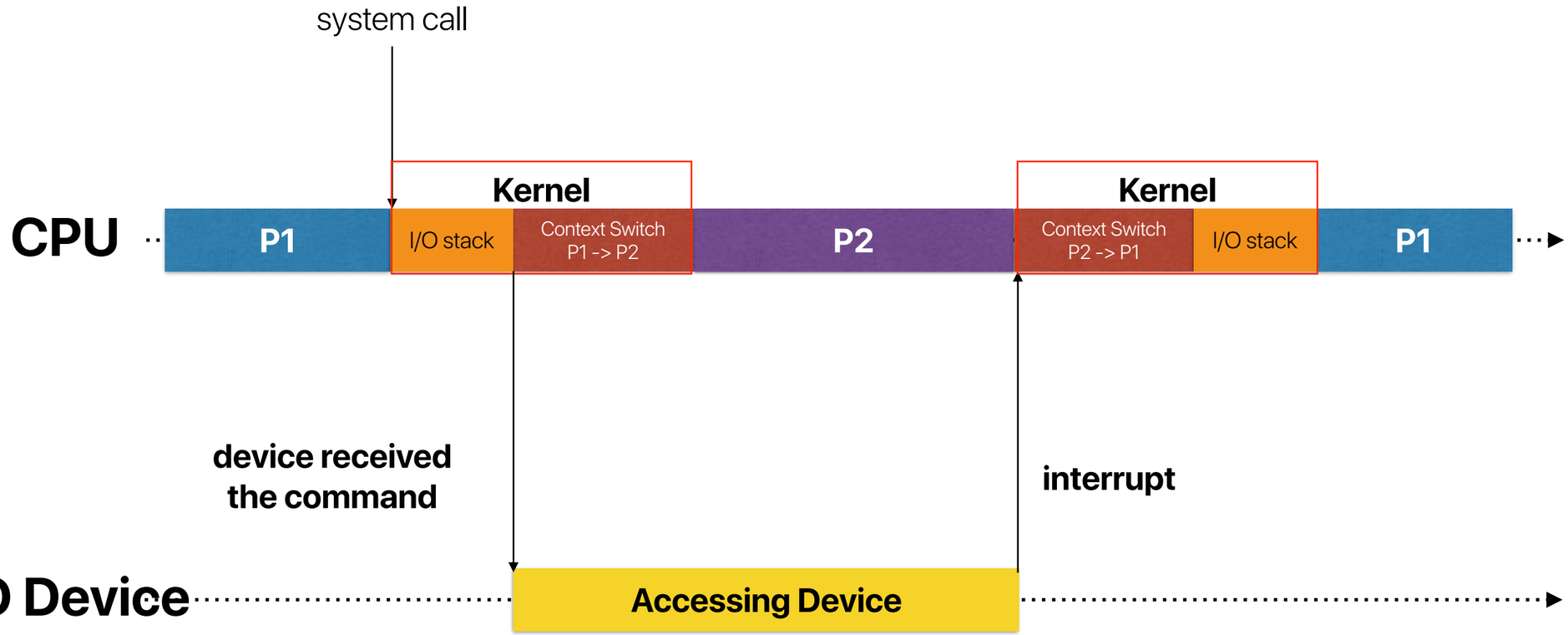
Non-volatile Storage & Streamline Data Exchanges

Hung-Wei Tseng

Polling/Interrupt/DMA/PIO?

	Polling	Interrupt
DMA	CPU OPs more than IRQ Shorter latency High throughput	Lowest CPU Operations Longer latency High throughput
PIO	Most CPU operations Shorter latency Low throughput	CPU OPs more than IRQ Longer latency Low throughput

