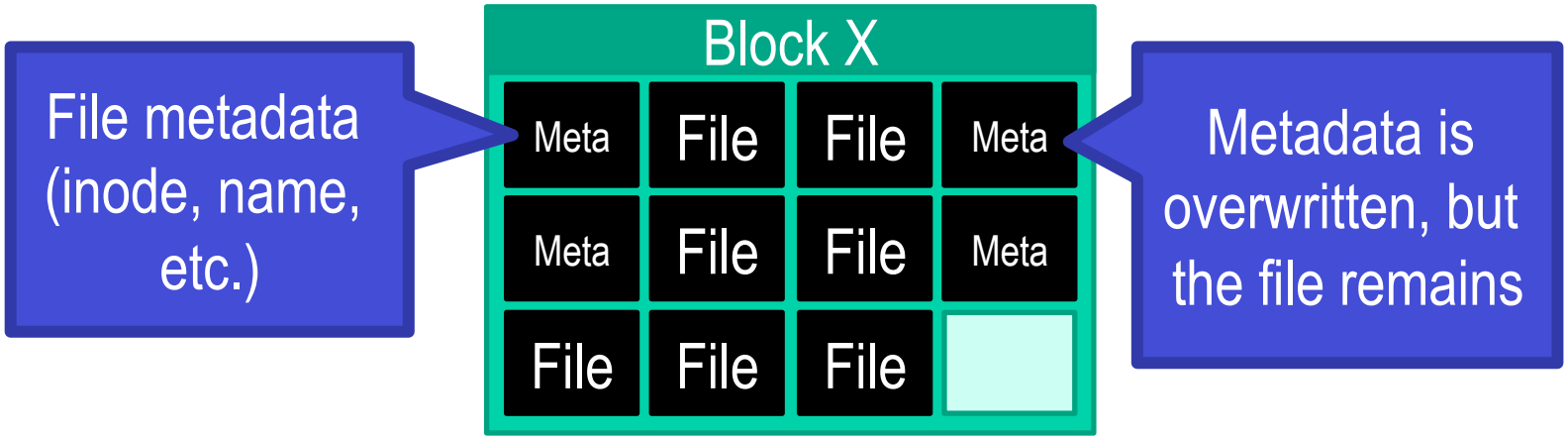
Garbage Collection

- Garbage collection (GC) is vital for the performance of SSDs
- Older SSDs had fast writes up until all pages were written once
  - Even if the drive has lots of “free space,” each write is amplified, thus reducing performance
- Many SSDs over-provision to help the GC
  - 240 GB SSDs actually have 256 GB of memory
- Modern SSDs implement background GC
  - However, this doesn’t always work correctly

The Ambiguity of Delete

- Goal: the SSD wants to perform background GC
  - But this assumes the SSD knows which pages are invalid
- Problem: most file systems don't actually delete data
  - On Linux, the "delete" function is unlink()
  - Removes the file meta-data, but not the file itself

Delete Example

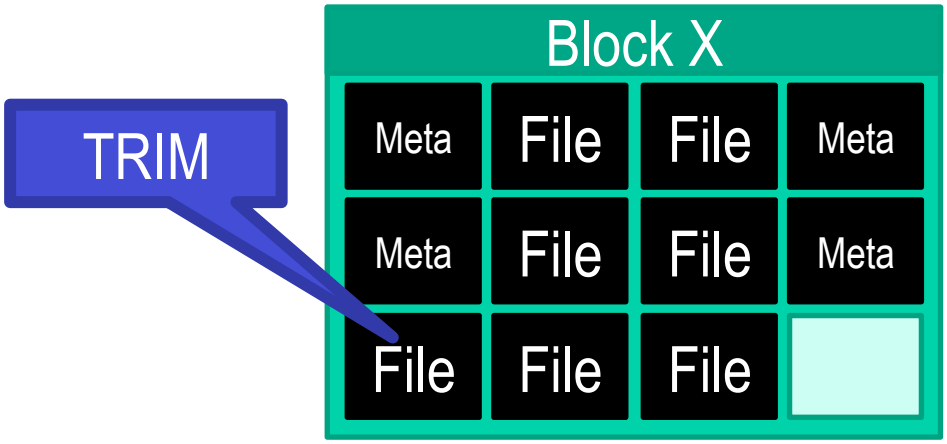


1. File is written to SSD
2. File is deleted
3. The GC executes
   - 9 pages look valid to the SSD
   - The OS knows only 2 pages are valid

