File Structures

Ch03. A Disks, Raids and SSDs

2020. Spring

Instructor: Joonho Kwon

jhkwon@pusan.ac.kr

Data Science Lab @ PNU

datalab

data science laboratory

Slides are based on the textbook and Christo Wilson.

References

- **Operating Systems: Three Easy Pieces**
  - http://pages.cs.wisc.edu/~remzi/OSTEP/
    - **Ch37. Hard Disk Drives**
    - **Ch38. Redundant Disk Arrays (RAID)**
    - **Ch44. Flash-based SSDs**

- **Database Implementation**

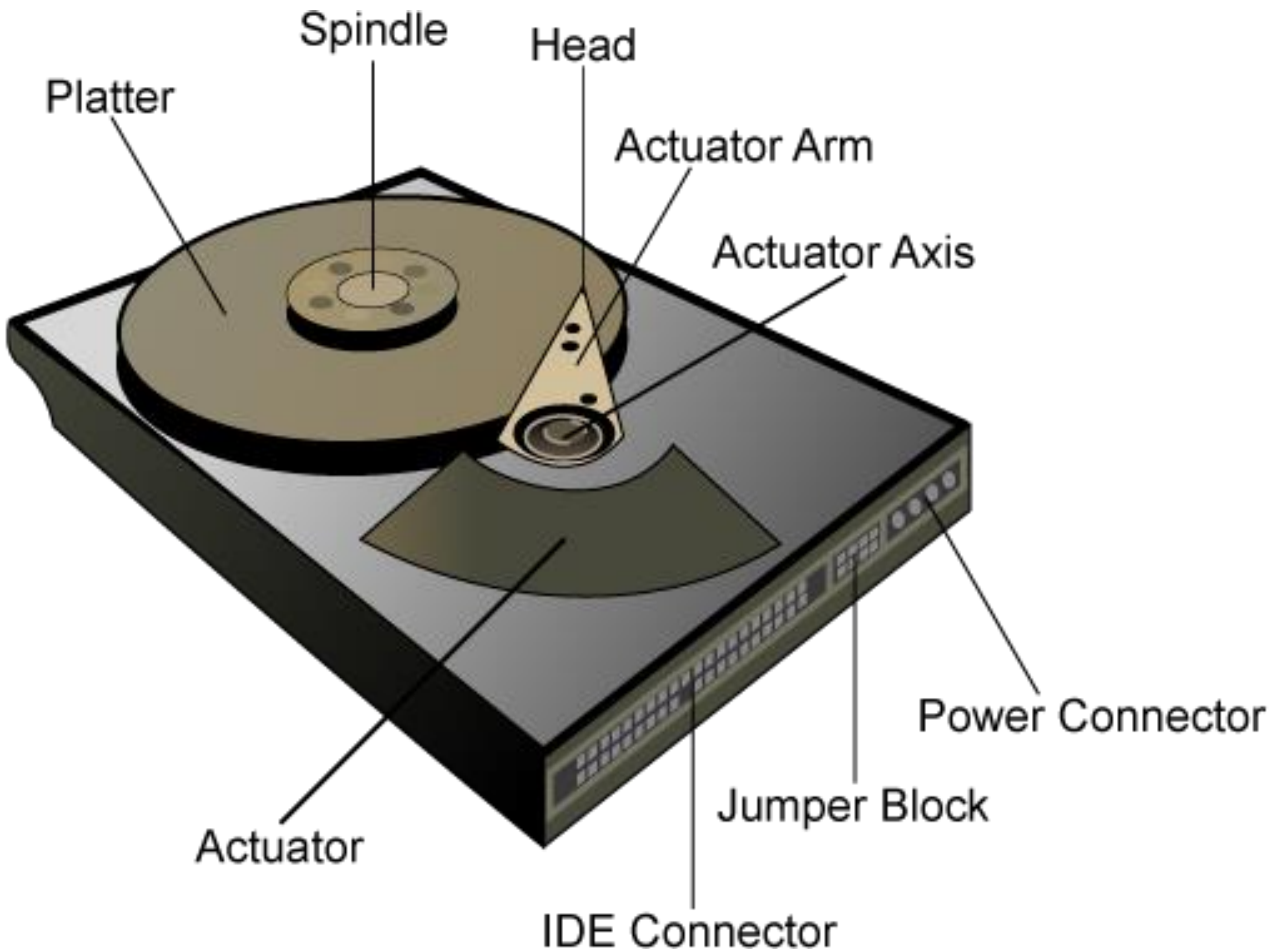
Why Study Disks?

- Good Design
  - Responsive to the constrains of the medium and to the environment
- If files were stored just in memory
  - Data structures are enough
  - No need for file structures
- Secondary Storage
  - Very different from memory
    - Take much more time
    - Not all accesses are equal