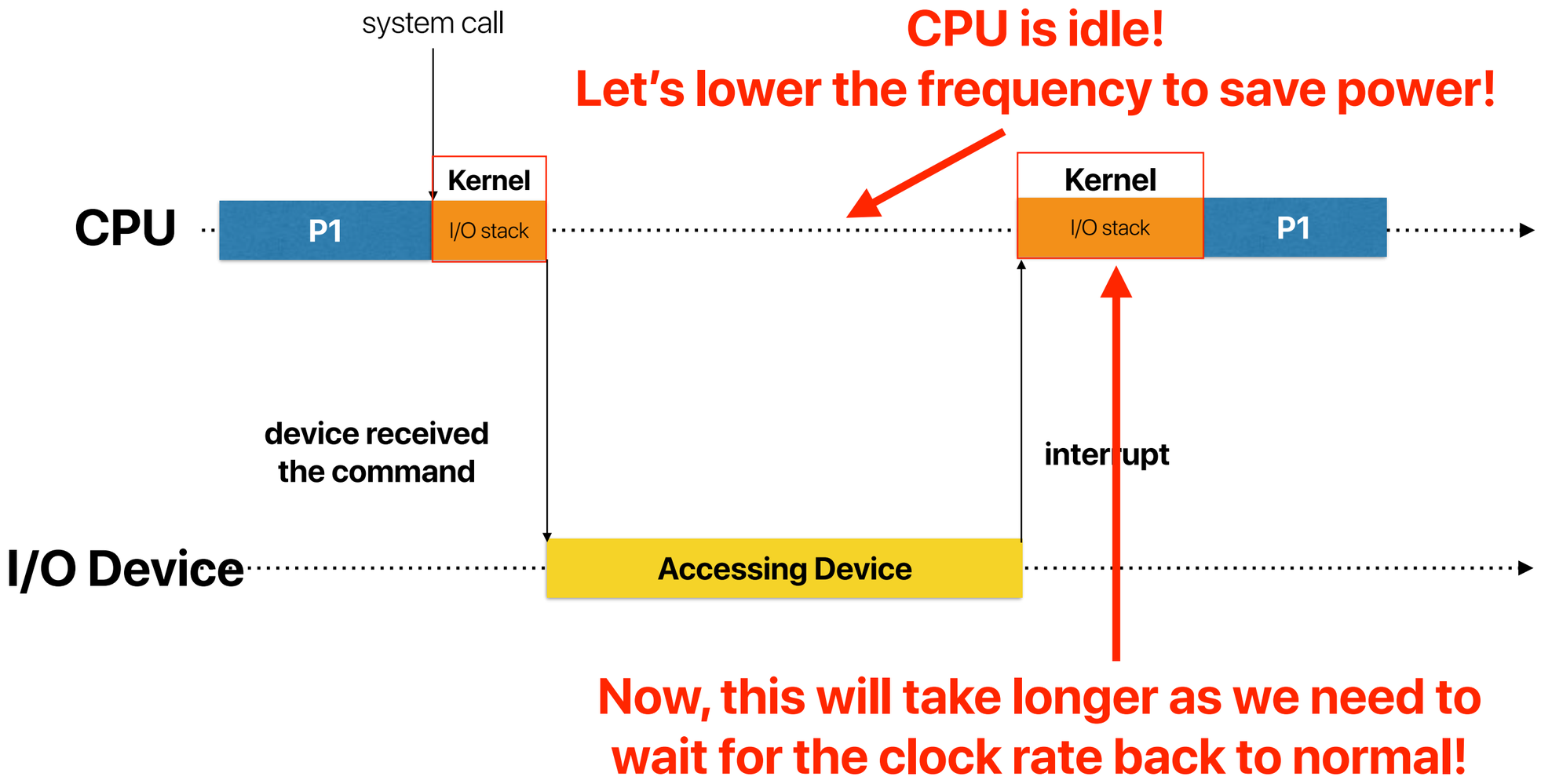
To switch or not to switch that's the question.



If $T_{\text{Context switch P1->P2}} + T_{\text{Context switch P2->P1}} < T_{\text{Accessing peripherals}}$

makes sense to context switch

What if we don't switch?

