

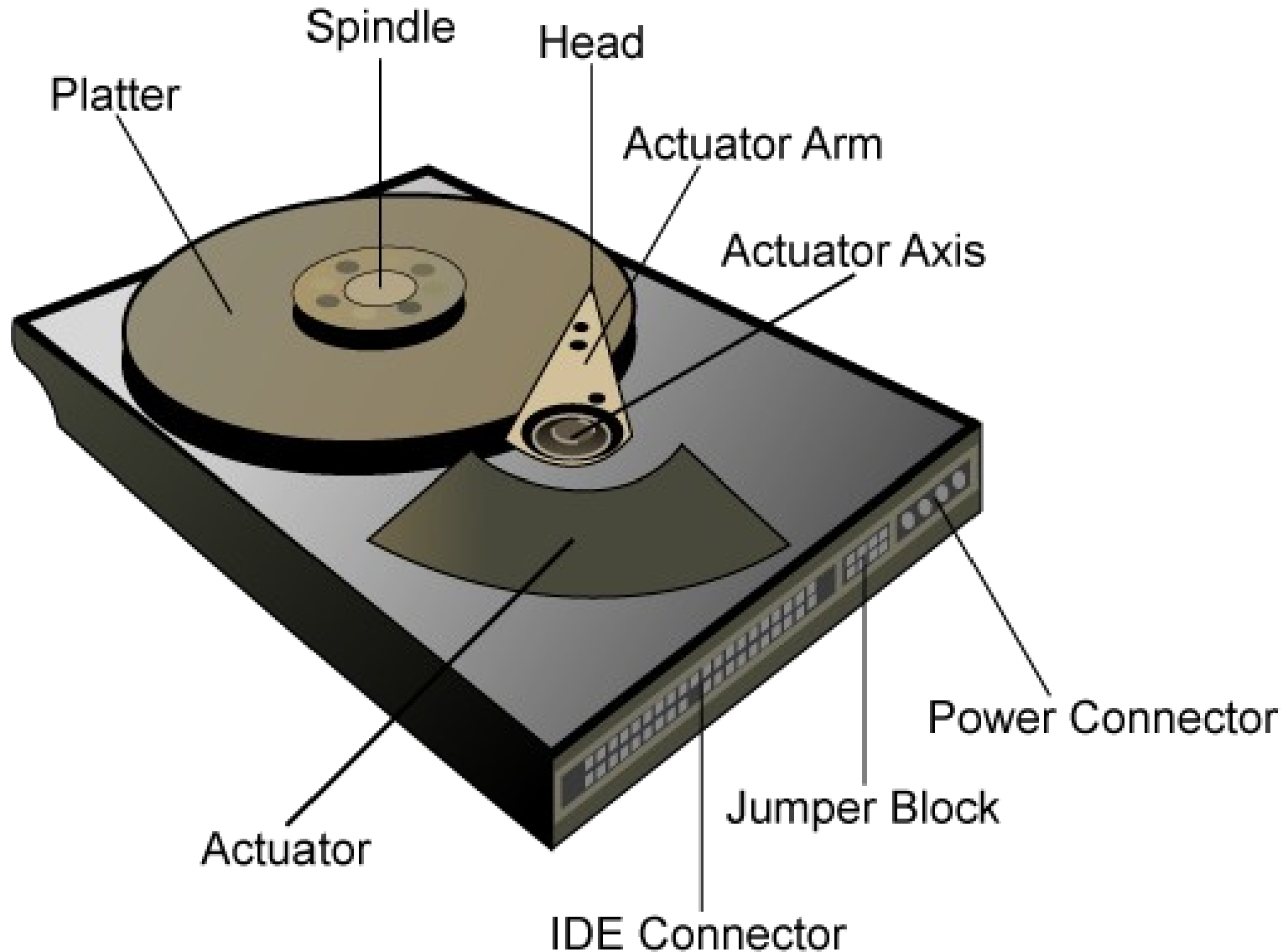
CS 5600

Computer Systems

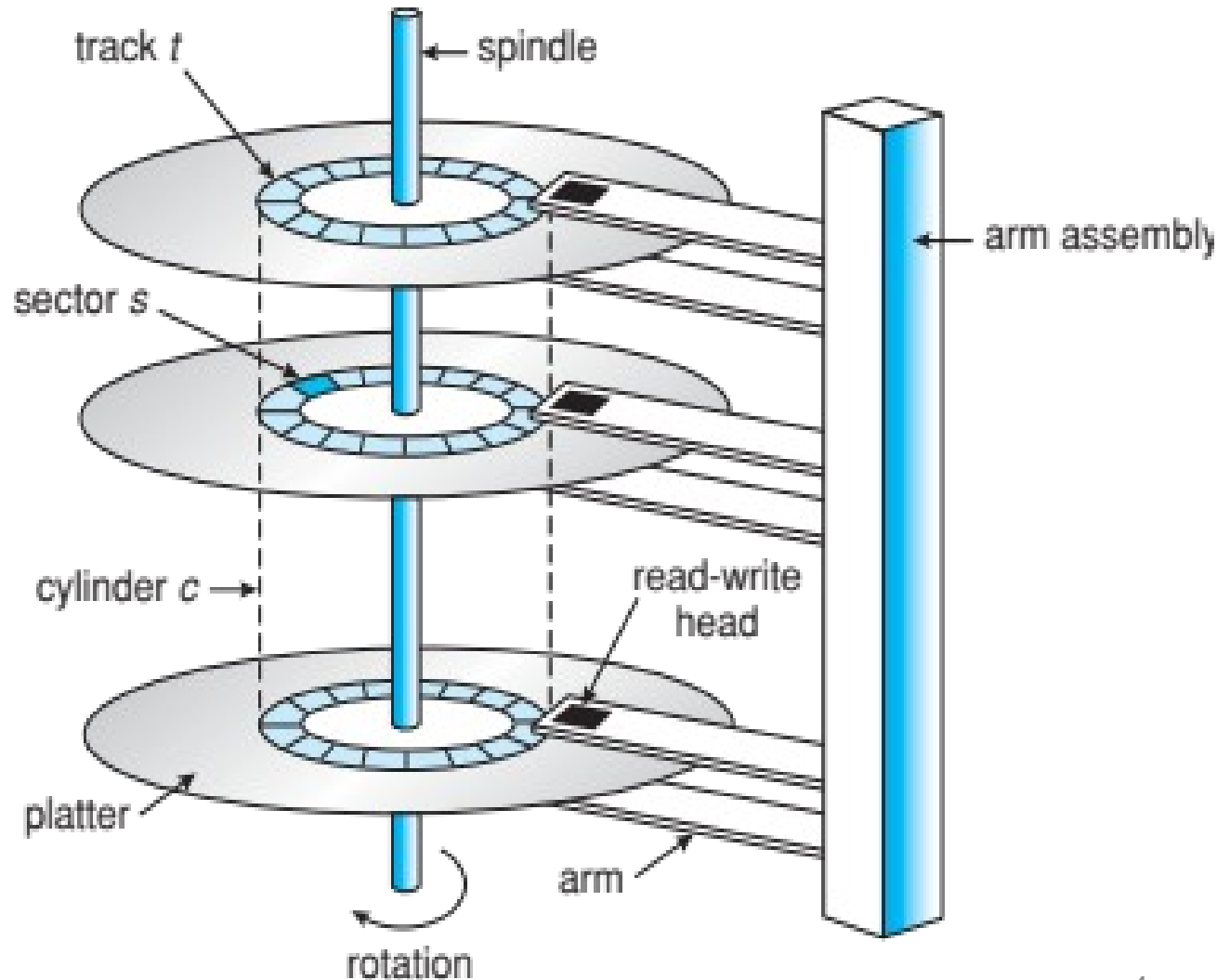
Lecture 8: Storage Devices

- Hard Drives
- RAID
- SSD

Hard Drive Hardware



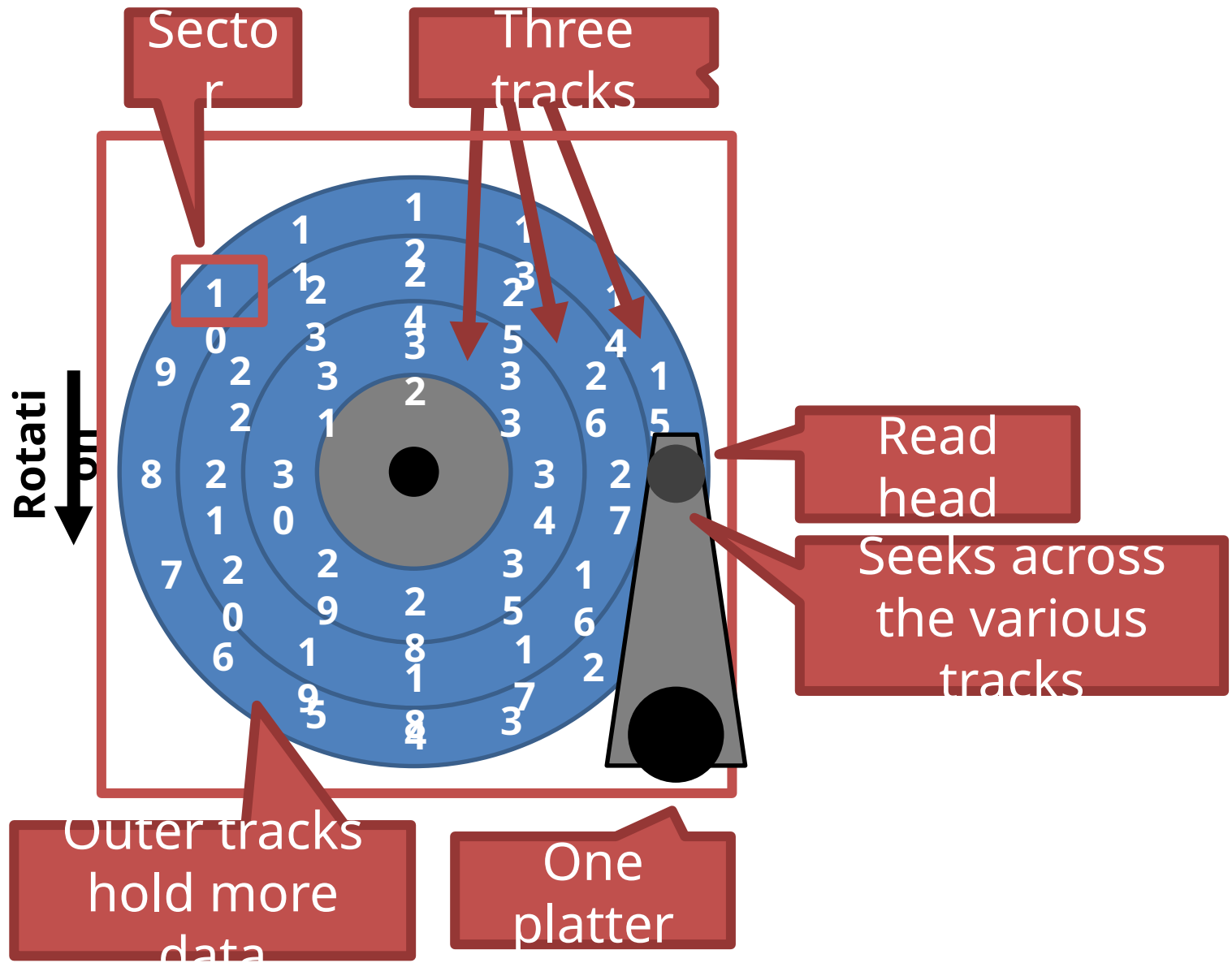
A Multi-Platter Disk



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)