- Lack of explicit delete means the GC wastes effort copying useless pages
- Hard drives are not GCed, so this was never a problem

TRIM

- New SATA command TRIM (SCSI – UNMAP)
  - Allows the OS to tell the SSD that specific LBAs are invalid, may be GCed



- OS support for TRIM
  - Win 7, OSX Snow Leopard, Linux 2.6.33, Android 4.3
- Must be supported by the SSD firmware

Wear Leveling

- Recall: each flash cell wears out after several thousand writes
- SSDs use wear leveling to spread writes across all cells
  - Typical consumer SSDs should last ~5 years

Flash Wear Leveling

Dynamic Wear Leveling:

- Controller selects new free data block before write
- Mapping points to new block location after write

Disadvantage:

- Static blocks where data do not change (eg. OS file) wears slower than other blocks
- Reduce lifespan of the flash drive

Flash Wear Leveling

Static Wear Leveling:

- Works the same as dynamic wear leveling
- Static blocks that do not change are periodically moved

Disadvantage:

- Moving static blocks takes time

Flash Wear Leveling

Dynamic vs. Static

	Static	Dynamic
Performance	Slower	Faster
Complexity	More Complex	Less Complex
Endurance	Longer	Shorter

What to use? Use Both!

W xamples

If the GC runs now, page G must be copied

Dynamic Wear Leveling

Wait as long as possible before garbage collecting

Block X			
K	D	G	C’
L	E	A’	D’
C	F	B’	E’

Block Y			
F’	C”	F”	G’
A”	D”	H	A”’
B”	E”	I	B”’

Static Wear Leveling

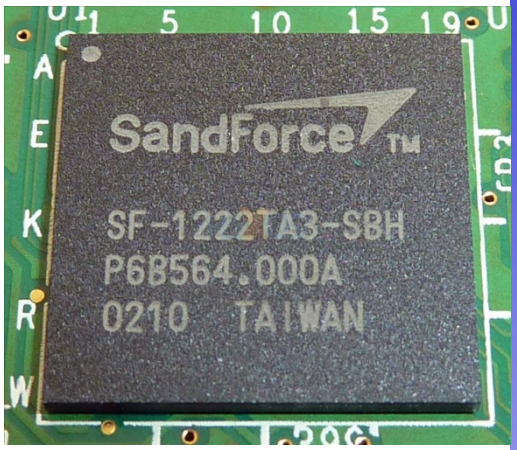
Blocks with long lived data receive less wear

Block X			
M*	D	G	J
N*	E	H	K
O*	F	I	L

Block Y			
A	D	G	J
B	E	H	K
C	F	I	L

SSD controller periodically swap long lived data to different blocks

SSD Controllers



- SSDs are extremely complicated internally
- All operations handled by the SSD controller
  - Maps LBAs to physical pages
  - Keeps track of free pages, controls the GC
  - May implement background GC
  - Performs wear leveling via data rotation
- Controller performance is crucial for overall SSD performance

Wear Leveling

Dynamic wear leveling

- Uses a map to link logical block addresses (LBAs) from the OS to the physical flash memory.
- Each time the OS writes replacement data, the map is updated so the original physical block is marked as invalid data, and a new block is linked to that map entry.
- Each time a block of data is re-written to the flash memory, it is written to a new location.
- However, flash memory blocks that never get replacement data would sustain no additional wear, thus the name comes from only the dynamic data being recycled.
- Such a device may last longer than one with no wear leveling, but there are blocks still remaining as active that will go unused when a device is no longer operable

Static wear leveling

- Also uses a map to link the LBA to physical memory addresses.
- Static wear leveling works the same as dynamic wear leveling except the static blocks that do not change are periodically moved so that these low usage cells are able to be used by other data.
- This rotational effect enables an SSD to continue to operate until most of the blocks are near their end of life.
- This is also sometimes referred to as global wear leveling, as the entire image is leveled.