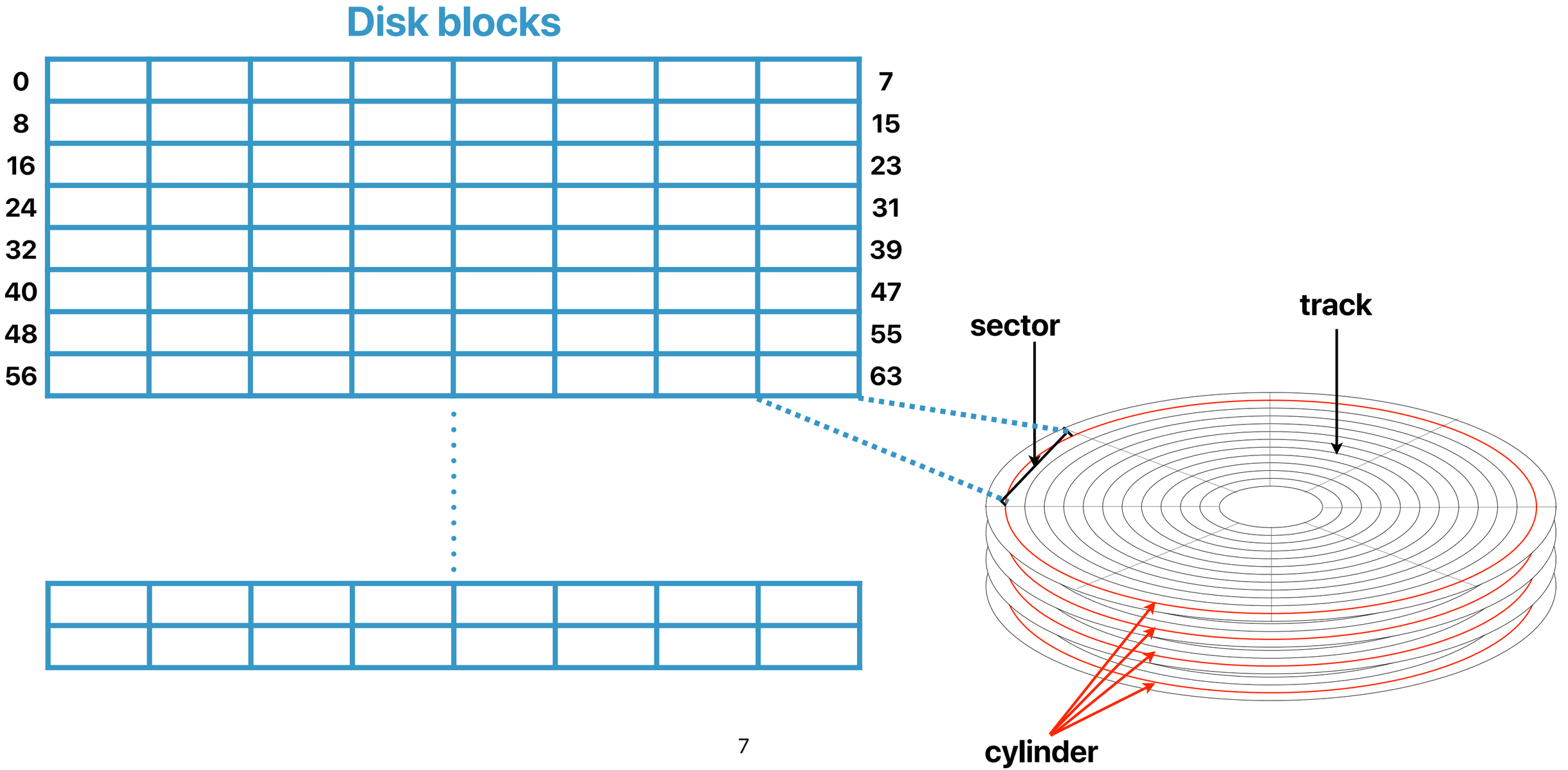


Outline

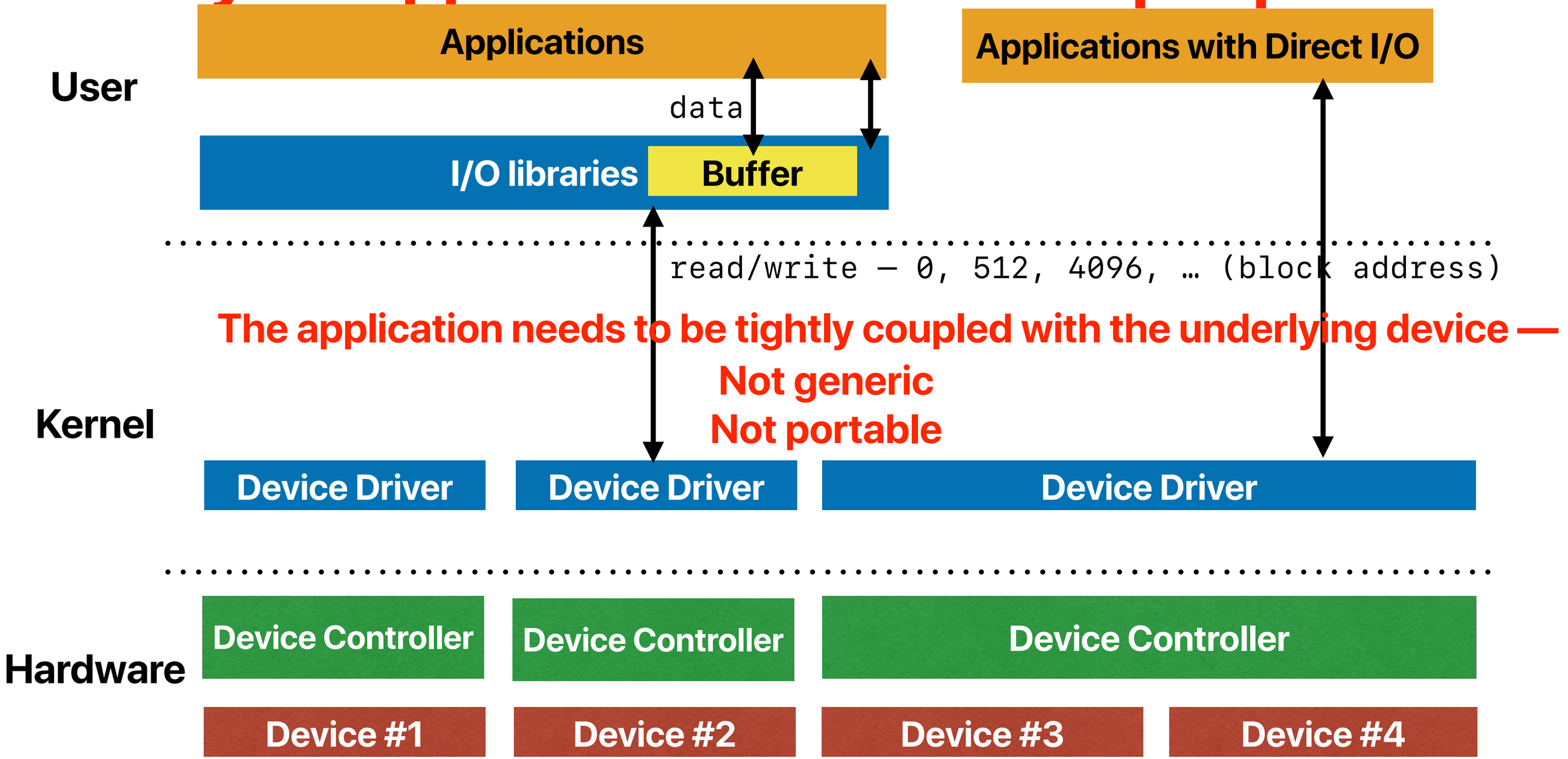
- Storage systems — starting from HDD
- NVMe (Non-volatile memory express)
- Efficient communication on PCIe
- Near-storage processing
  - SmartSSDs
  - ActiveDisks