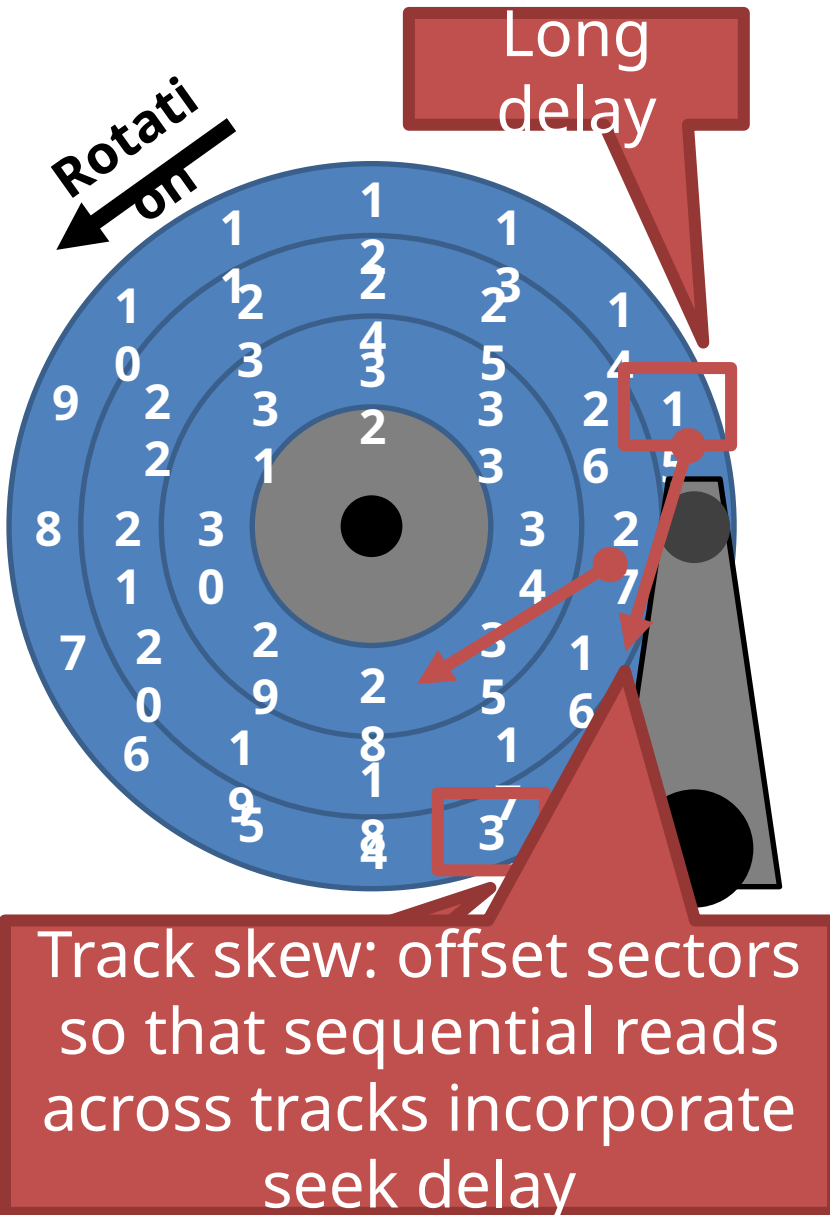
Geometry Example



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

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 Rate	125 MB/s	105 MB/s

Transfer time

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

Assume we are transferring 4096 bytes

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

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

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

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

Sequential vs. Random Access

Rate of I/O

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

Access Type	Transfer Size		Cheetah 15K.5	Barracuda
		$T_{I/O}$	6 ms	13.2 ms

		$R_{I/O}$	125 MB/s	95 MB/s
Max Transf			125 MB/s	95 MB/s

Random I/O results in very poor disk performance!

Caching

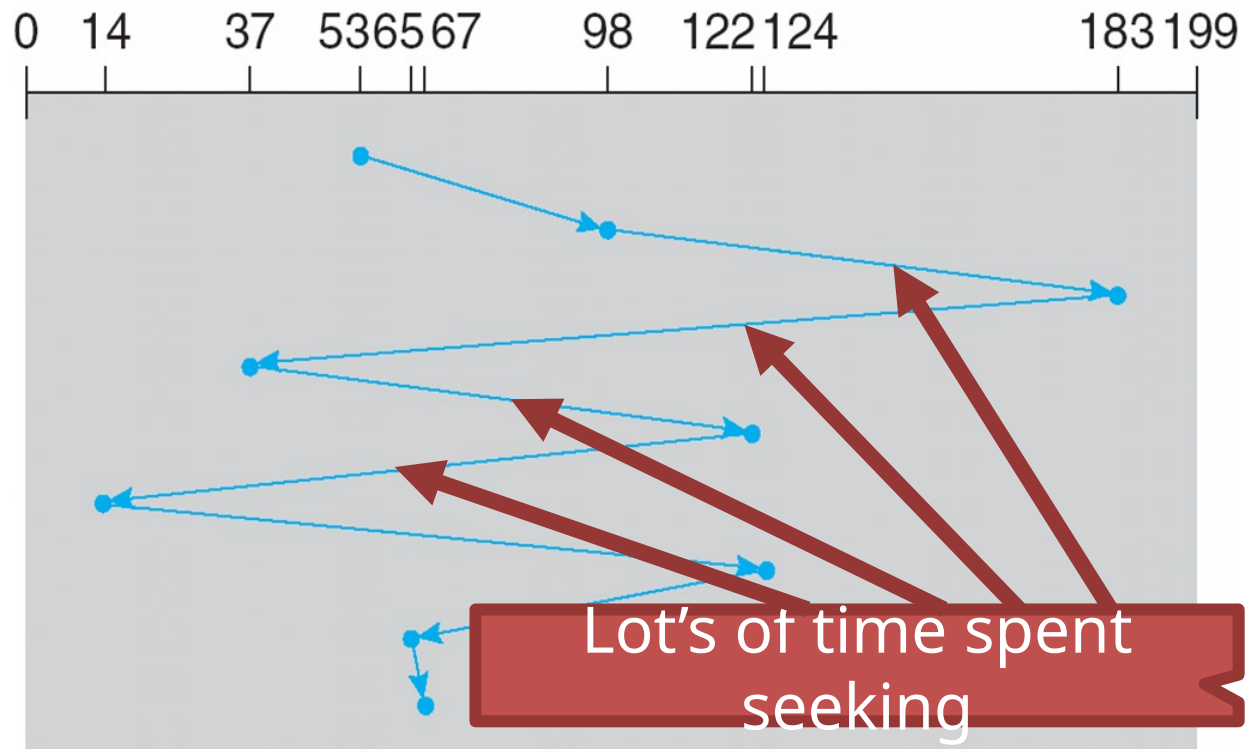
- Many disks incorporate caches (**track buffer**)
 - Small amount of RAM (8, 16, or 32 MB)
- Read caching
 - Reduces read delays due to seeking and rotation
- Write caching
 - **Write back cache**: drive reports that writes are complete after they have been cached
 - Possibly dangerous feature. Why?
 - **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 (FCFC)
 - 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