Item	Static	Dynamic
Endurance	Longer life expectancy	Shorter life expectancy
Performance	Slower	Faster
Design Complexity	More complex	Less complex
Typical Use	SSDs	USB Flash Drives

Flavors of NAND Flash Memory

Multi-Level Cell (MLC)

- Multiple bits per flash cell
  - For two-level: 00, 01, 10, 11
  - 2, 3, and 4-bit MLC is available
- Higher capacity and cheaper than SLC flash
- Lower throughput due to the need for error correction
- 3000 – 5000 write cycles
- Consumes more power

Consumer-grade drives

Single-Level Cell (SLC)

- One bit per flash cell
  - 0 or 1
- Lower capacity and more expensive than MLC flash
- Higher throughput than MLC
- 10000 – 100000 write cycles

Expensive, enterprise drives

Triple-Level Cell (TLC)

Flash File System

- While a block device layer can emulate a disk drive so that a general-purpose file system can be used on a flash-based storage device, this is suboptimal for several reasons:
  - Erasing blocks:
    - flash memory blocks have to be explicitly erased before they can be written to.
    - time taken to erase blocks can be significant, thus it is beneficial to erase unused blocks while the device is idle.
  - Random access:
    - general-purpose file systems are optimized to avoid disk seeks whenever possible, due to the high cost of seeking.
    - Flash memory devices impose no seek latency.
  - Wear leveling:
    - flash memory devices tend to wear out when a single block is repeatedly overwritten; flash file systems are designed to spread out writes evenly.

You are given a storage system with 64 disks. It is a RAID 6 system. There is a disk failure. The IO capacity within the system is 84 Gb/s. With a disk failure, all the other disks need to rebuild the RAID disk so that involves data from all other 62 disks. Assume the bandwidth of each disk to be 100 Mbytes/s. Assume that regular storage operations require 40 Gbits/s.

(a) How long does it take to rebuild the RAID if each disk is 1TB.

(i) 62 disks will generate 62 TB of data. Amount of bandwidth remaining in the system after regular operations is (84-40) = 44 Gb/s. So, we need to transfer 62 TB at 44 Gb/s. That will take 62*8*1000/44 = 11272 s = 3.13hrs. **[CORRECT ?]**

How much data can be gotten out of all 62 disk? 62*100*8 = 49.6 Gb/s. So the disks can spit out more data than what is allowed by the internal network.

(ii) Since all the data needs to be written to one disk, the bandwidth of writing to that disk is the bottleneck. So, 1 TB has to be written at 100 MB/s which takes 1000000/100 = 10000s = 2.78 hrs. **[CORRECT ?]**

You are given a storage system with 64 disks. It is a RAID 6 system. There is a disk failure. The IO capacity within the system is 84 Gb/s. With a disk failure, all the other disks need to rebuild the RAID disk so that involves data from all other 62 disks. Assume the bandwidth of each disk to be 100 Mbytes/s. Assume that regular storage operations require 40 Gbits/s.

(b) How long does it take to rebuild the RAID if each disk is 4TB.

(i) 62 disks will generate 62*4 TB of data. Amount of bandwidth remaining in the system after regular operations is (84-40) = 44 Gb/s. So, we need to transfer 62*4 TB at 44 Gb/s. That will take 62*8*1000*4/44 = 11272*4 s = 45088 s = 3.13 * 4 hrs = 12.52 hrs.

[CORRECT]

(ii) *40000s = 11.11 hrs (done similarly as (a))*

You are given a storage system with 64 disks. It is a RAID 6 system. There is a disk failure. The IO capacity within the system is 84 Gb/s. With a disk failure, all the other disks need to rebuild the RAID disk so that involves data from all other 62 disks. Assume the bandwidth of each disk to be 100 Mbytes/s. Assume that regular storage operations require 40 Gbits/s.

(c) How long does it take to rebuild the RAID if each disk is 8TB.

(i) 62 disks will generate 62*8 TB of data. Amount of bandwidth remaining in the system after regular operations is (84-40) = 44 Gb/s. So, we need to transfer 62*8 TB at 44 Gb/s. That will take 62*8*1000*8/44 = 11272*8 s = 90176 s = 3.13 * 8 hrs = 25.04 hrs. **[CORRECT]**

(ii) *80000s = 22.22 hrs* (*done similarly as (a)*)

You are given a storage system with 64 disks. It is a RAID 6 system. There is a disk failure. The IO capacity within the system is 84 Gb/s. With a disk failure, all the other disks need to rebuild the RAID disk so that involves data from all other 62 disks. Assume the bandwidth of each disk to be 100 Mbytes/s. Assume that regular storage operations require 40 Gbits/s.

(d) How long does it take to rebuild the RAID if each disk is 8TB assuming the storage is idle (you can use the entire IO capacity for RAID rebuild).