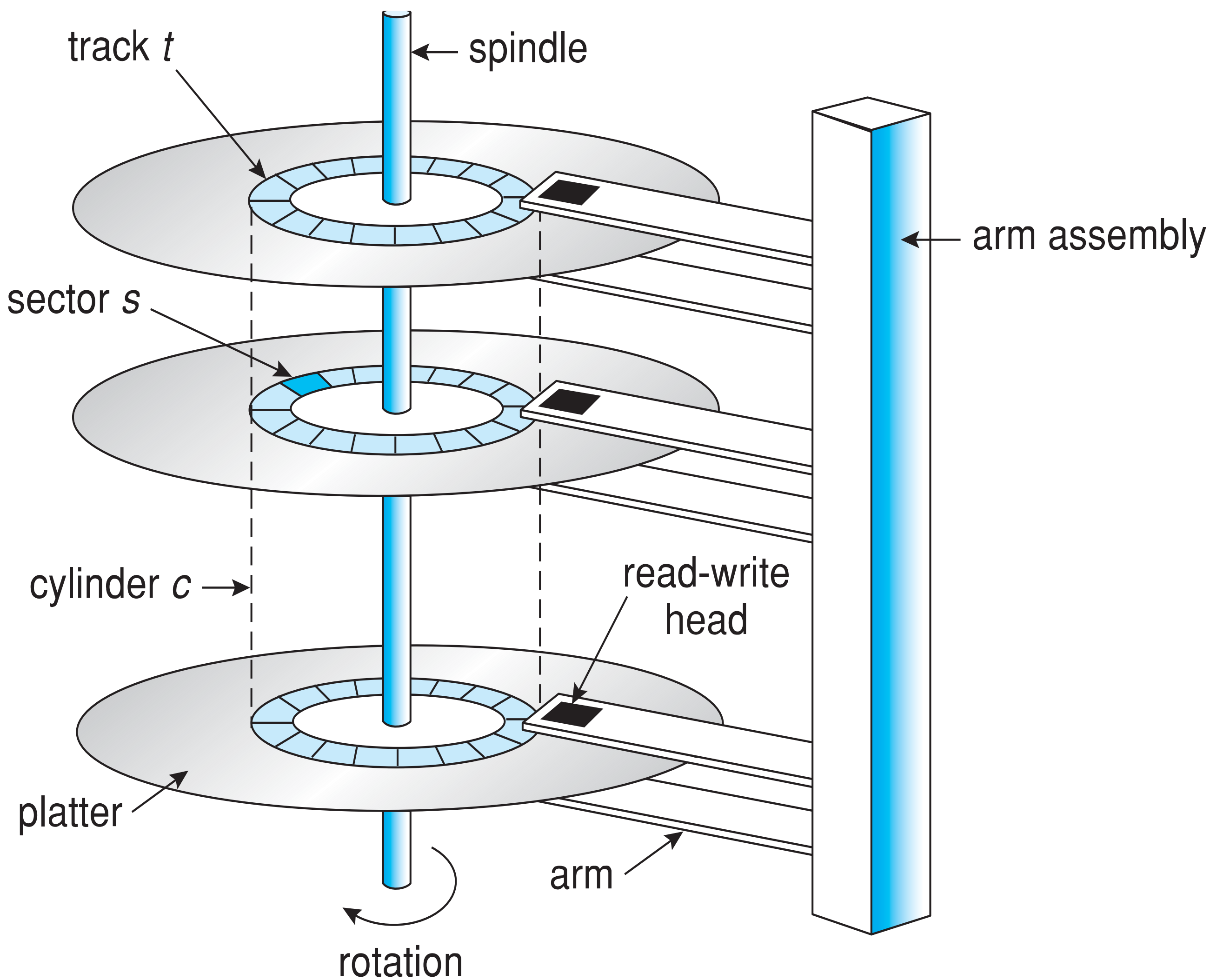
Contents

- Ch37. Hard Disk Drives
- Ch38. RAID
- Ch39. SSD

Hard Drive Hardware



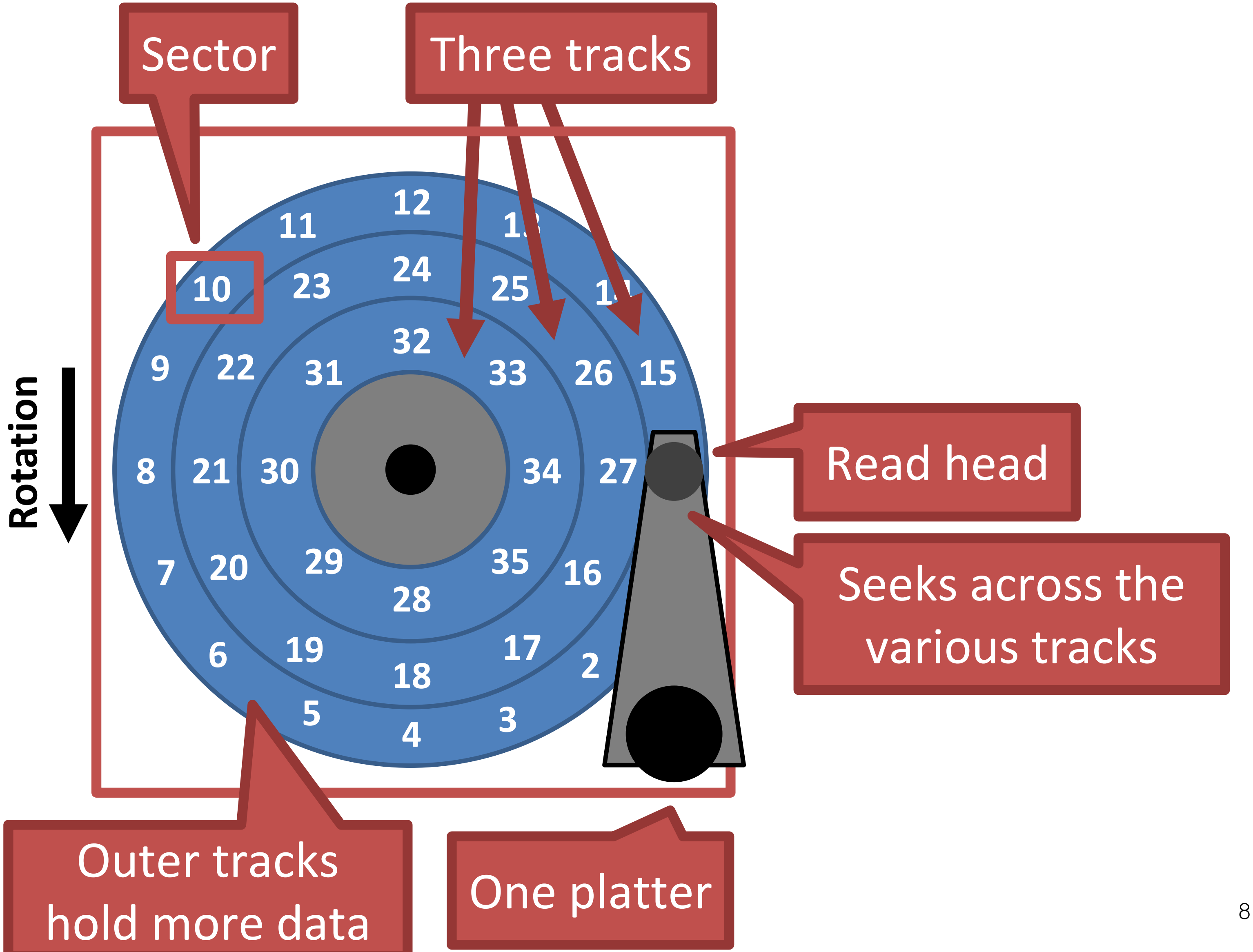
A Multi-Platter Disk



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)

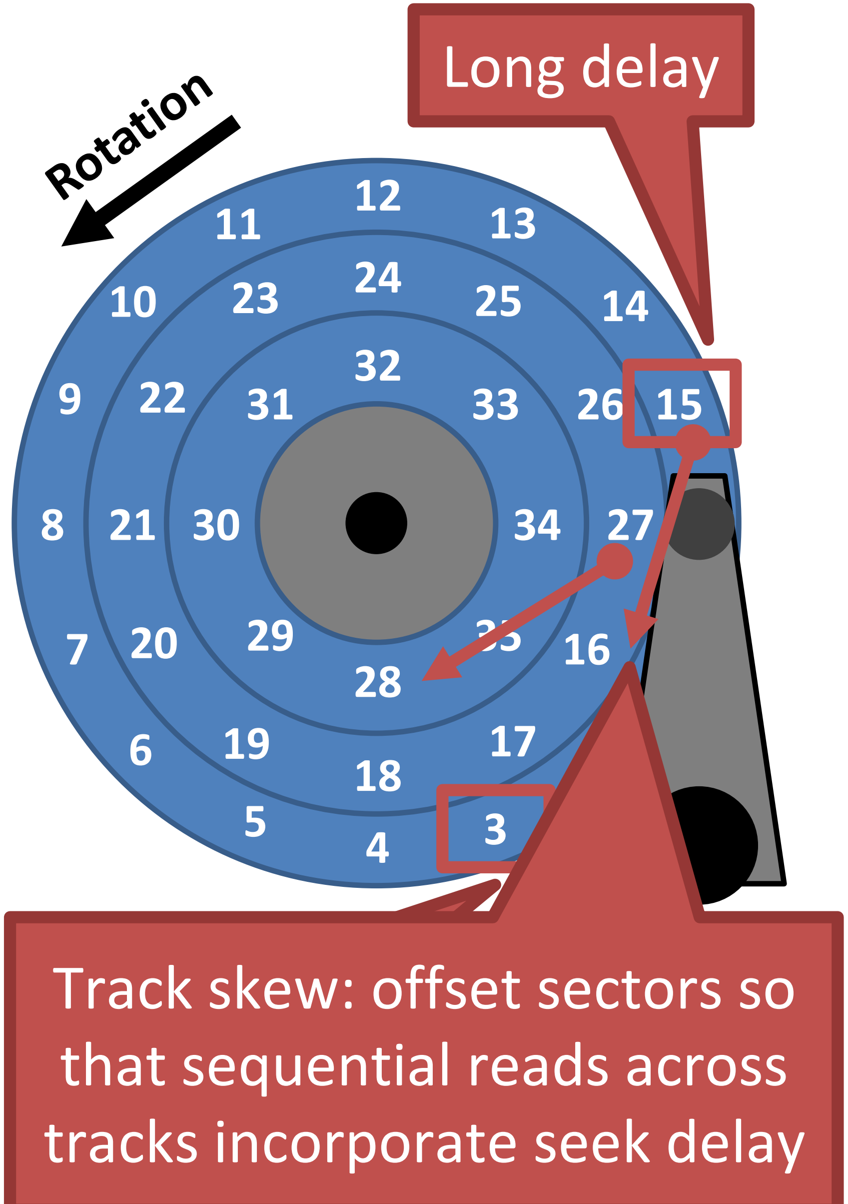
Geometry Example



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

Access Time

- Data blocks can only be read and written if disk heads and platters are positioned accordingly.
- This design has implications on the **access time** to read/write a given block:

Access Time

1. Move disk arms to desired track (seek time t_s)
2. Disk controller waits for desired block to rotate under disk head (rotational delay t_r)
3. Read/write data (transfer time t_{tr})

➔ Access Time: $t = t_s + t_r + t_{tr}$

Example: Seagate Cheetah 15K.7

Seagate

	Cheetah 15K.7
Capacity	600 GB
RPM	15000
Avg. Seek	3.4 ms
Max Transfer	163 MB/s

- Performance characteristics:
  - 4 disks, 8 heads, avg. 512 kB/track, 600 GB capacity
  - rotational speed: 15000 rpm (revolutions per minute)
  - average seek time: 3.4 ms
  - transfer rate 163 MB/s
- Q: What is the access time to read an 8KB block?
  - Average seek time: $t_s = 3.40\ ms$
  - Average rotational delay: $t_r = \frac{1}{2} \cdot \frac{1}{15000\ min^{-1}} = 2.00 ms$
  - Transfer time for 8KB: $t_{tr} = \frac{8\ KB}{163\ MB/s} = 0.05\ ms$

Access Time for an 8KB block: $t = t_s + t_r + t_{tr} = 5.45\ ms$

Sequential vs. Random Access

	Cheetah 15K.7
Capacity	600 GB
RPM	15000
Avg. Seek t_s	3.4 ms
Max Transfer	163 MB/s
t_r	2.0 ms
t_{tr}	0.05 ms

- Q: Read 1,000 blocks of size 8 kB
- Random access: easy
  - $t_{rnd} = 1{,}000 * 5.45\,ms = 5.45\,s$
- Sequential access
  - $t_{seq} = t_s + t_r + 1000 \cdot t_{tr} + 16 \cdot t_{s,track-to-track}$
  - $= 3.4 \text{ ms} + 2.0 \text{ ms} + 50 \text{ ms} + 3.2 \approx 58.6\,ms$
    - Cheetah 15K.7 stores an average of 512 kB per track, with a 0.2 ms track-to-track seek time;
    - 8 kB blocks are spread across 16 tracks

Sequential vs. Random Access

- Comparison
  - Random I/O: $5.45\ s$
  - Sequential I/O: $58.6\ ms$

Random I/O results in very poor disk performance!