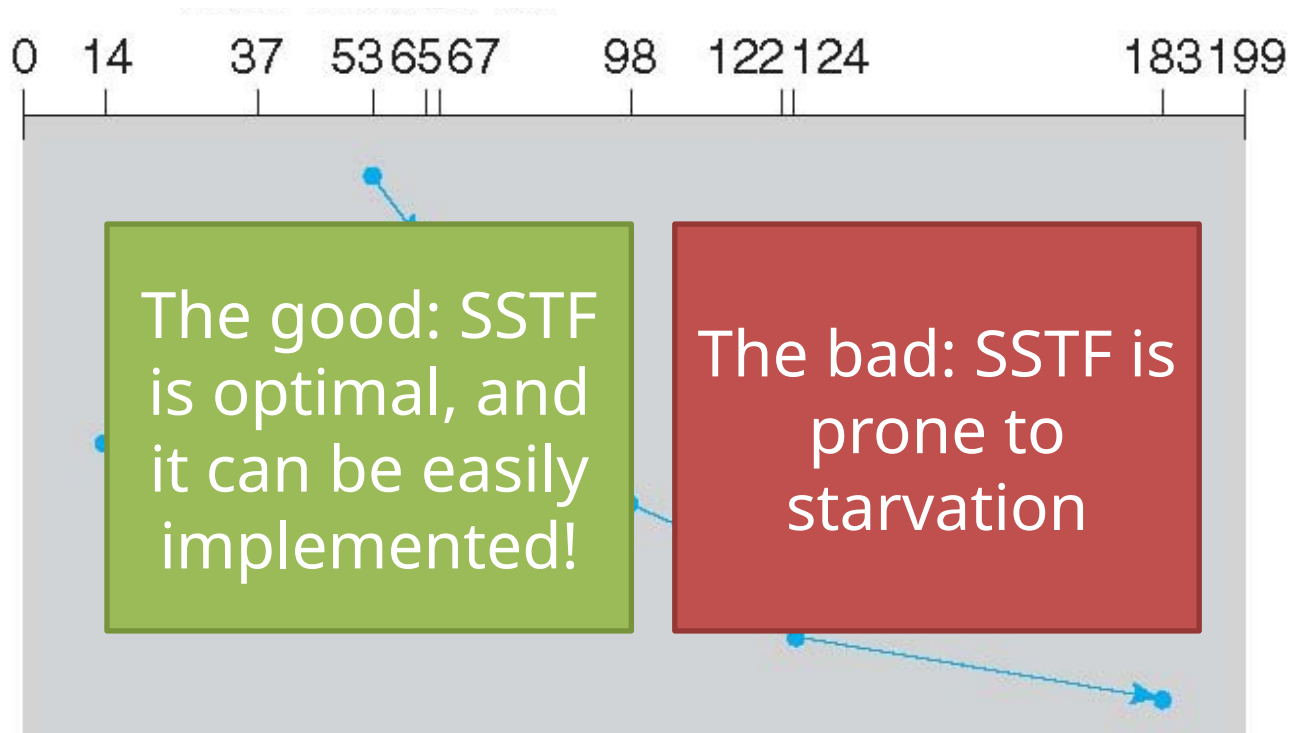


SSTF Scheduling

- 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

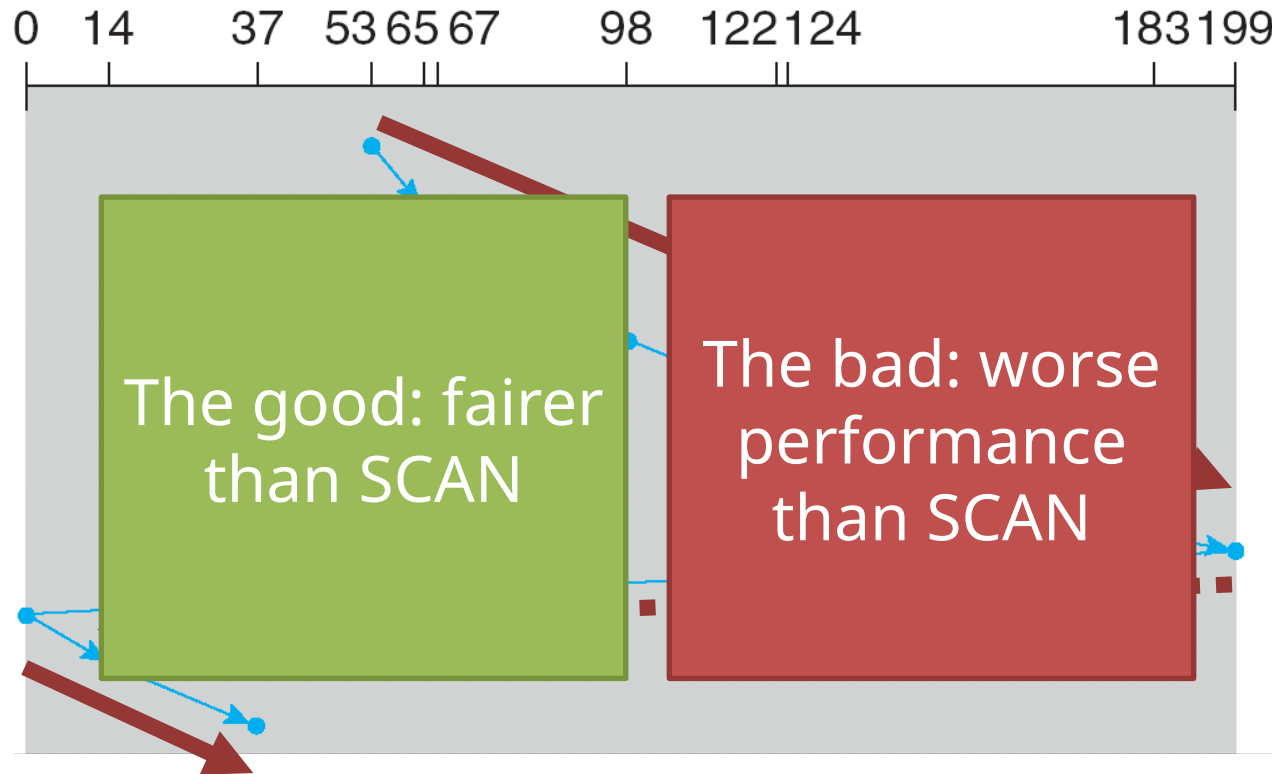


- Total movement: 236 cylinders

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

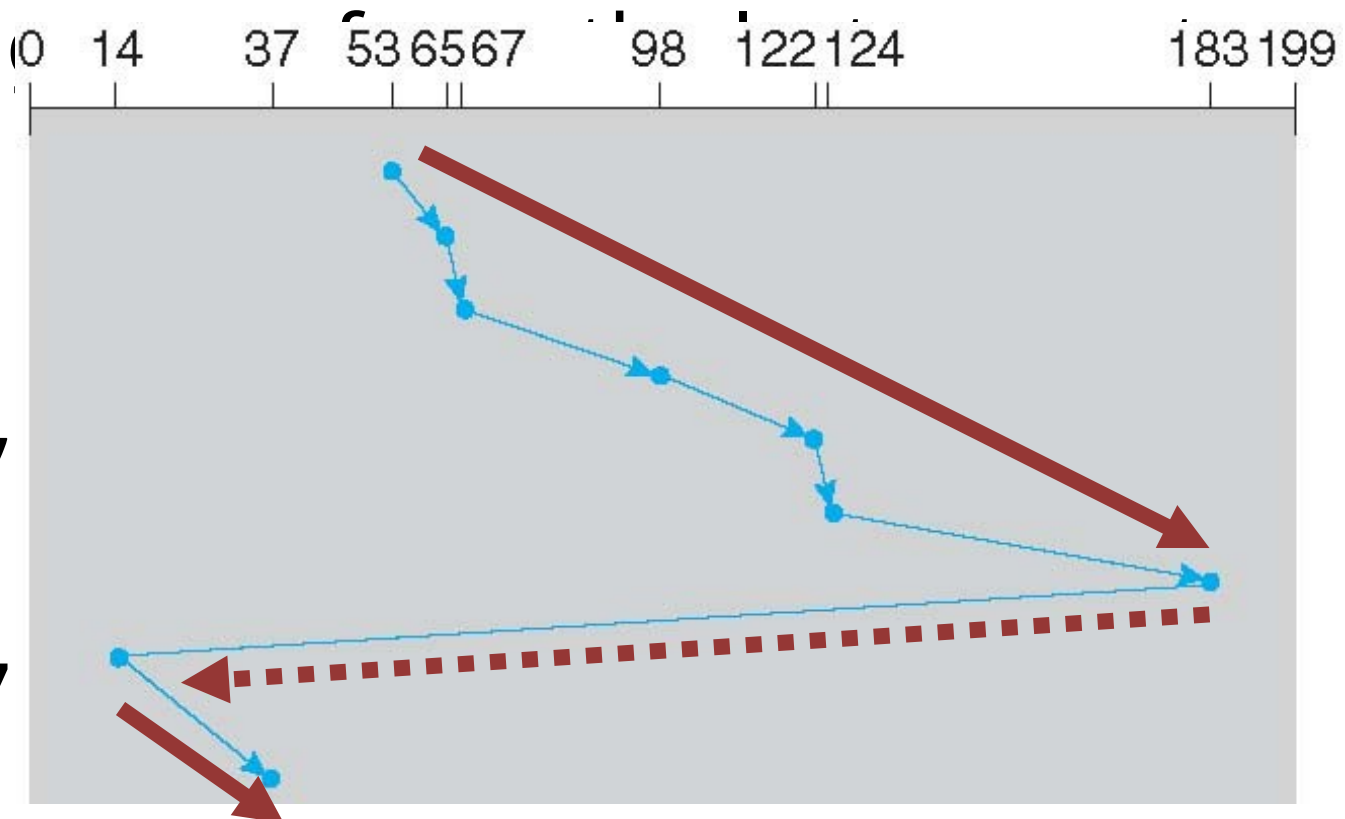
C-LOOK Example

- Peek at the upcoming addresses in the queue

- Head only

- Head starts at block 53

- Queue: 98, 183, 37, 122, 14, 124, 65, 67



- Total movement: 322 cylinders

Implementing Disk Scheduling

- We have talked about several 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