Hard Disk Drives

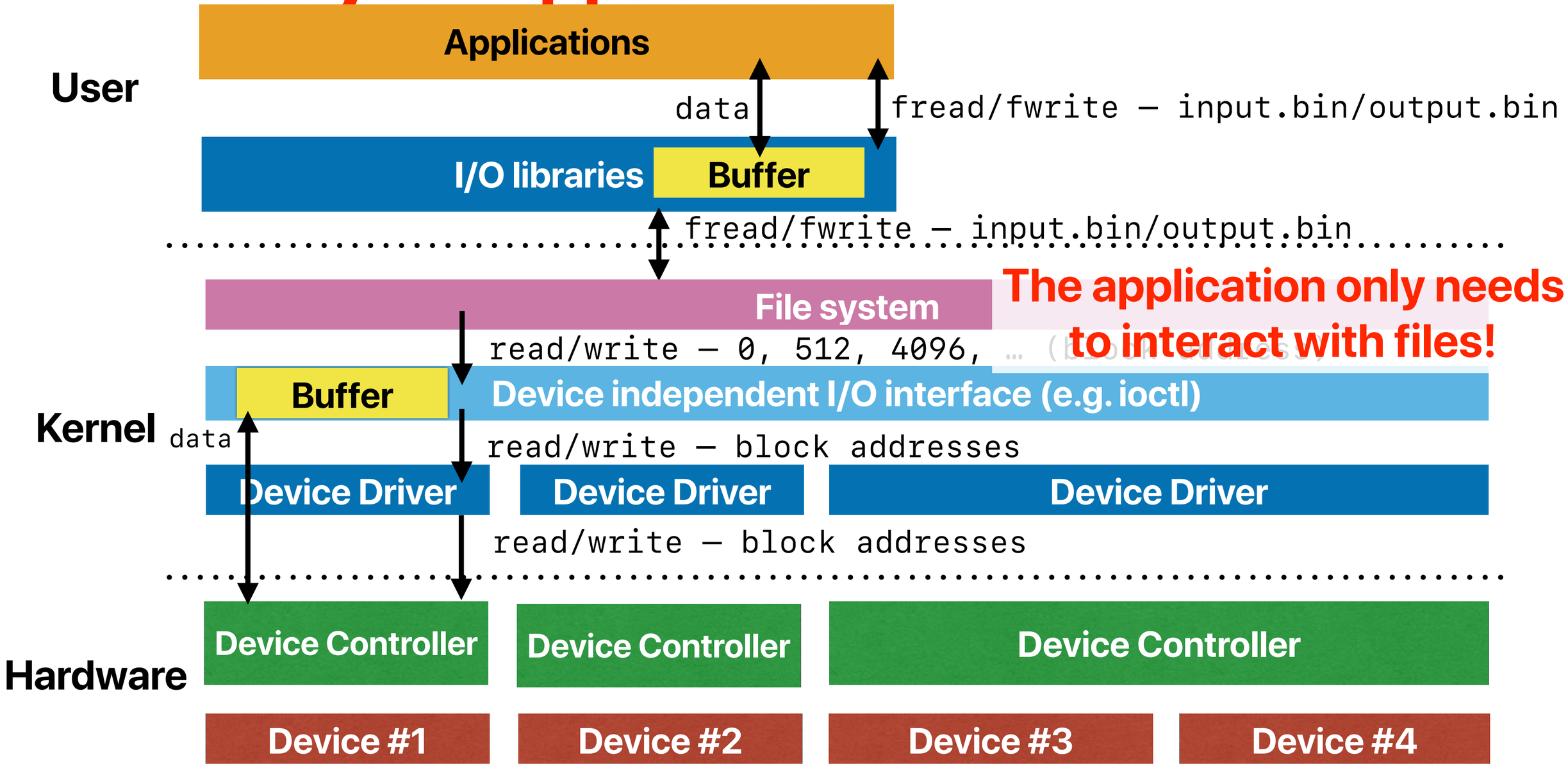
Numbering the disk space with block addresses