- Guidelines
  - Sequential I/O is **much** faster than random I/O
  - **Avoid random I/O** whenever possible
  - As soon as we need at least 58.6 ms 5,450 ms = 1.07% of a file, we better read the **entire** file sequentially

Evolution of Hard Disk Technology

- Disk seek and rotational latencies have only marginally improved over the last years (10% per year)

- **But:**
  - Throughput (i.e., transfer rates) improve by 50% per year
  - Hard disk capacity grows by 50% every year

- **Therefore:**
  - Random access cost hurts even more as time progresses

Ways to Improve I/O Performance

- The latency penalty is hard to avoid

- **But:**
  - Throughput can be increased rather easily by exploiting **parallelism**

- **Idea:**
  - Use multiple disks and access them in parallel, try to hide latency

Contents

- Ch37. Hard Disk Drives
- Ch38. RAID
- Ch39. 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 (1/2)

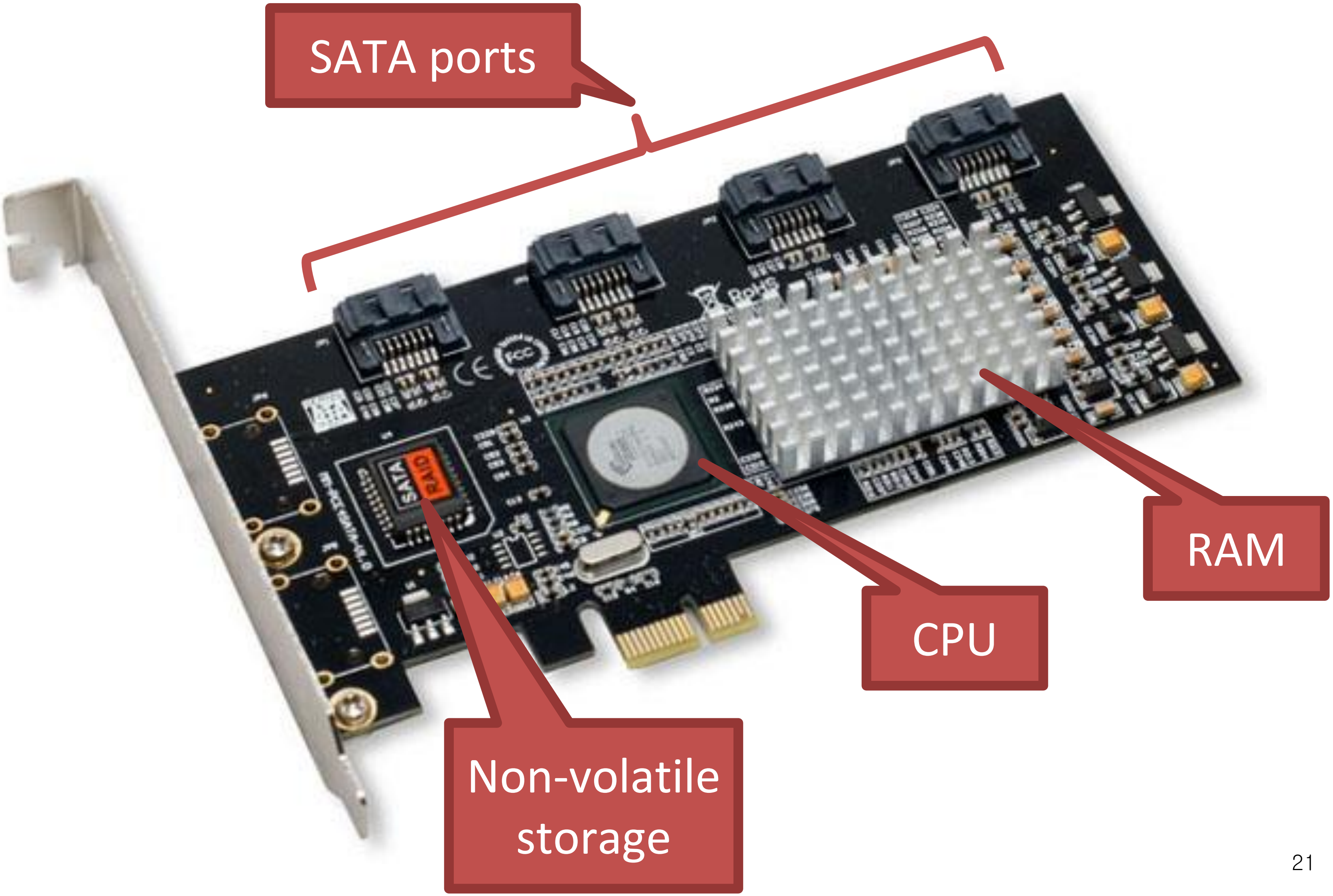
- RAID
  - use multiple disks to create the illusion of a large, faster, more reliable disk
- Externally, RAID looks like a single disk
  - i.e. RAID is transparent
  - Data blocks are read/written as usual
  - No need for software to explicitly manage multiple disks or perform error checking/recovery

Redundant Array of Inexpensive Disks (2/2)

- RAID
  - use multiple disks to create the illusion of a large, faster, more reliable disk

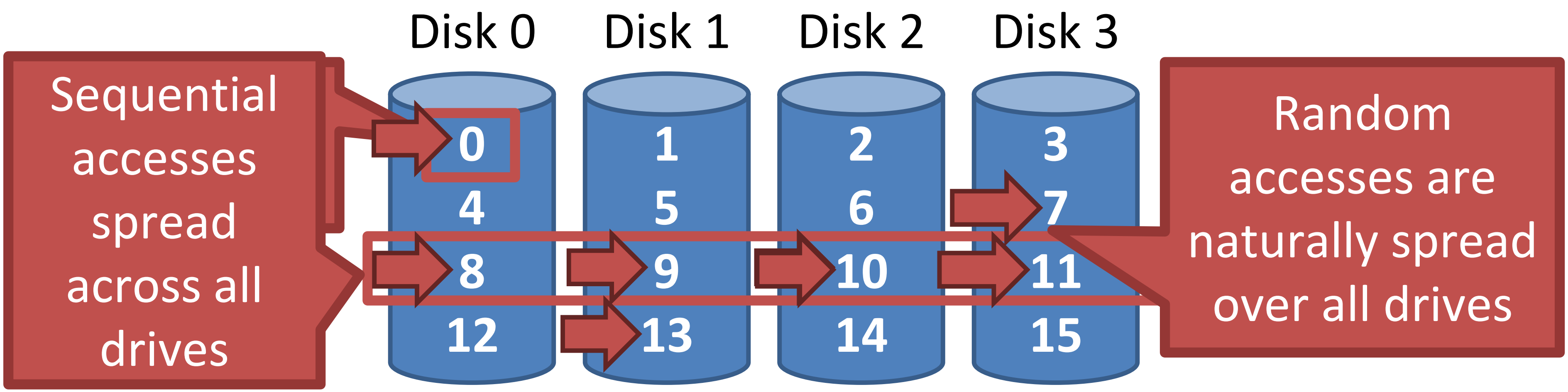
- Internally, RAID is a complex computer system
  - Disks managed by a dedicated CPU + software
  - RAM and non-volatile memory
  - Many different configuration options (RAID levels)