- Hard Drives
- RAID
- SSD

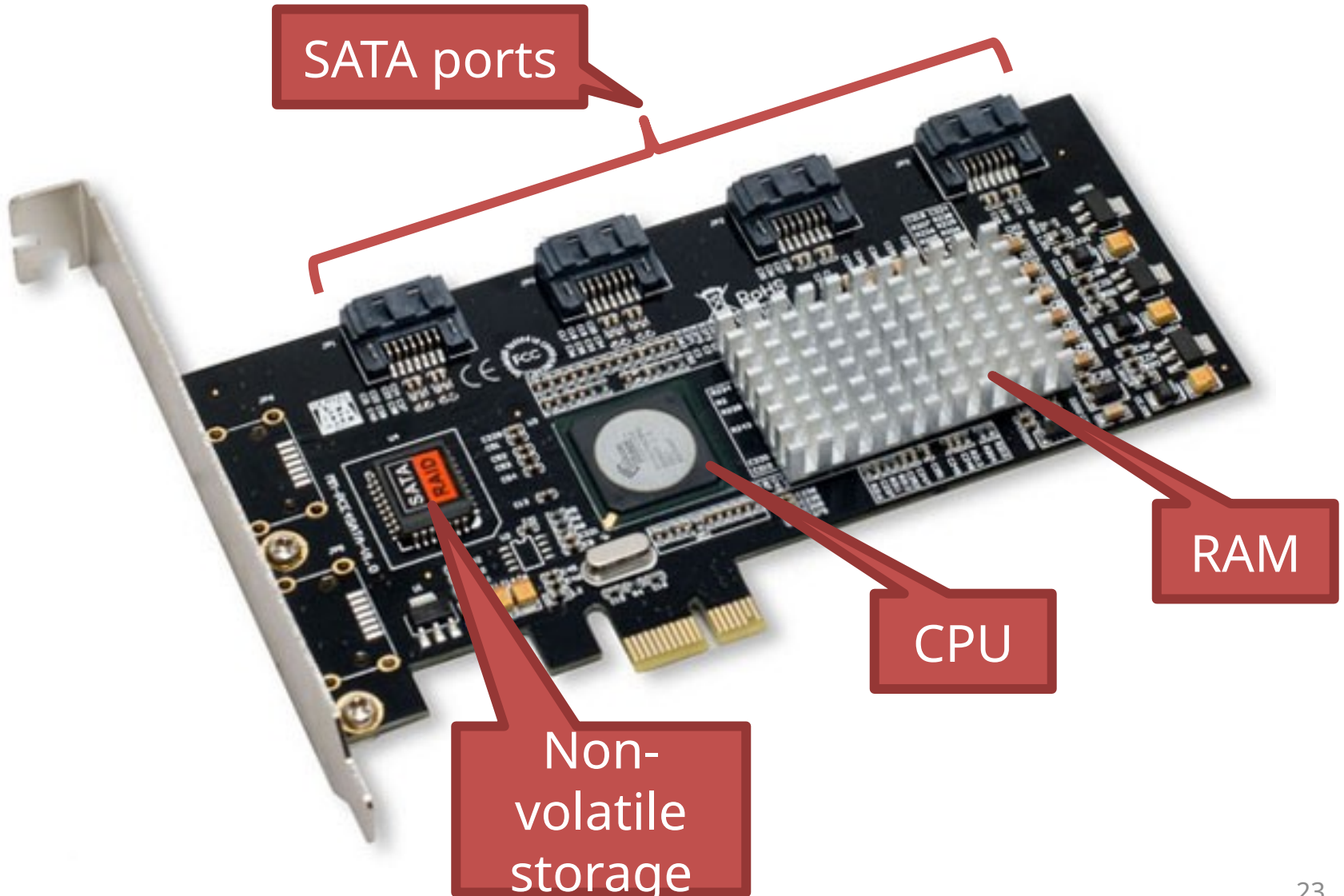
Beyond Single Disks

- Hard drives are great devices
 - Relatively fast, persistent storage
- Shortcomings:
 - How to cope with disk failure?
 - Mechanical parts break over time
 - Sectors may become silently corrupted
 - Capacity is limited
 - Managing files across multiple physical devices is cumbersome
 - Can we make 10x 1 TB drives look like a 10 TB drive?

Redundant Array of Inexpensive Disks

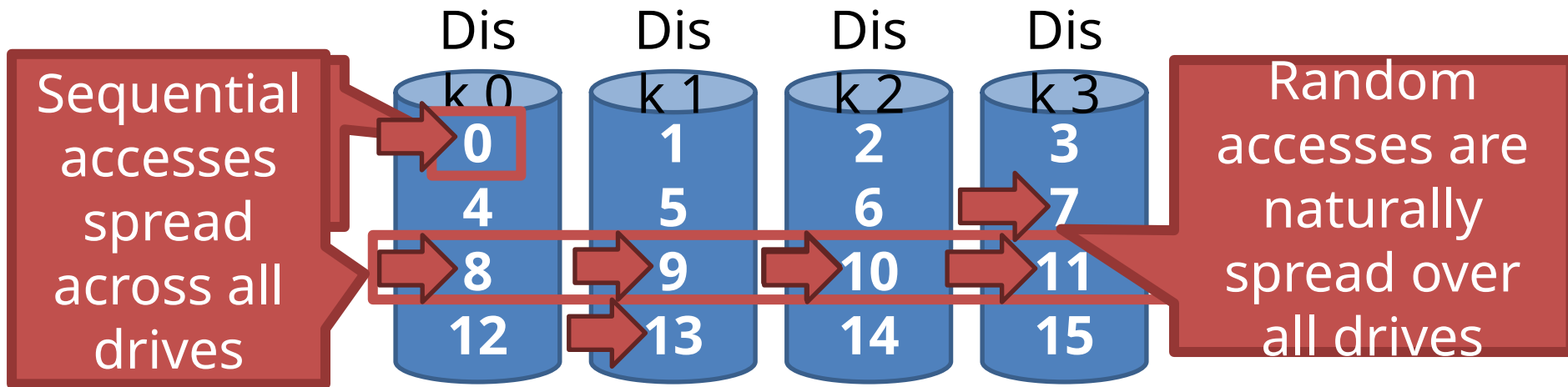
- RAID: use multiple disks to create the illusion of a large, faster, more reliable disk
- Externally, RAID looks like a single disk
 - i.e. RAID is transparent
 - Data blocks are read/written as usual
 - No need for software to explicitly manage multiple disks or perform error checking/recovery
- Internally, RAID is a complex computer system
 - Disks managed by a dedicated CPU + software
 - RAM and non-volatile memory
 - Many different configuration options (RAID levels)

Example RAID Controller



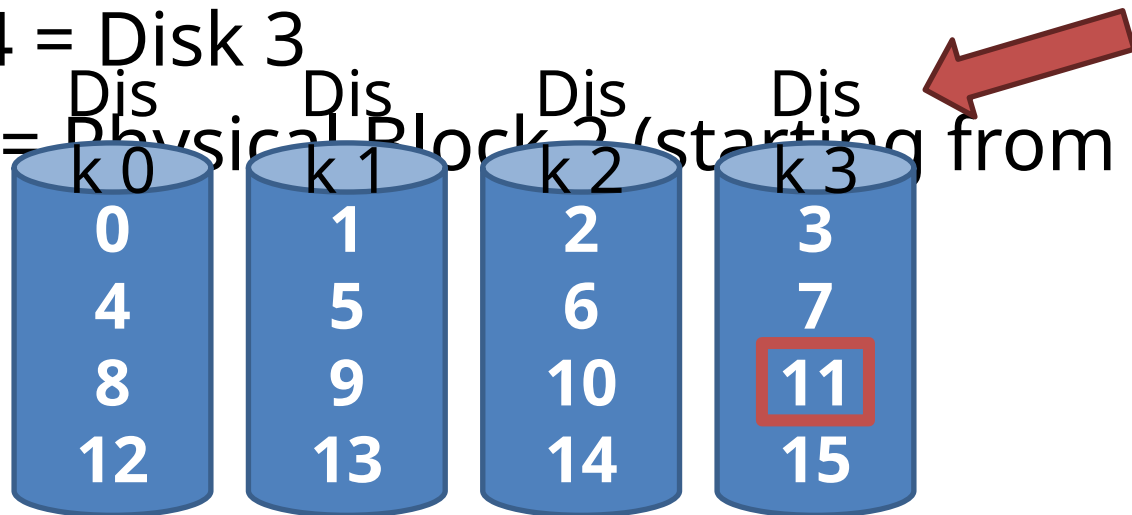
RAID 0: Striping

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

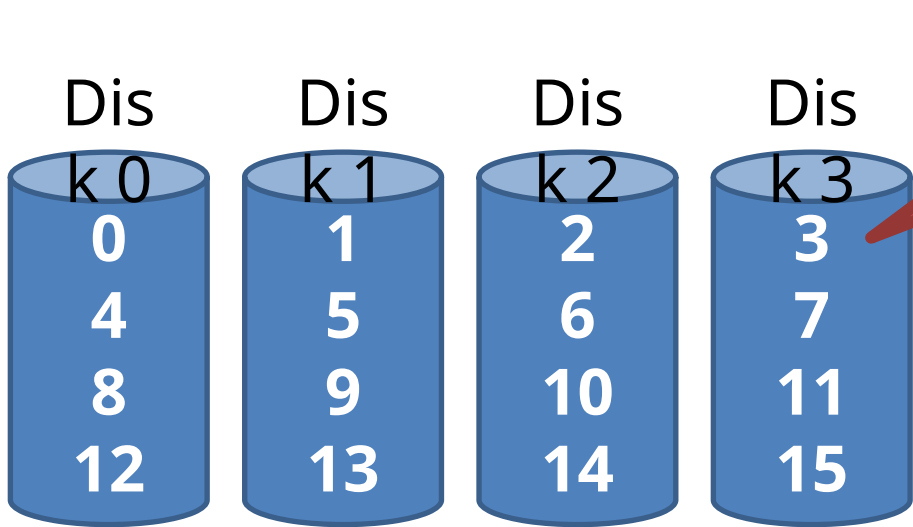


Addressing Blocks

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

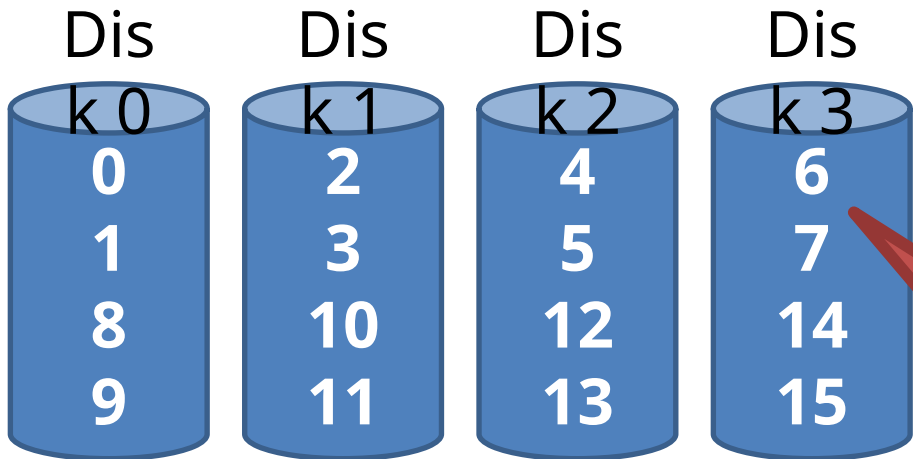


Chunk Sizing



Chunk size = 1
block

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



Chunk size = 2
block

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
 - $S = \text{transfer_size} / \text{time_to_access}$
 - $10 \text{ MB} / (7 \text{ ms} + 3 \text{ ms} + 10 \text{ MB} / 50 \text{ MB/s}) = 47.62 \text{ MB/s}$



Average seek time	7 ms
Average rotational delay	3 ms
Transfer rate	50

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
 - $R = \text{transfer_size} / \text{time_to_access}$
 - $10 \text{ KB} / (7 \text{ ms} + 3 \text{ ms} + 10 \text{ KB} / 50 \text{ MB/s}) = 0.98 \text{ MB/s}$