How your application interact with peripherals

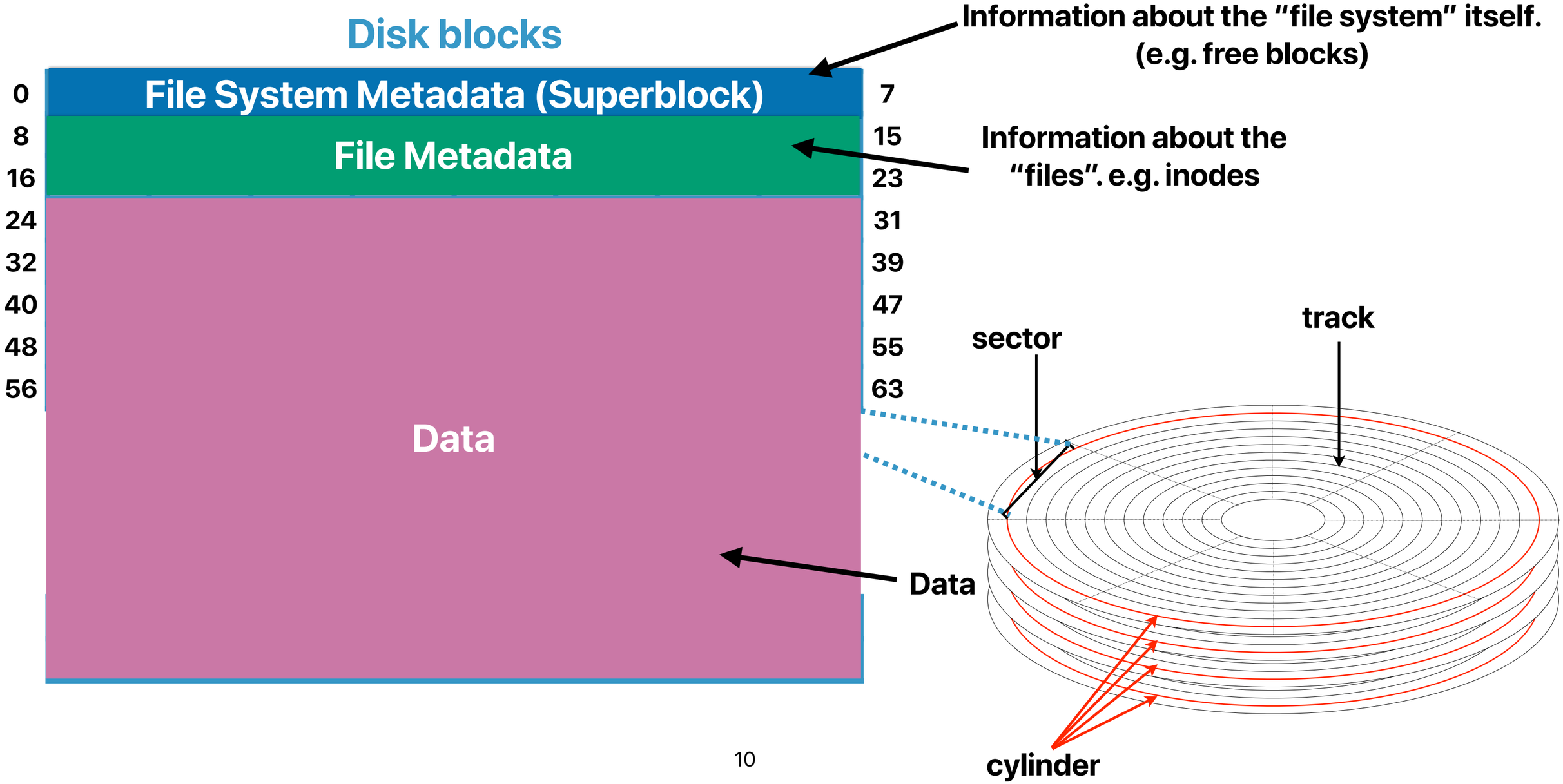


How your application reaches H.D.D.