Example RAID Controller



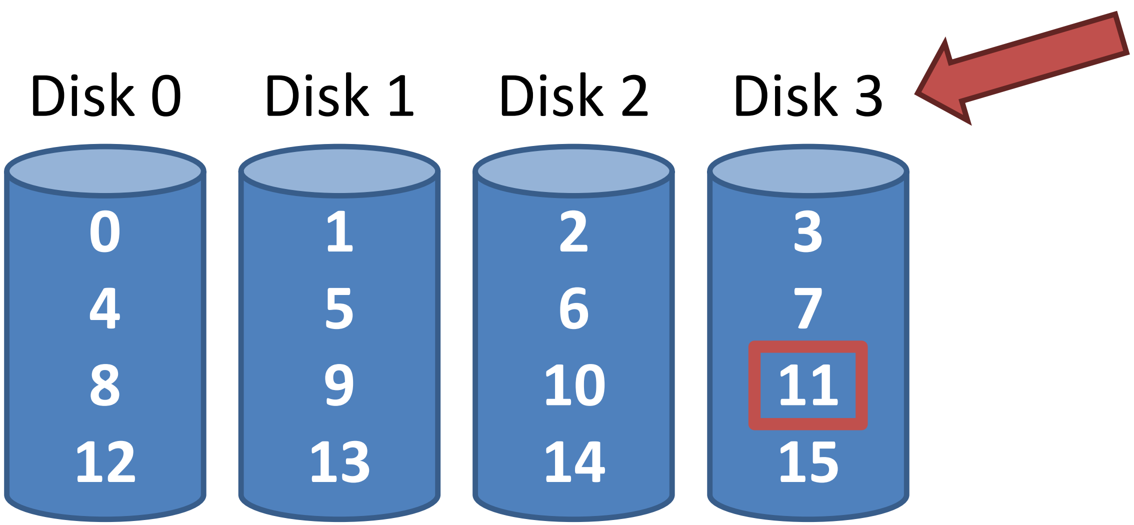
RAID 0: Striping

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

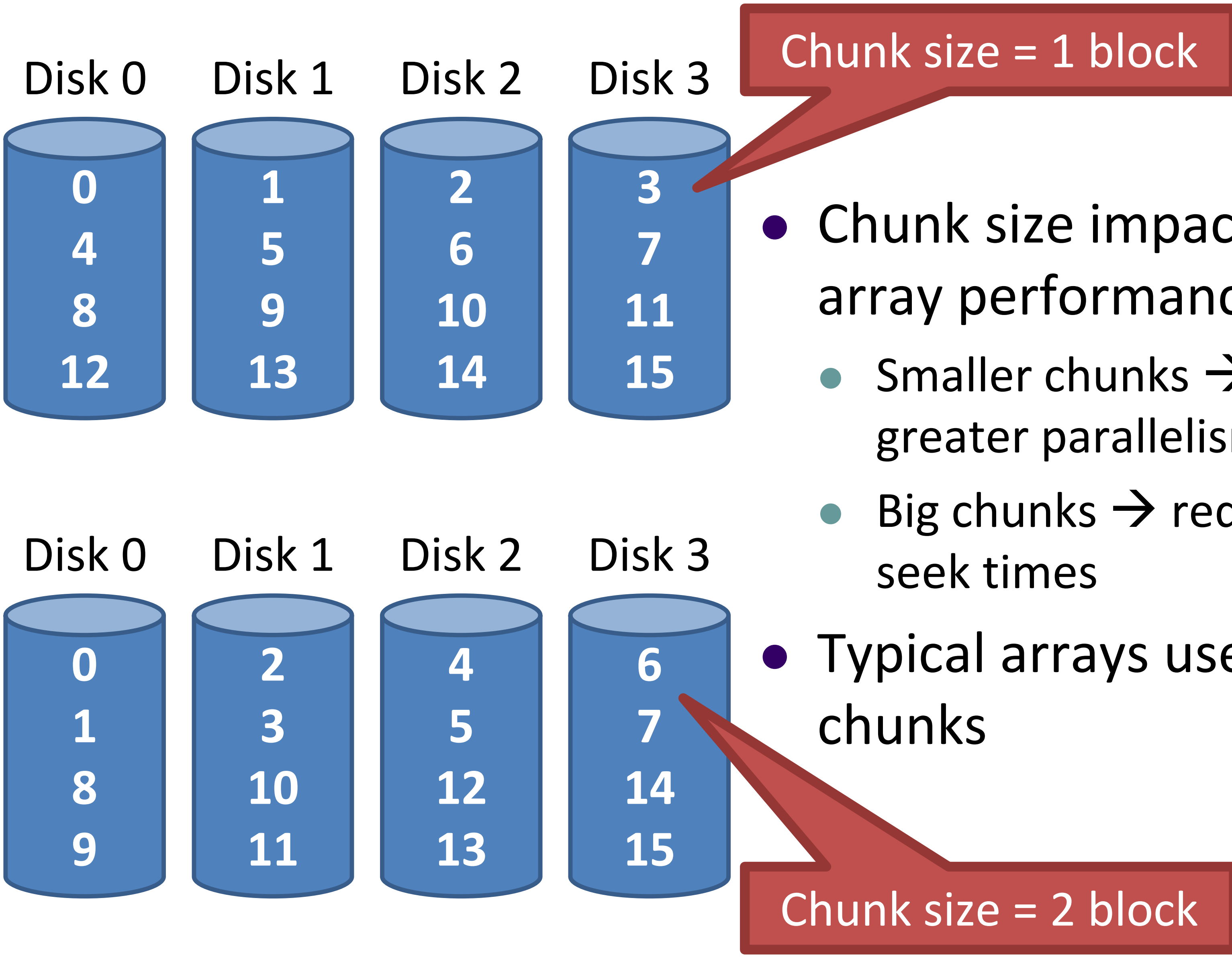


Addressing Blocks

- How do you access specific data blocks?
  - 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 (1/2)

- 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 (2/2)

- 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

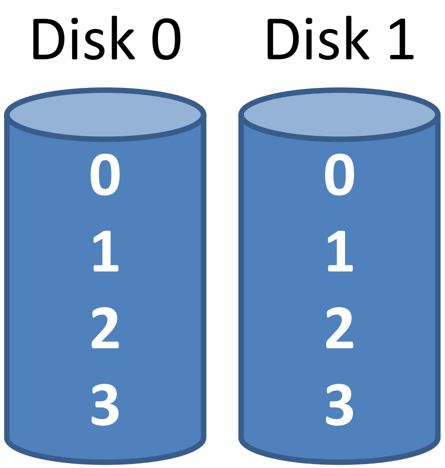
Average seek time	7 ms
Average rotational delay	3 ms
Transfer rate	50 MB/s

Analysis of RAID 0

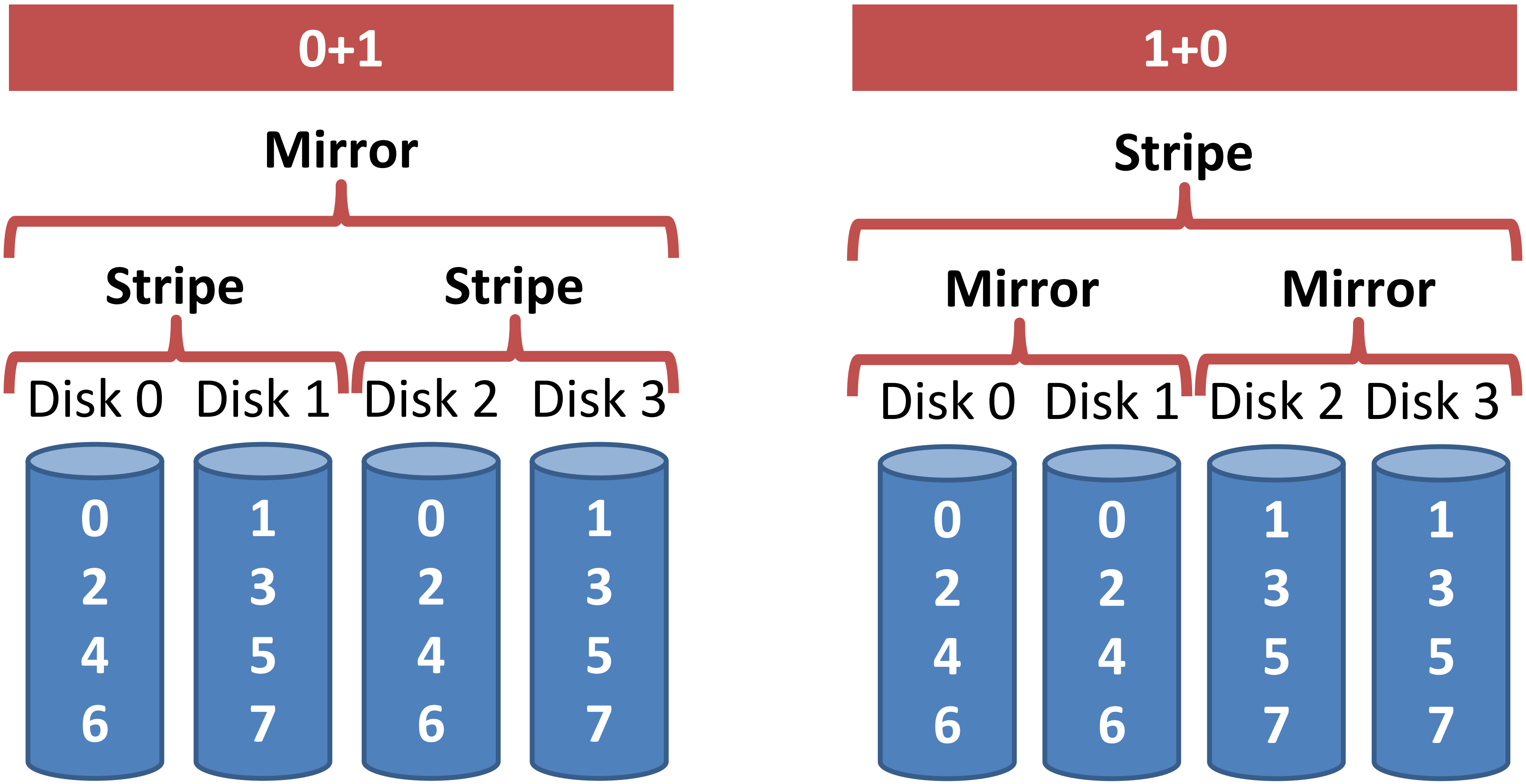
- Capacity: *N*
  - All space on all drives can be filled with data
- Reliability: 0
  - If any drive fails, data is permanently lost
- Sequential read and write: *N * S*
  - Full parallelization across drives
- Random read and write: *N * R*
  - Full parallelization across all drives