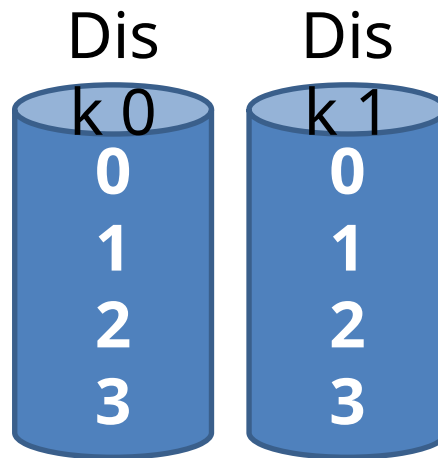
Average seek time	7 ms
Average rotational delay	3 ms
Transfer rate	50

Analysis of RAID 0

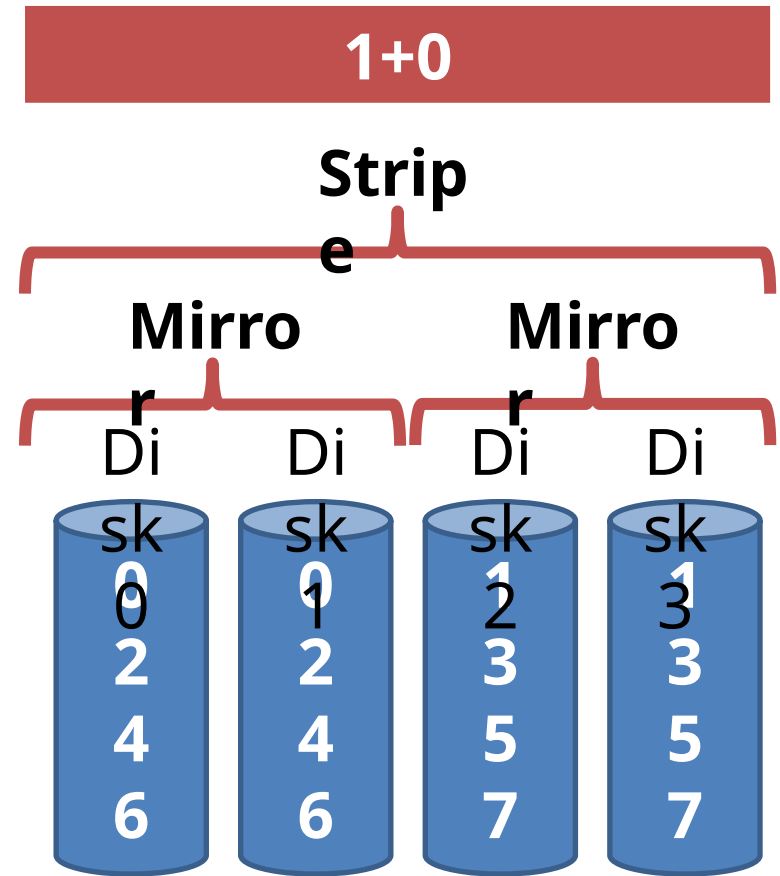
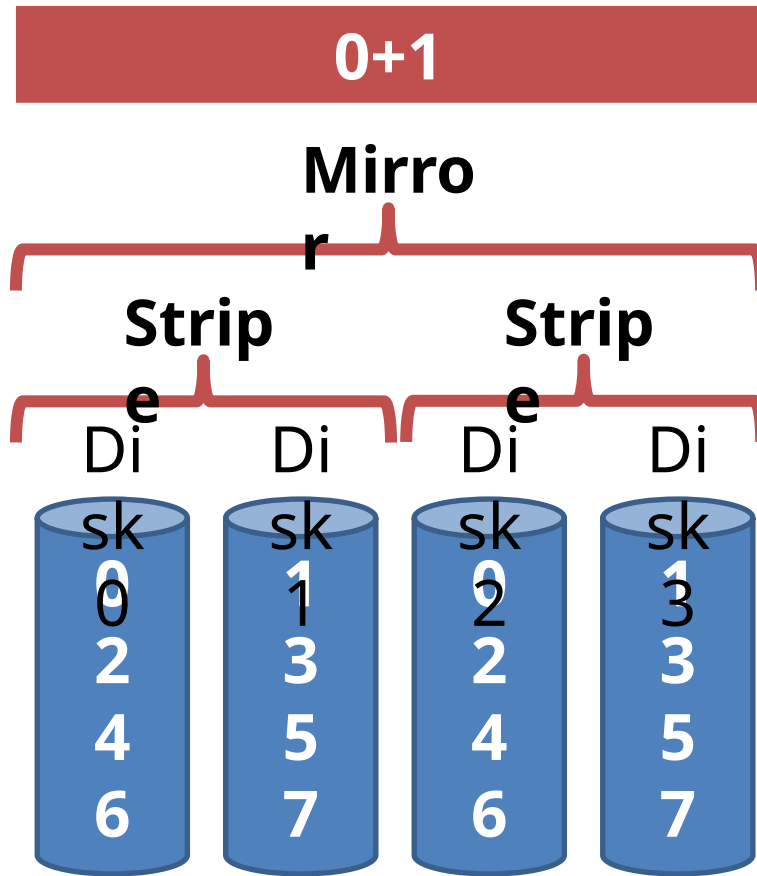
- Capacity: N
 - All space on all drives can be filled with data
- Reliability: 0
 - If any drive fails, data is permanently lost
- Sequential read and write: $N * S$
 - Full parallelization across drives
- Random read and write: $N * R$
 - Full parallelization across all drives

RAID 1: Mirroring

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



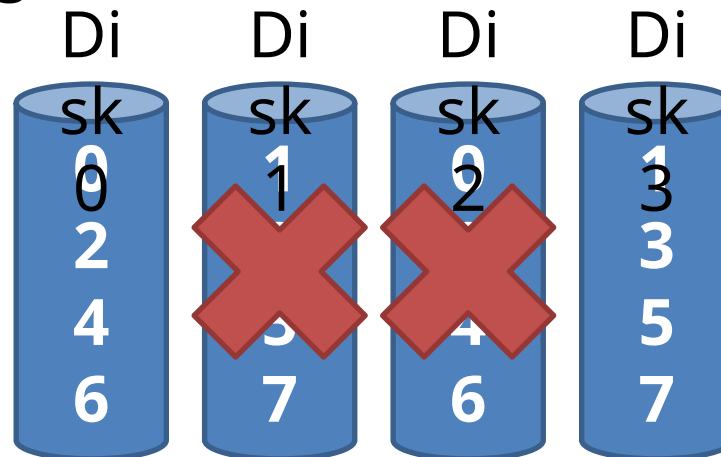
RAID 0+1 and 1+0 Examples



- Combines striping and mirroring
- Superseded by RAID 4, 5, and 6

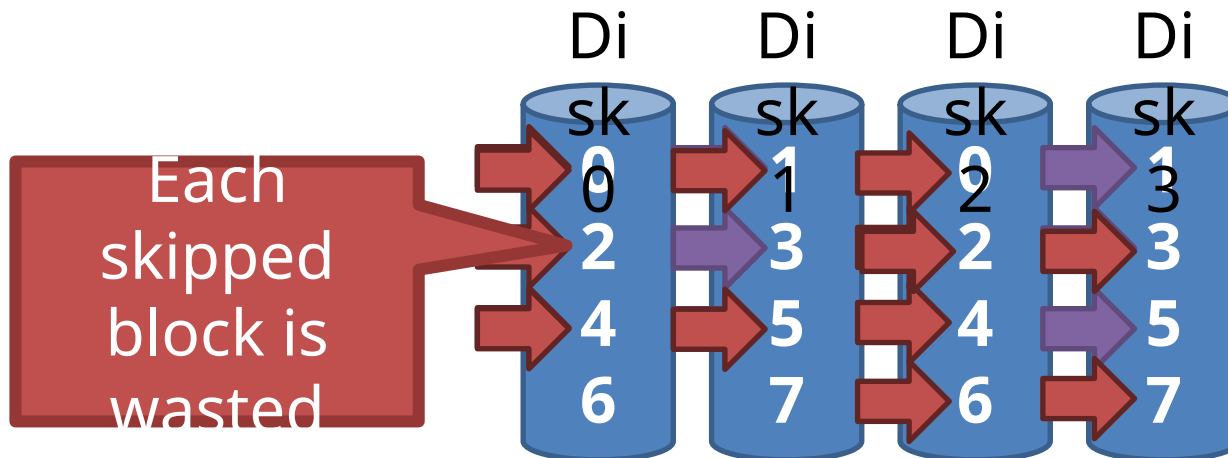
Analysis of RAID 1 (1)

- Capacity: $N / 2$
 - Two copies of all data, thus half capacity
- Reliability: 1 drive can fail, sometime more
 - If you are lucky, $N / 2$ drives can fail without data loss



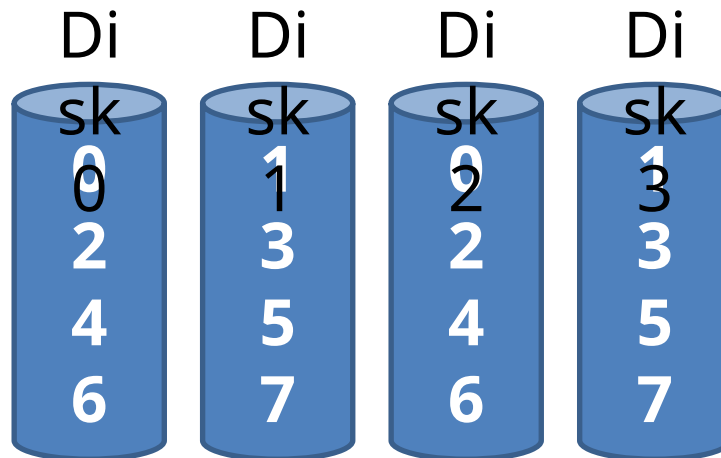
Analysis of RAID 1 (2)

- Sequential write: $(N / 2) * S$
 - Two copies of all data, thus half throughput
- Sequential read: $(N / 2) * S$
 - Half of the read blocks are wasted, thus halving throughput