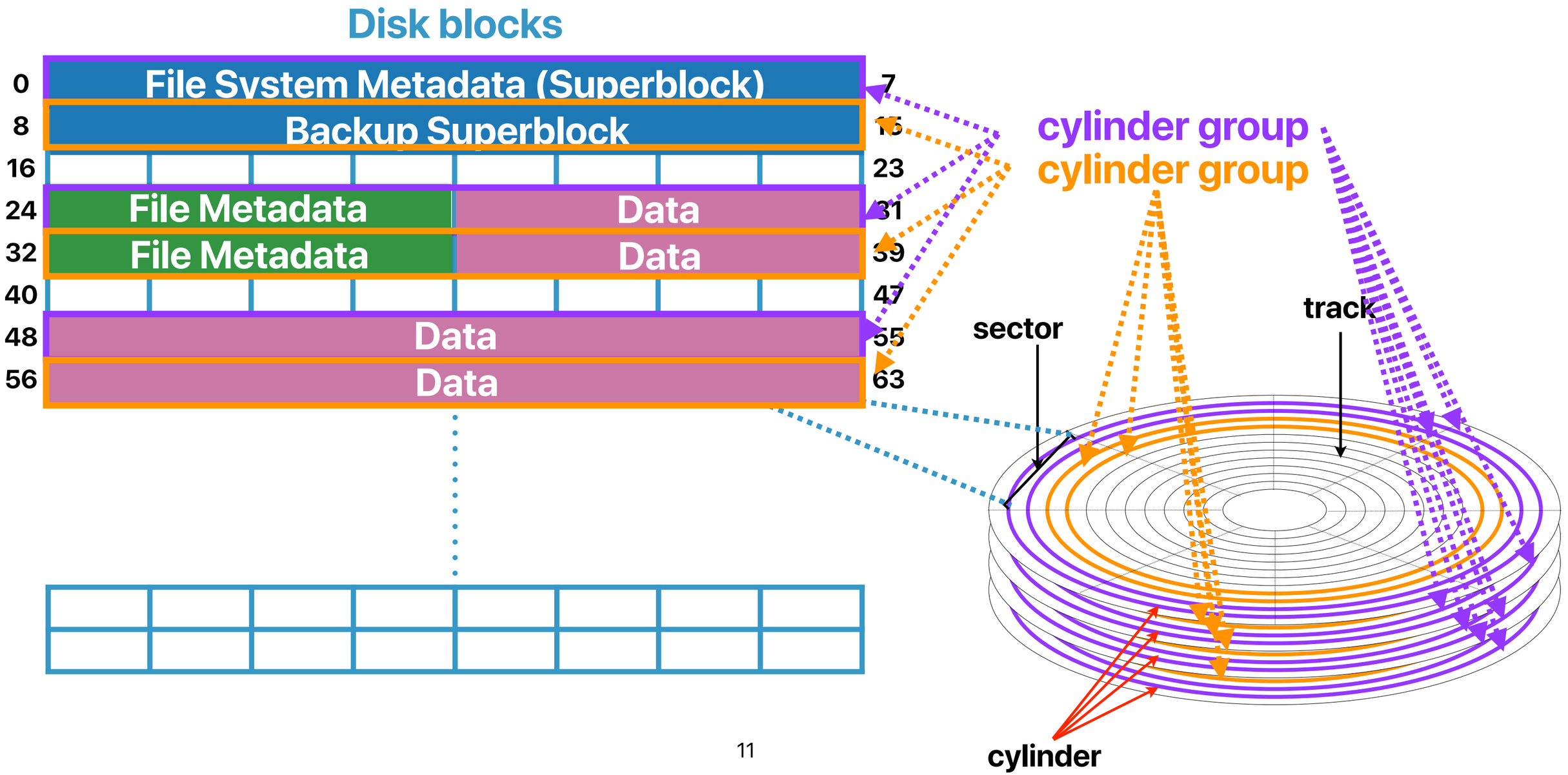


The application only needs to interact with files!