RAID 1: Mirroring

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



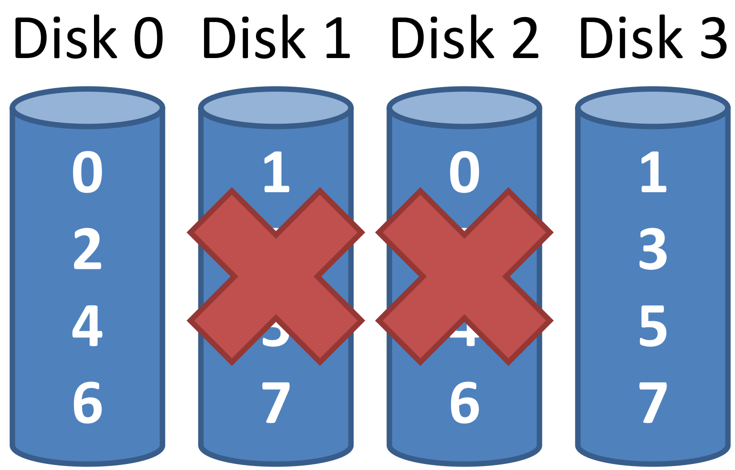
RAID 0+1 and 1+0 Examples



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

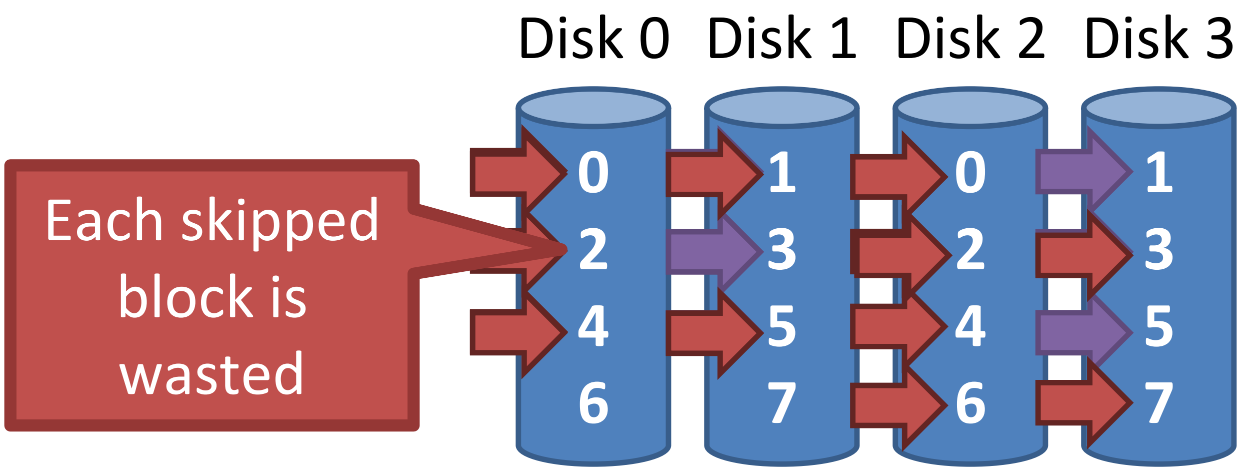
Analysis of RAID 1 (1/3)

- 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



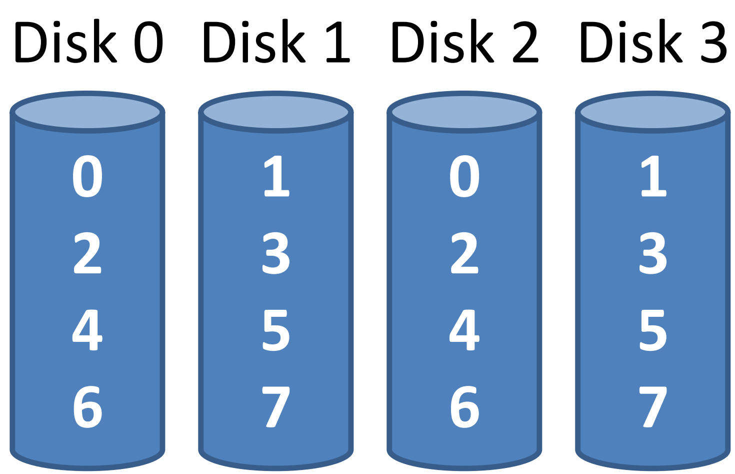
Analysis of RAID 1 (2/3)

- 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



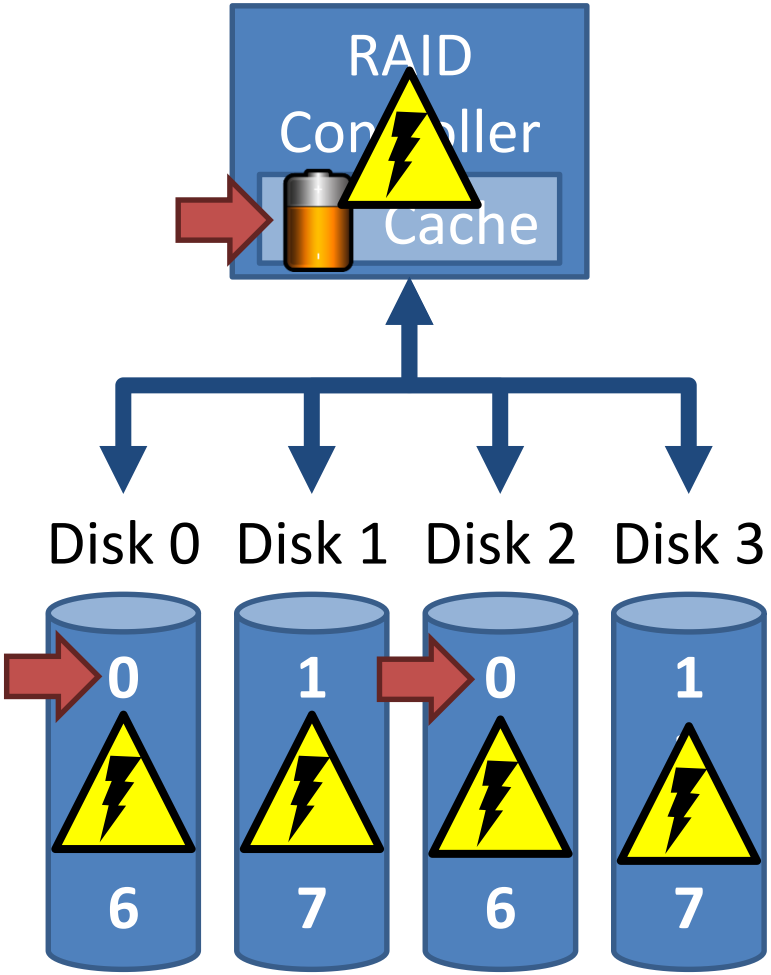
Analysis of RAID 1 (3/3)

- 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

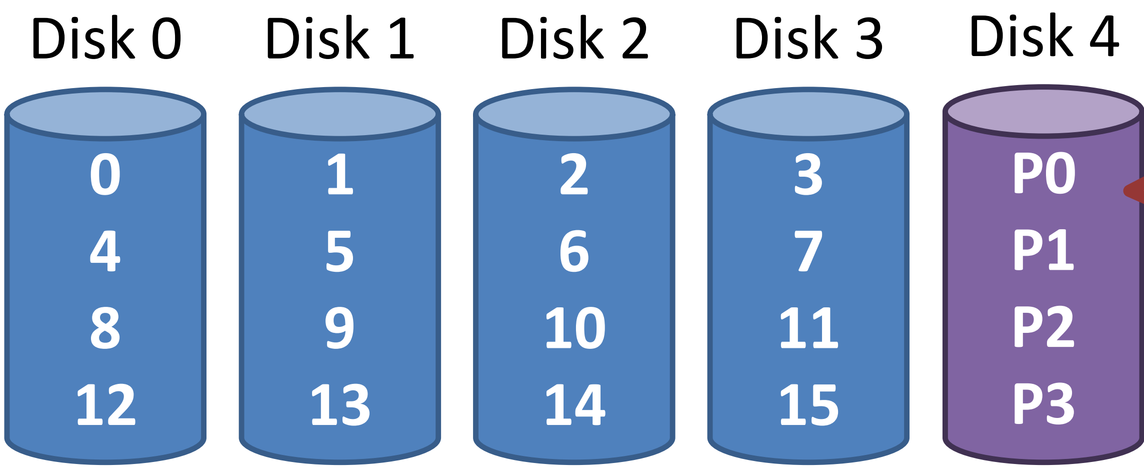
- Mirrored writes should be atomic
  - All copies are written, or none are written
- However, this is difficult to guarantee
  - Example: power failure
- Many RAID controllers include a write-ahead log
  - Battery backed, non-volatile storage of pending writes



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 *N* only stores parity information for the other *N-1* disks

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	0	1	1	0 ^ 0 ^ 1 ^ 1 = 0
0	1	0	0	0 ^ 1 ^ 0 ^ 0 = 1
1	1	1	1	1 ^ 1 ^ 1 ^ 1 = 0
0	1	1	1	0 ^ 1 ^ 1 ^ 1 = 1

Parity calculated using XOR

Updating Parity on Write

- How is parity updated when blocks are written?