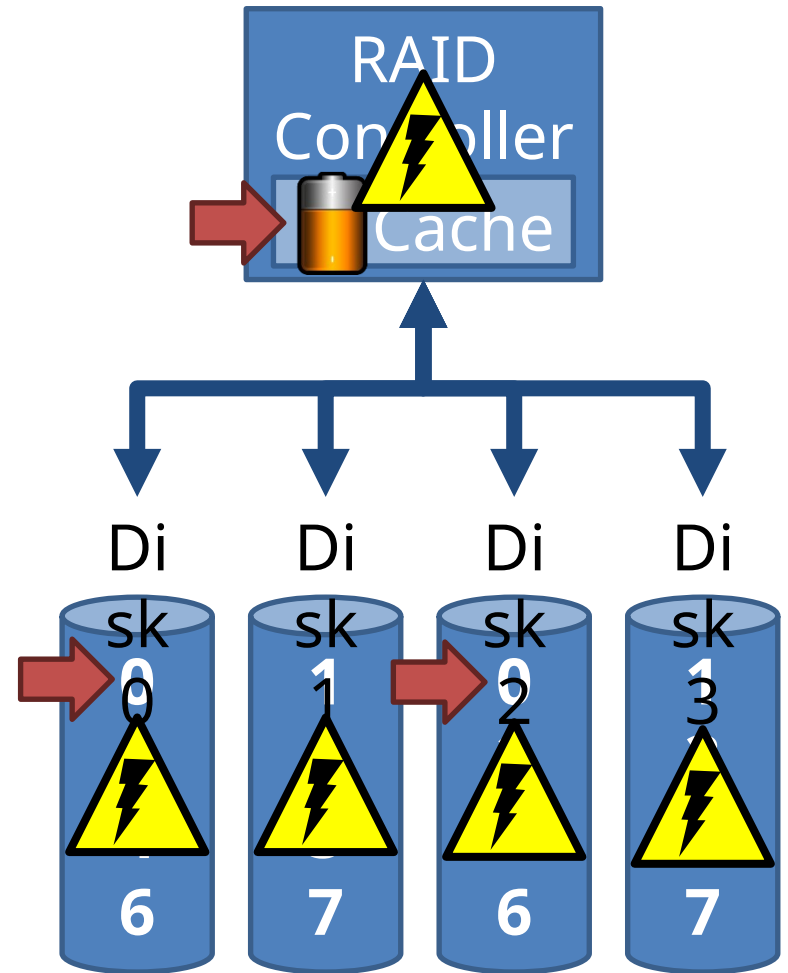
Analysis of RAID 1 (3)

- Random read: $N * R$
 - Best case scenario for RAID 1
 - Reads can parallelize across all disks
- Random write: $(N / 2) * R$
 - Two copies of all data, thus half throughput



The Consistent Update Problem

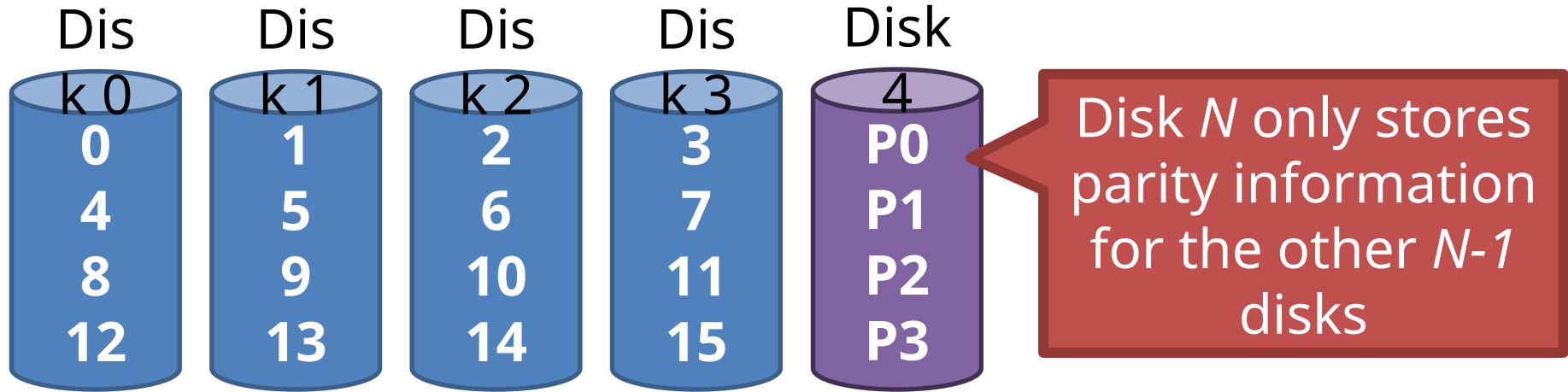
- Mirrored writes should be **atomic**
 - All copies are written, or none are written
- However, this is difficult to guarantee
 - Example: power failure
- Many RAID controllers include a **write-ahead log**
 - Battery backed, non-volatile storage of pending writes



Decreasing the Cost of Reliability

- RAID 1 offers highly reliable data storage
- But, it uses $N / 2$ of the array capacity
- Can we achieve the same level of reliability without wasting so much capacity?
 - Yes!
 - Use information coding techniques to build light-weight error recovery mechanisms

RAID 4: Parity Drive

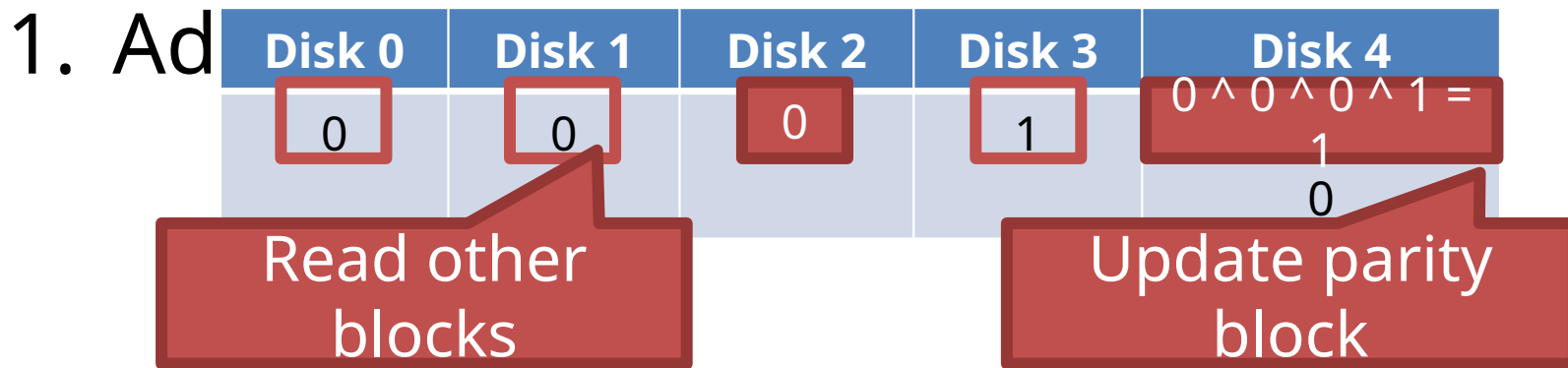


Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	0	1	1	$0 \wedge 0 \wedge 1 \wedge 1 = 0$
0	1	0	0	$0 \wedge 1 \wedge 0 \wedge 0 = 1$
1	1	1	1	$1 \wedge 1 \wedge 1 \wedge 1 = 0$
0	1	1	1	$0 \wedge 1 \wedge 1 \wedge 1 =$

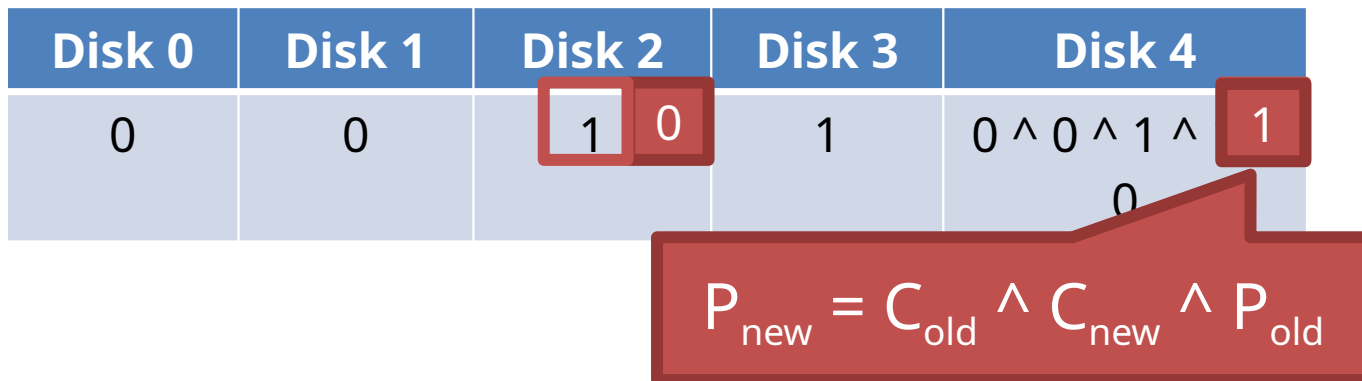
Callout: Parity calculated using XOR

Updating Parity on Write

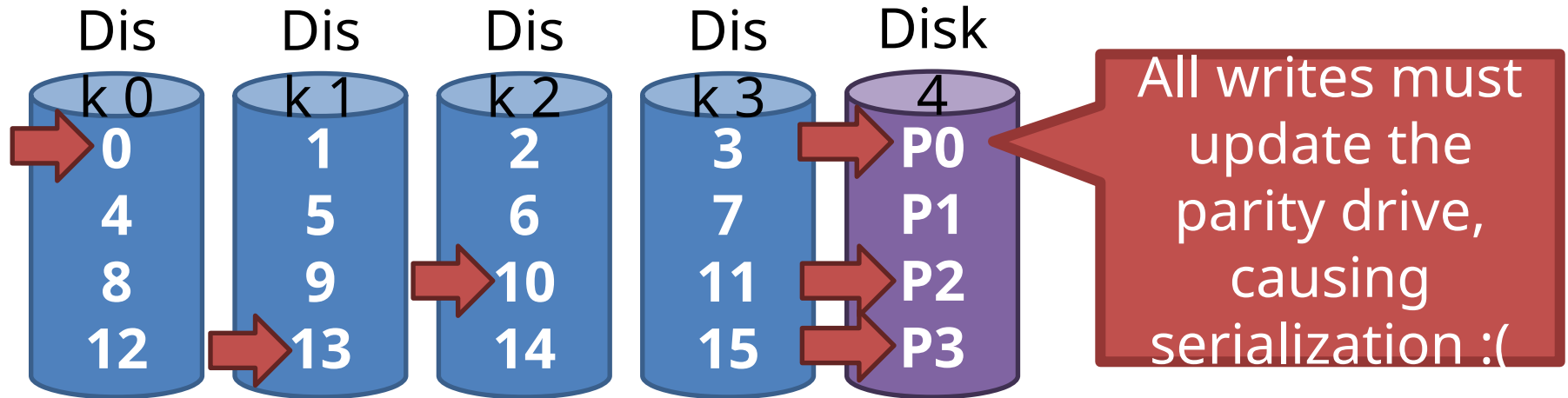
- How is parity updated when blocks are written?