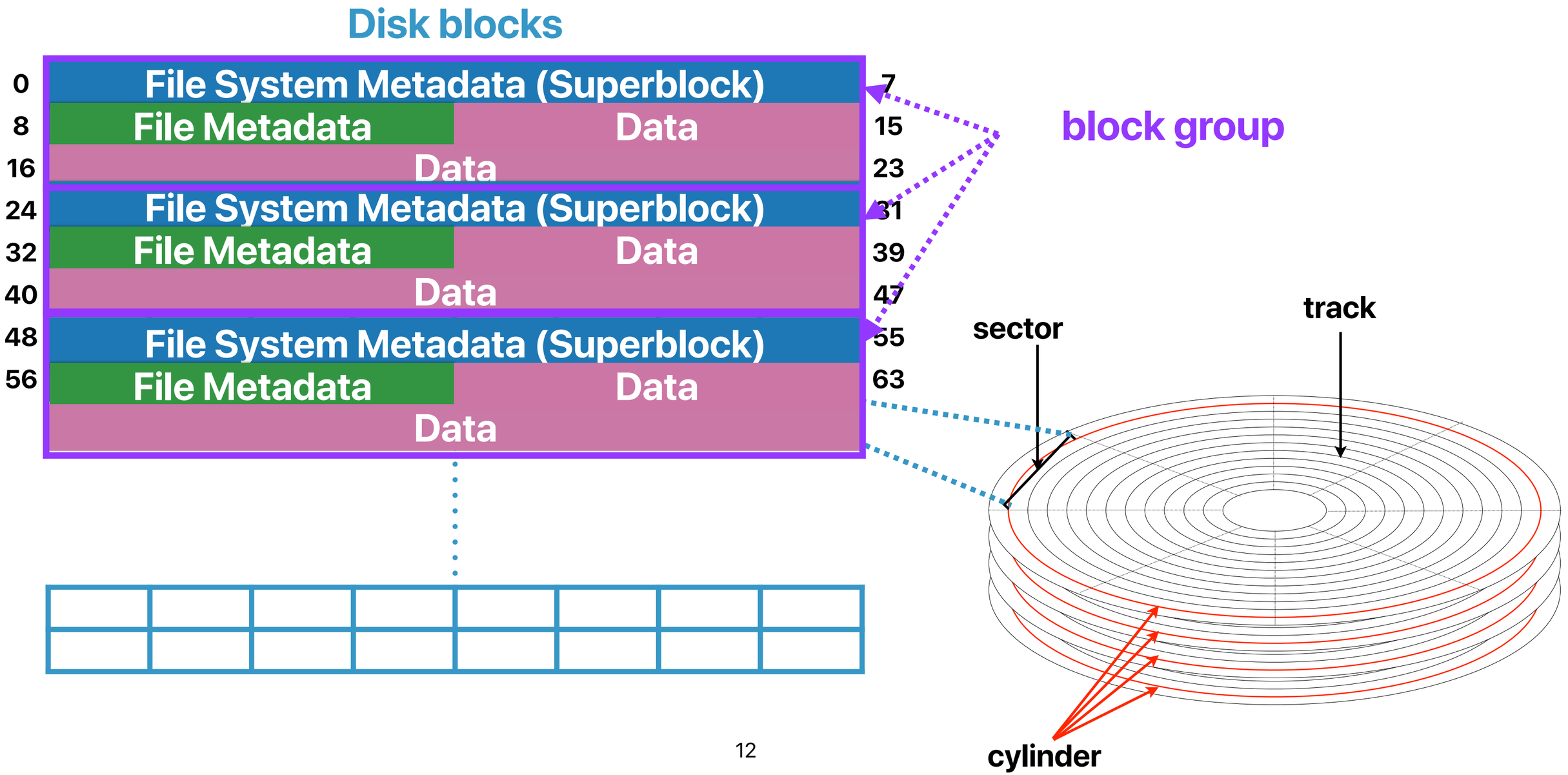
How the original UNIX file system use disk blocks