1. Additive parity

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	0	0	1	0 ^ 0 ^ 0 ^ 1 = 1

Read other blocks

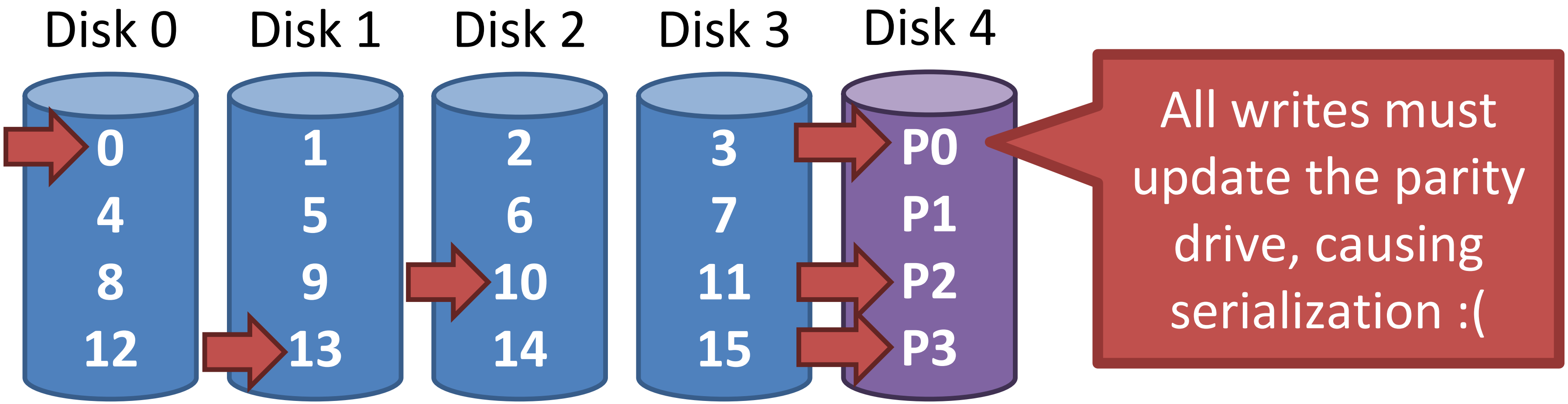
Update parity block

2. Subtractive parity

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	0	1 0	1	0 ^ 0 ^ 1 ^ 1 1

$P_{new} = C_{old} \wedge C_{new} \wedge P_{old}$

Random Writes and RAID 4

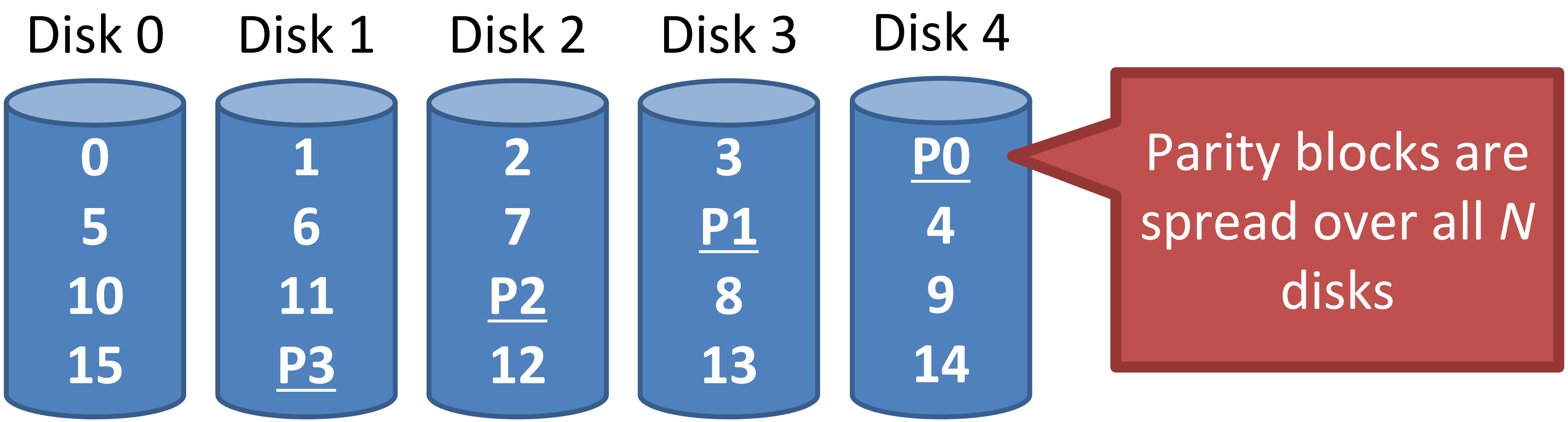


- Random writes in RAID 4
  1. Read the target block and the parity block
  2. Use subtraction to calculate the new parity block
  3. Write the target block and the parity block
- RAID 4 has terrible write performance
  - Bottlenecked by the parity drive

Analysis of RAID 4

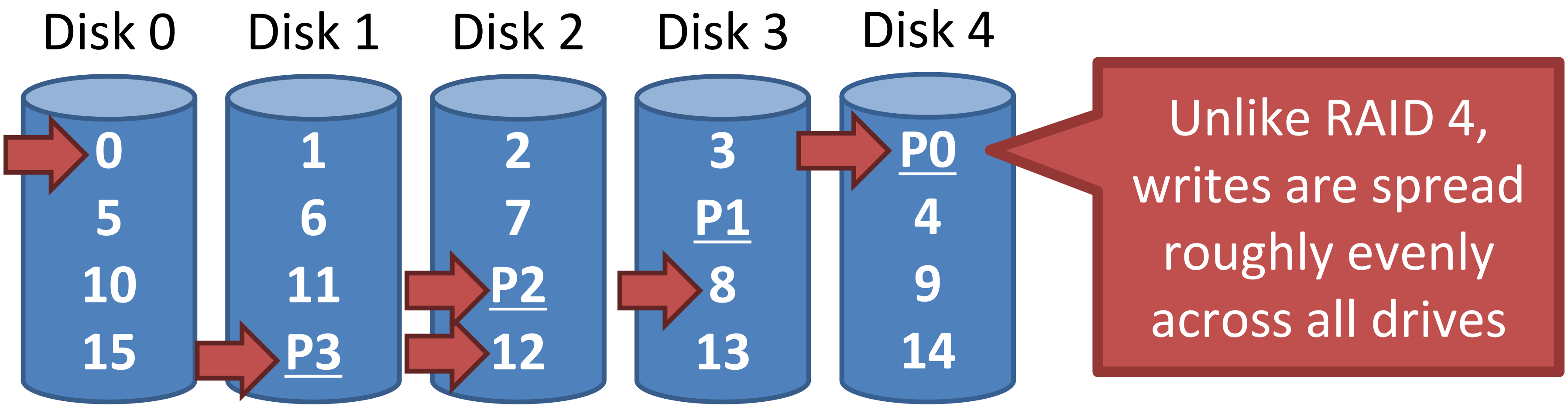
- Capacity: $N - 1$
  - Space on the parity drive is lost
- Reliability: 1 drive can fail
- Sequential Read and write: $(N - 1) * S$
  - Parallelization across all non-parity blocks
- Random Read: $(N - 1) * R$
  - Reads parallelize over all but the parity drive
- Random Write: $R / 2$
  - Writes serialize 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 ^ 0 ^ 1 ^ 1 = 0
1	0	0	0 ^ 1 ^ 0 ^ 0 = 1	0
1	1	1 ^ 1 ^ 1 ^ 1 = 0	1	1
1	0 ^ 1 ^ 1 ^ 1 = 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

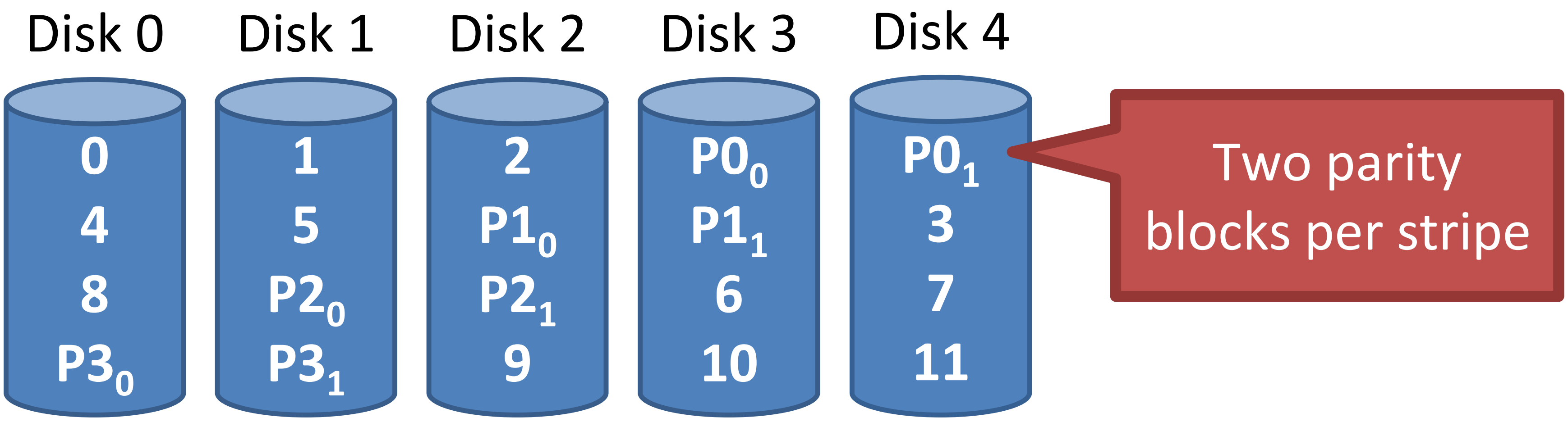
- Capacity: *N – 1* [same as RAID 4]
- Reliability: 1 drive can fail [same as RAID 4]
- Sequential Read and write: *(N – 1) * S* [same]
  - Parallelization across all non-parity blocks
- Random Read: *N * R* [vs. *(N – 1) * R*]
  - Unlike RAID 4, reads parallelize over all drives
- Random Write: *N /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

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

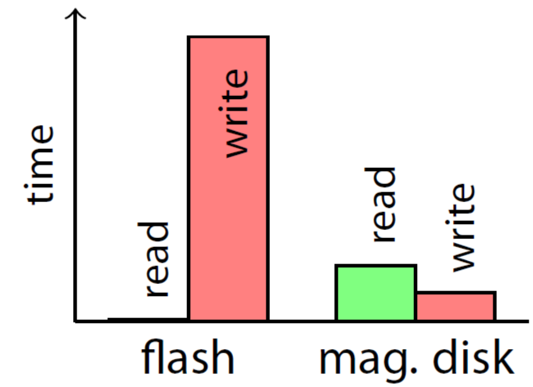
Contents

- Ch37. Hard Disk Drives
- Ch38. RAID
- Ch39. 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