2. Subtractive parity



Random Writes and RAID 4

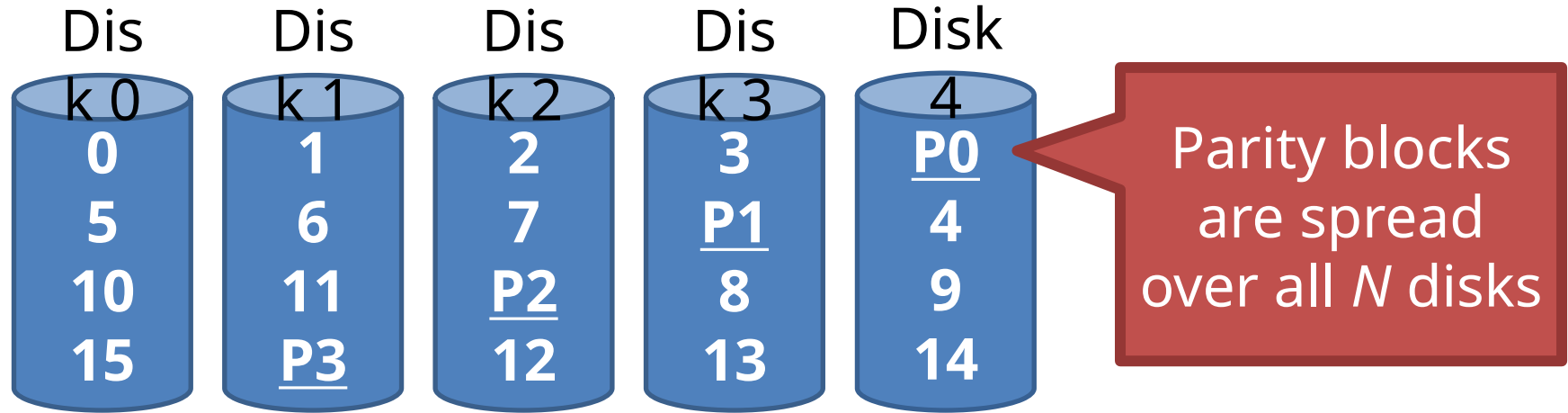


- Random writes in RAID 4
 1. Read the target block and the parity block
 2. Use subtraction to calculate the new parity block
 3. Write the target block and the parity block
- RAID 4 has terrible write performance
 - Bottlenecked by the parity drive

Analysis of RAID 4

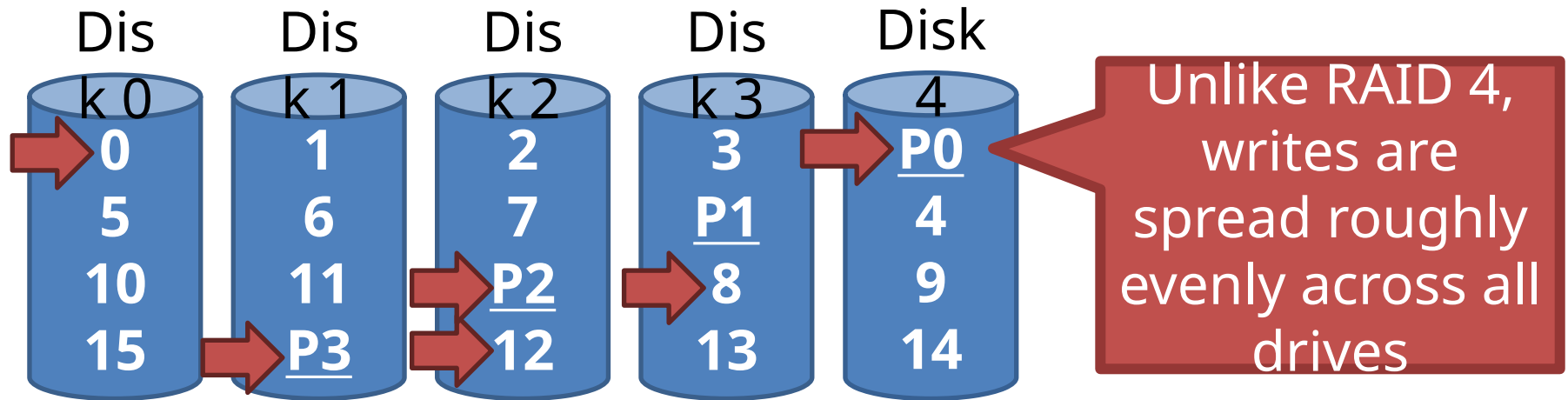
- Capacity: $N - 1$
 - Space on the parity drive is lost
- Reliability: 1 drive can fail
- Sequential Read and write: $(N - 1) * S$
 - Parallelization across all non-parity blocks
- Random Read: $(N - 1) * R$
 - Reads parallelize over all but the parity drive
- Random Write: $R / 2$
 - Writes serialize due to the parity drive
 - Each write requires 1 read and 1 write of the parity drive, thus $R / 2$

RAID 5: Rotating Parity



Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	0	1	1	$0 \wedge 0 \wedge 1 \wedge 1 = 0$
1	0	0	$0 \wedge 1 \wedge 0 \wedge 0 = 1$	0
1	1	$1 \wedge 1 \wedge 1 \wedge 1 = 0$	1	1

Random Writes and RAID 5



- Random writes in RAID 5
 1. Read the target block and the parity block
 2. Use subtraction to calculate the new parity block
 3. Write the target block and the parity block
- Thus, 4 total operations (2 reads, 2 writes)
 - Distributed across all drives

Analysis of Raid 5

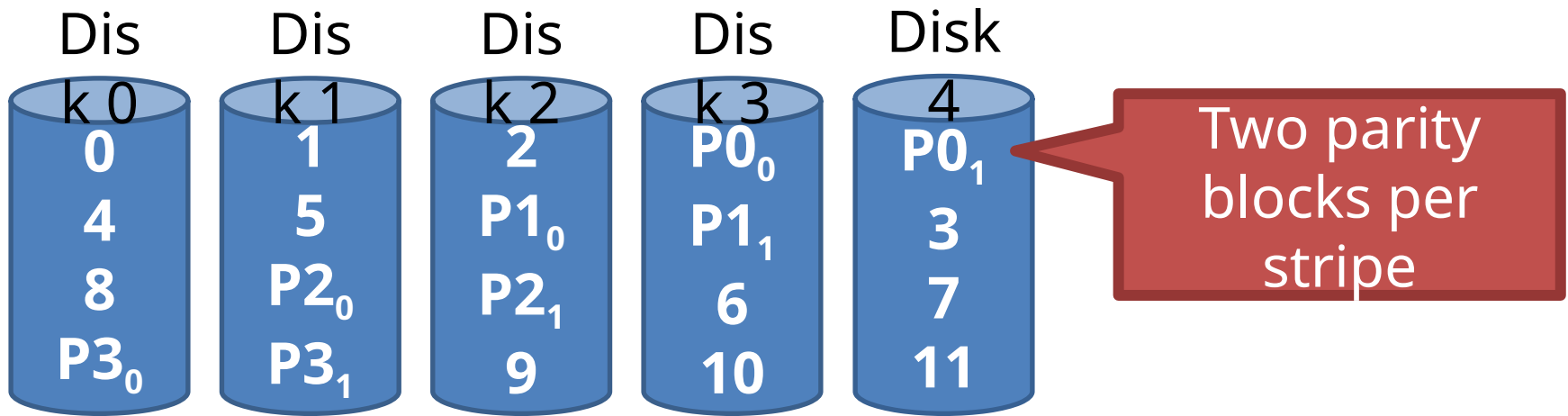
- Capacity: $N - 1$ [same as RAID 4]
- Reliability: 1 drive can fail [same as RAID 4]
- Sequential Read and write: $(N - 1) * S$ [same]
 - Parallelization across all non-parity blocks
- Random Read: $N * R$ [vs. $(N - 1) * R$]
 - Unlike RAID 4, reads parallelize over all drives
- Random Write: $N / 4 * R$ [vs. $R / 2$ for RAID 4]
 - Unlike RAID 4, writes parallelize over all drives
 - Each write requires 2 reads and 2 write, hence $N / 4$

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	$N * S$	$(N / 2) * S$	$(N - 1) * S$	$(N - 1) * S$
	Sequential Write	$N * S$	$(N / 2) * S$	$(N - 1) * S$	$(N - 1) * S$
	Random Read	$N * R$	$N * R$	$(N - 1) * R$	$N * R$

RAID 6



- Any two drives can fail
- $N - 2$ usable capacity
- No overhead on read, significant overhead on write
- Typically implemented using Reed-Solomon codes

Choosing a RAID Level

- Best performance and most capacity?
 - RAID 0
- Greatest error recovery?
 - RAID 1 (1+0 or 0+1) or RAID 6
- Balance between space, performance, and recoverability?
 - RAID 5

Other Considerations

- Many RAID systems include a **hot spare**
 - An idle, unused disk installed in the system
 - If a drive fails, the array is immediately rebuilt using the hot spare
- RAID can be implemented in hardware or software
 - Hardware is faster and more reliable...
 - But, migrating a hardware RAID array to a different hardware controller almost never works
 - Software arrays are simpler to migrate and cheaper, but have worse performance and weaker reliability
 - Due to the **consistent update** problem