If storage is idle then total io bandwidth of 84 Gb/s is available for RAID rebuild. So, time taken is 62*8*1000*8/84 = 47238 s = 13 hrs.

But, IO out of disks is only 49.6 Gb/s and thus the RAID rebuild time will be 62*8*1000*8/49.6 = 80000 s = 22.22 hrs **[CORRECT]**

Although the entire system is available, it is still bottlenecked by 100 MB/s. So, time taken is 80000s = 22.22 hrs **[ALSO CORRECT]**

You are given a storage system with 64 disks. It is a RAID 6 system. There is a disk failure. The IO capacity within the system is 84 Gb/s. With a disk failure, all the other disks need to rebuild the RAID disk so that involves data from all other 62 disks. Assume the bandwidth of each disk to be 100 Mbytes/s. Assume that regular storage operations require 40 Gbits/s.

(e) Now assume that you create 8 LUNs each with 8 disks. Each LUN has 8 disks with RAID 6. What is the new usable capacity of the storage box?

Number of LUNs with 64 disks = 8. Usable capacity of each LUN = 6*(size of disk). Total usable capacity = 6*8*(size of disk)
1TB => 6*8*1 TB = 48 TB
4TB => 6*8*4 TB = 192 TB
8TB => 6*8*8 TB = 384 TB

You are given a storage system with 64 disks. It is a RAID 6 system. There is a disk failure. The IO capacity within the system is 84 Gb/s. With a disk failure, all the other disks need to rebuild the RAID disk so that involves data from all other 62 disks. Assume the bandwidth of each disk to be 100 Mbytes/s. Assume that regular storage operations require 40 Gbits/s.
Now assume that you create 8 LUNs each with 8 disks. Each LUN has 8 disks with RAID 6.

(f) How long does it take to rebuild the RAID if each disk is 1TB?

One failure only affects the rebuild of one LUN. So, 6 disks will generate 6 TB of data. Amount of bandwidth remaining in the system after regular operations is (84-40) = 44 Gb/s. So, we need to transfer 6 TB at 44 Gb/s. That will take 6*8*1000/44 = 1090.9 s = 0.303 hrs.

How much data can be gotten out of all 6 disk? 6*100*8 = 4.8 Gb/s. So the disks are bottlenecks and so real time taken is 6*8*1000/4.8 = 10000s = 2.78 hrs. **[CORRECT]**

Since all the data needs to be written to one disk, the bandwidth of writing to that disk is the bottleneck. So, 1 TB has to be written at 100 MB/s which takes 1000000/100 = 10000s = 2.78 hrs

You are given a storage system with 64 disks. It is a RAID 6 system. There is a disk failure. The IO capacity within the system is 84 Gb/s. With a disk failure, all the other disks need to rebuild the RAID disk so that involves data from all other 62 disks. Assume the bandwidth of each disk to be 100 Mbytes/s. Assume that regular storage operations require 40 Gbits/s.
Now assume that you create 8 LUNs each with 8 disks. Each LUN has 8 disks with RAID 6.

(g) How long does it take to rebuild the RAID if each disk is 4TB?

6*4*8*1000/4.8 = 40000 s = 11.11 hrs **[CORRECT]**
4 TB TB has to be written at 100 MB/s which takes 4000000/100 = 40000s = 11.11 hrs

(h) How long does it take to rebuild the RAID if each disk is 8TB?

6*8*8*1000/4.8 = = 80000 s = 22.22 hrs **[CORRECT]**
8 TB TB has to be written at 100 MB/s which takes 8000000/100 = 80000s = 22.22 hrs

(i) How long does it take to rebuild the RAID if each disk is 8TB assuming the storage is idle (you can use the entire IO capacity for RAID rebuild).

6*8*8*1000/4.8 = 80000 s = 22.22 hrs **[CORRECT]**
8 TB TB has to be written at 100 MB/s which takes 8000000/100 = 80000s = 22.22 hrs

You are given a storage system with 64 disks. It is an erasure coded system with 8 LUNs each with 8 disks. The EC used is 6+2. There is a disk failure.

Is it any different from RAID 6 in performance?
Is it any different from RAID 6 in fault-tolerance?
Is it any worse?

The IO capacity within the system is 84 Gb/s. With a disk failure, all the other disks need to rebuild the EC disk so that involves data from other disks in a LUN. Assume the bandwidth of each disk to be 100 Mbytes/s. Assume that regular storage operations require 40 Gbits/s.

(j) Assume 1 TB disks