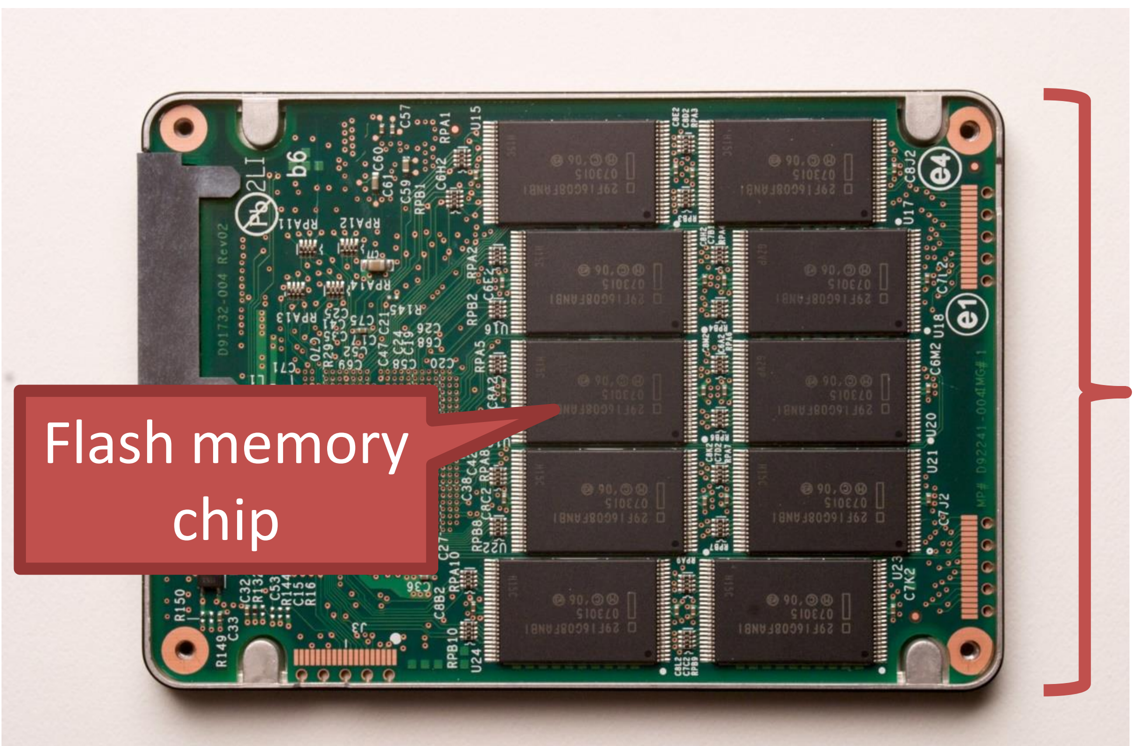
Solid State Drives (SSD) (1/2)



- SSDs provide **very low-latency random read access** (< 0.01 ms)
- **Random writes** however, are significantly **slower** than on traditional magnetic drives:
  - (Blocks of) Pages have to be **erased** before they can be updated
  - Once pages have been erased, sequentially writing them is almost as fast as reading

Solid State Drives (SSD) (2/2)

- 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



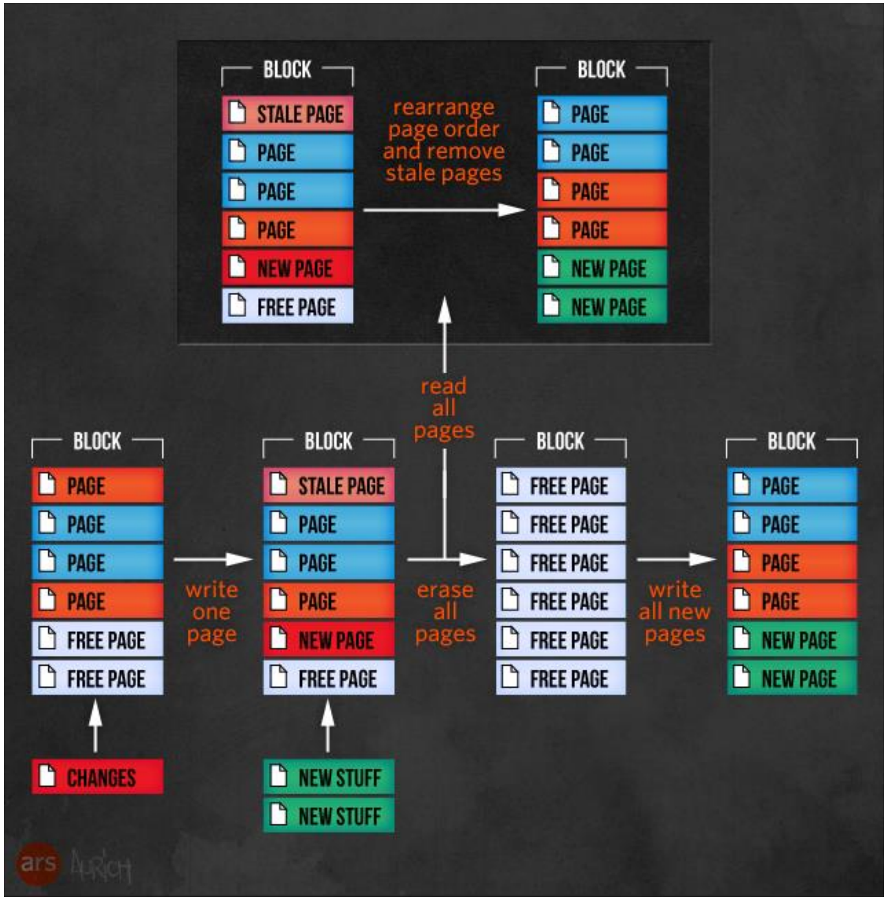
Advantages of SSDs

- 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

SSDs: Page-Level Writes, Block-Level Deletes

- Typical **page size**: 128 kB
- SSDs erase **blocks of pages**:
  - block ≈ 64 pages (8 MB)

- Example
  - Perform block-level delete to accomodate new data pages



https://arstechnica.com/gadgets/2015/04/ask-ars-my-ssd-does-garbage-collection-so-i-dont-need-trim-right/

Example: Seagate Pulsar.2

- Performance characteristics:
  - NAND flash memory, 800 GB capacity
  - standard 2.5” enclosure, no moving/rotating parts
  - data read/written in pages of 128 kB size
  - transfer rate 370 MB/s
- Q: What is the access time to read an 8KB block?
  - Average seek time: $t_s = 0.00\ ms$
  - Average rotational delay: $t_r = 0.00 ms$
  - Transfer time for 8KB: $t_{tr} = \frac{8\ KB}{370\ MB/s} = 0.30\ ms$

Access Time for an 8KB block: $t = t_s + t_r + t_{tr} = 0.3\ ms$

Sequential vs. Random Access with SSDs

	Pulsar.2
Avg. Seek t_s	0.0 ms
t_r	0.0 ms
t_{tr}	0.30 ms

- Q: Read 1,000 blocks of size 8 kB
- Random access: easy
  - $t_{rnd} = 1{,}000 * 0.3\ ms = 0.3\ s$
- Sequential access
  - $t_{seq} = \left\lceil \frac{1000 \cdot 8KB}{128\ kB} \right\rceil \cdot t_{tr} \approx 18.9\ ms$
    - Pulsar.2 (sequentially) reads data in 128 kB chunks.8

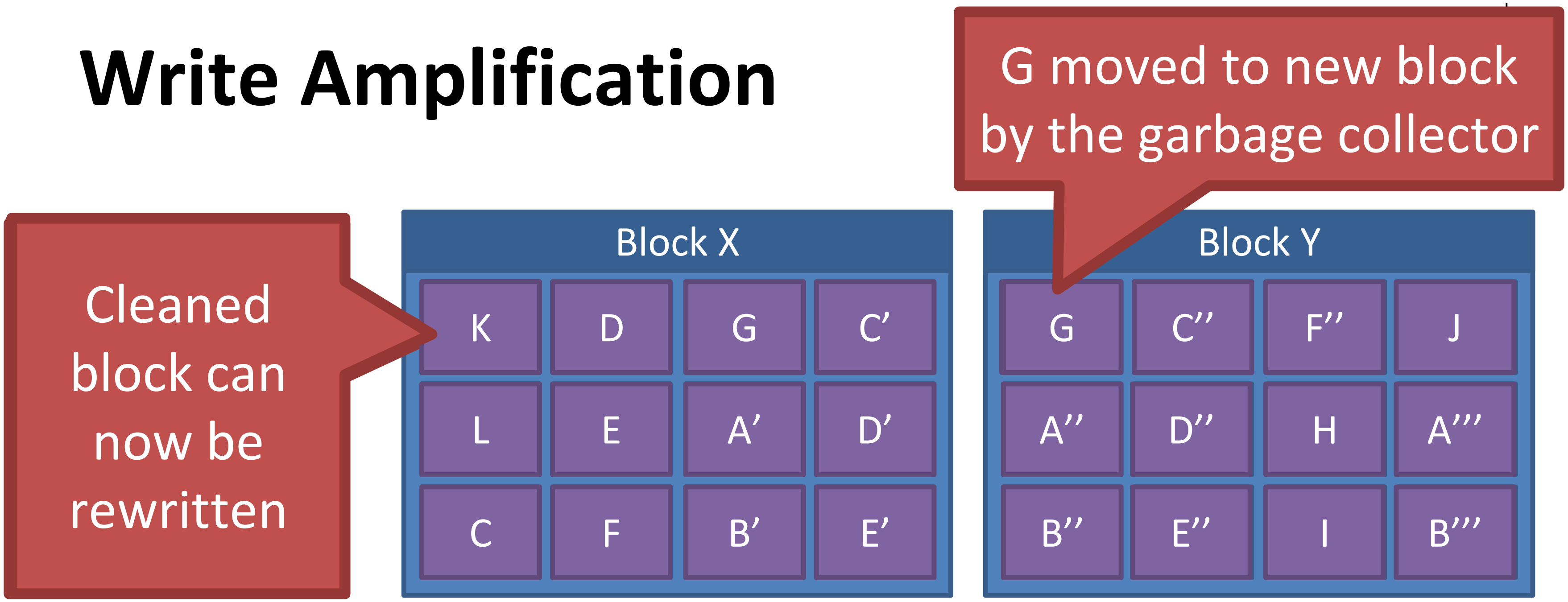
=> Sequential I/O still beats random I/O (but random I/O is more feasible again)