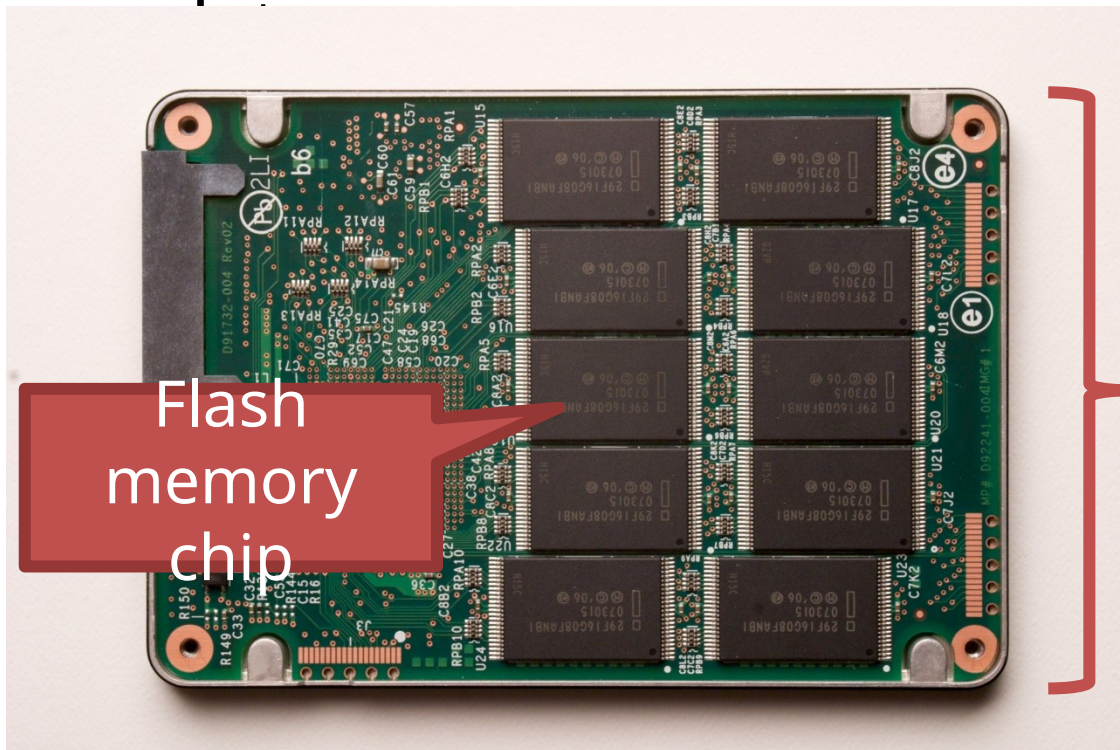
- Hard Drives
- RAID
- SSD

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

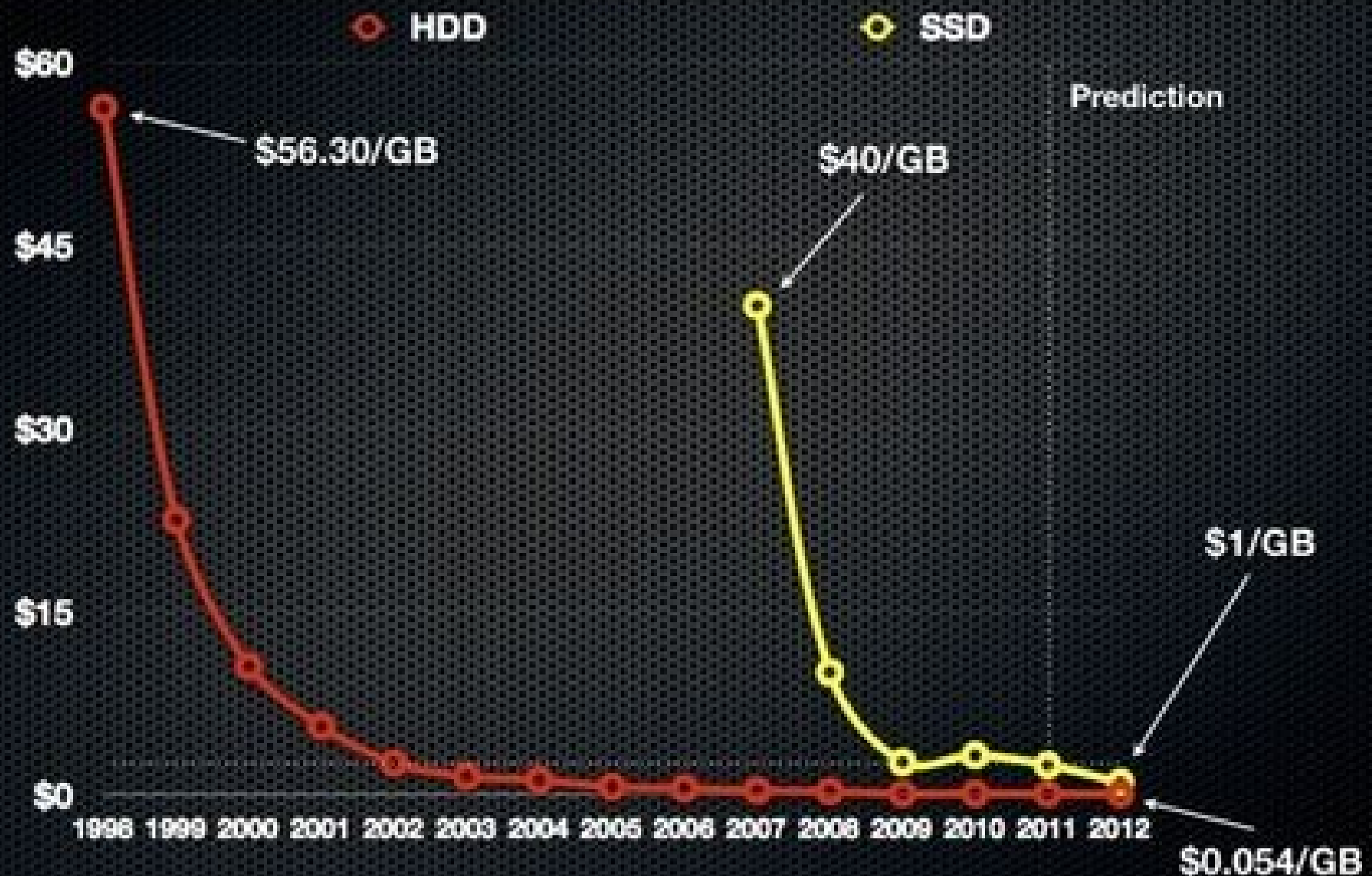


Data is striped across all chips

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



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 Amplification

G moved to new block by the garbage collector

Cleaned block can now be rewritten

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

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

- 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,

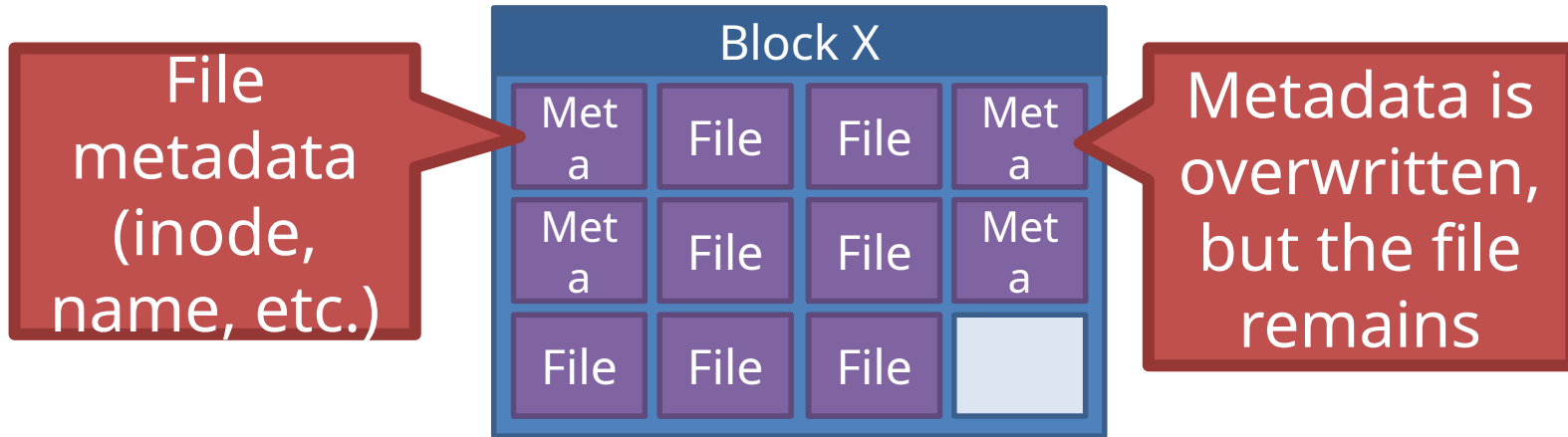
Garbage Collection

- 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

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

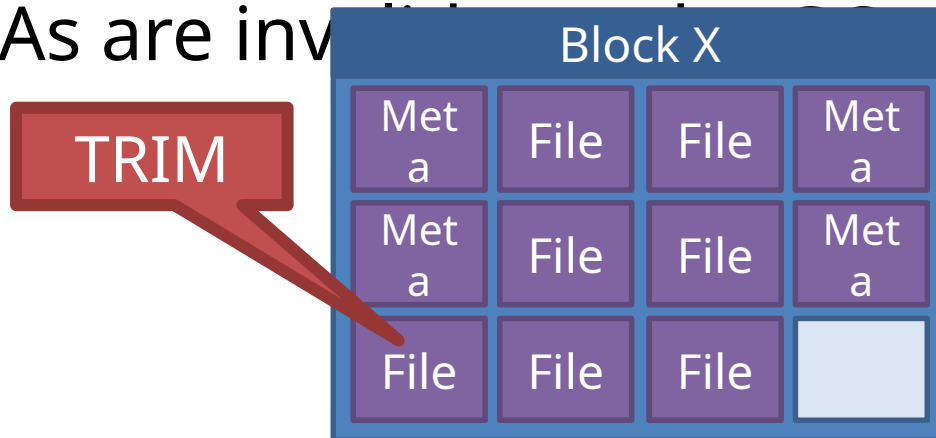


1. File is written to SSD
2. File is deleted
3. The GC executes
 - 9 pages look valid to the SSD
 - The OS knows only 2 pages are valid

- 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 and should be erased



- 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

We have two examples

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

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