- Adapting database technology to these characteristics is a current research topic

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



- 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

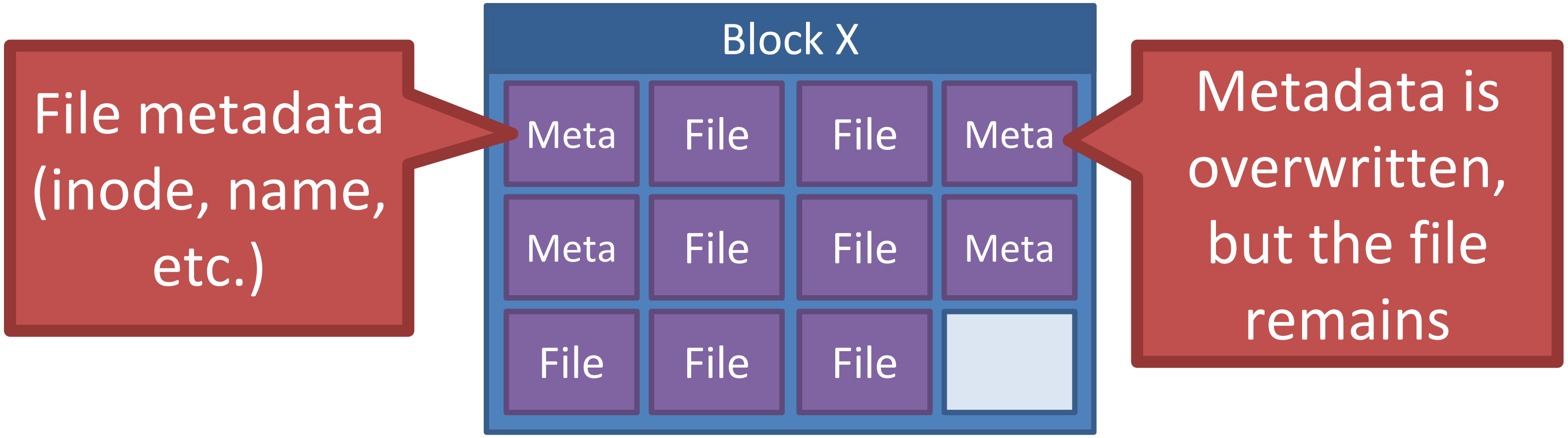
- Garbage collection (GC) is vital for the performance of SSDs
- Older SSDs had fast writes up until all pages were written once
  - Even if the drive has lots of “free space,” each write is amplified, thus reducing performance
- Many SSDs over-provision to help the GC
  - 240 GB SSDs actually have 256 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

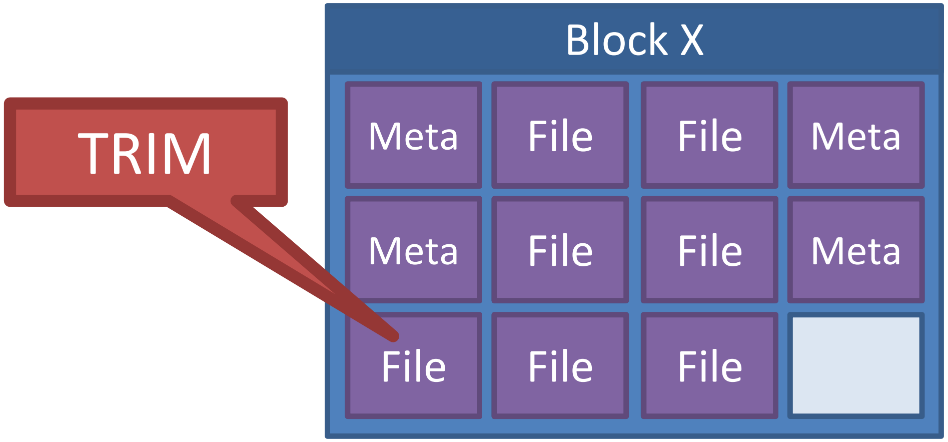


1. File is written to SSD
2. File is deleted
3. The GC executes
   - 9 pages look valid to the SSD
   - The OS knows only 2 pages are valid

- 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

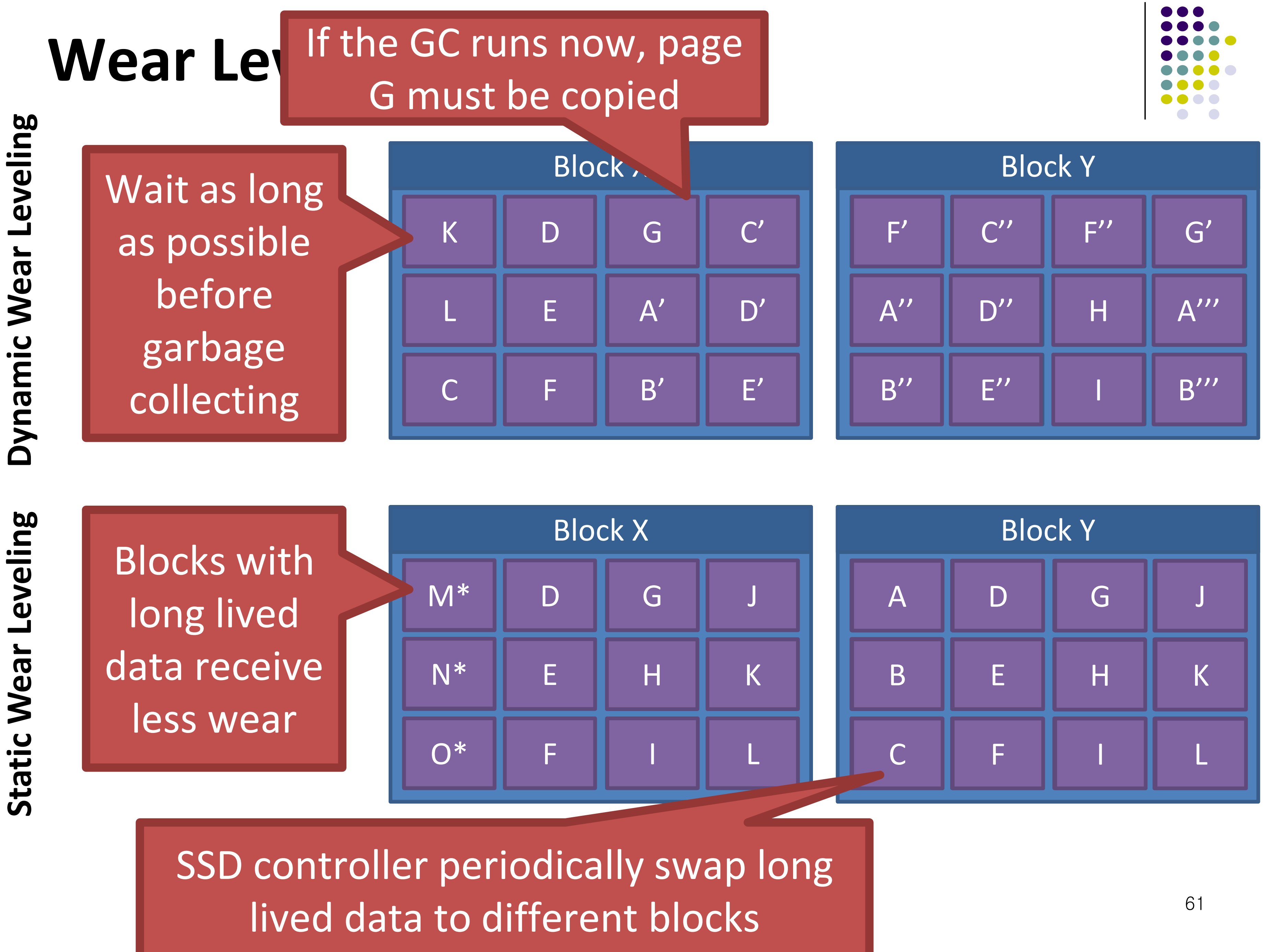


- OS support for TRIM
  - Win 7, OSX Snow Leopard, Linux 2.6.33, Android 4.3
- Must be supported by the SSD firmware

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

Wear Lev


If the GC runs now, page G must be copied
Dynamic Wear Leveling
Wait as long as possible before garbage collecting
Block
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

Q&A