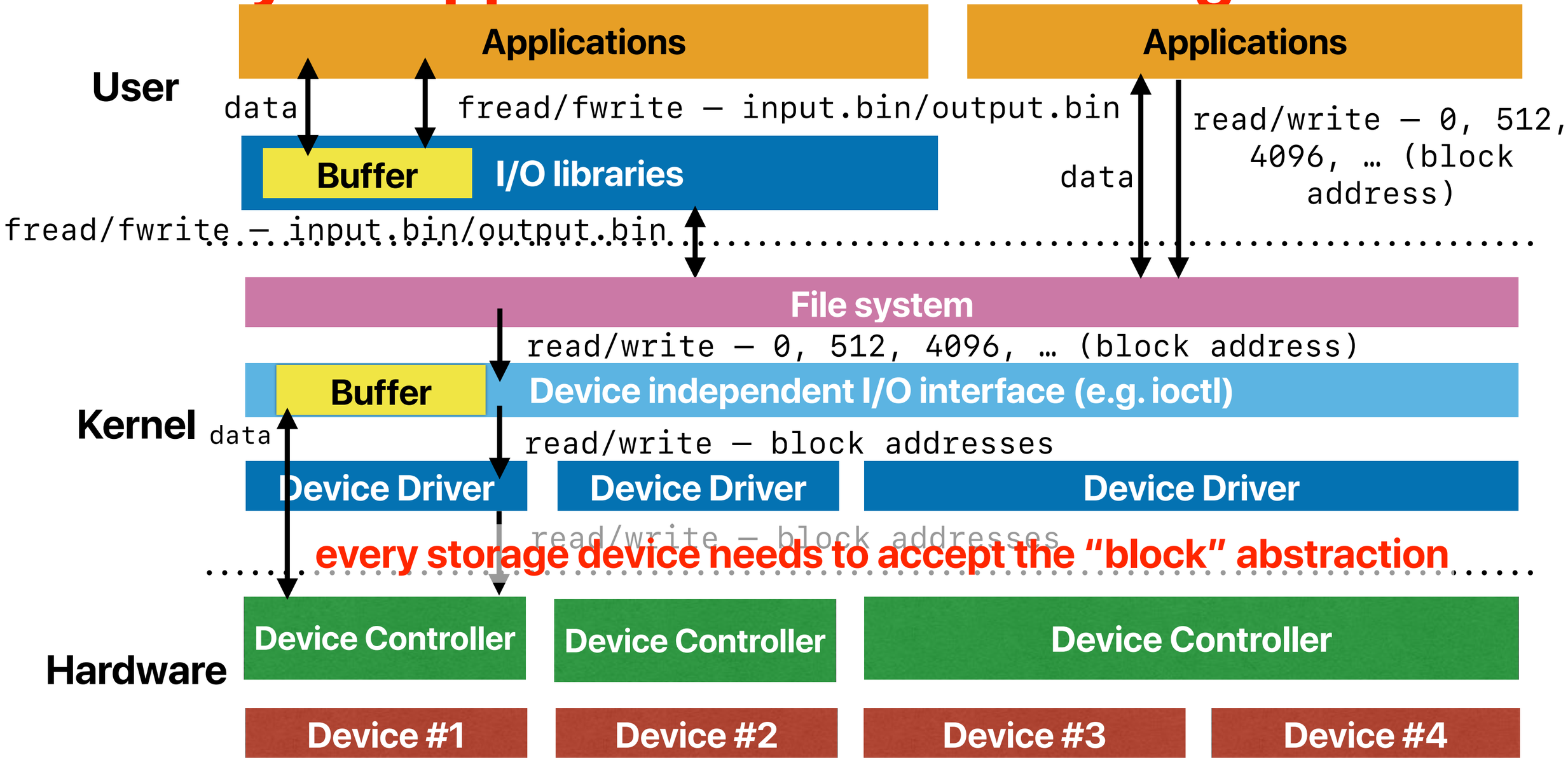
How FFS use disk blocks



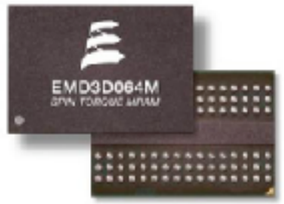
How ExtFS use disk blocks



How your application reaches storage devices



Non-volatile memory technologies



	H.D.D	Flash	Optane	STT-MRAM
Latency	~ 10-15 ms	~ 100 us (read) ~ 1 ms (write)	7 us (read) 18 us (write)	35 ns
Bandwidth	~200 MB/Sec	3.5 GB/sec (read) 2.1 GB/sec (write)	1.35 GB/sec (read) 290 MB/sec (write)	
Dollar/GB	0.0295	0.583	2.18	

Flash is still the most convincing technology for now

3:00

Based on what you've read, what are the differences between magnetic hard disks & NAND flash?

NAND Flash v.s. hard disks

NAND Flash v.s. hard disks

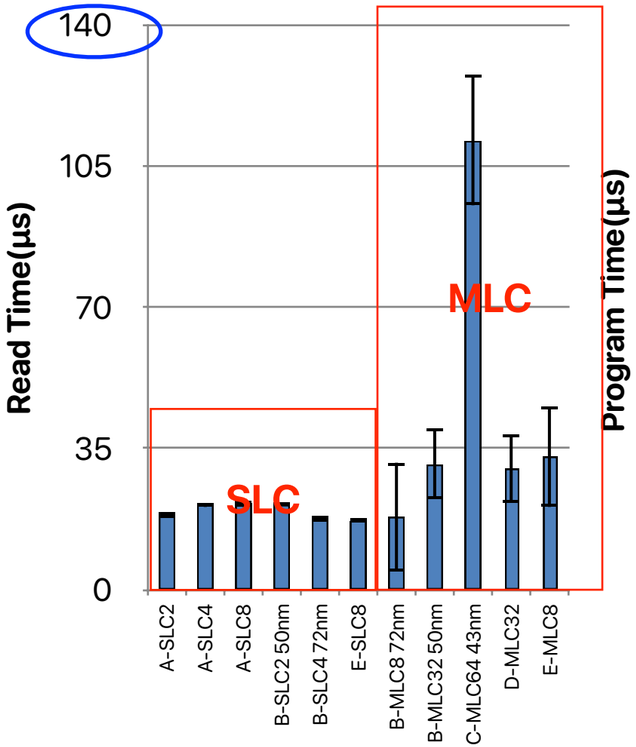
		NAND	Hard Disks
granularities	read	page	sector
	write	page	sector
	erase	block (a group of pages)	sector
performance	read	us	ms
	write	100s of us — ms	ms
	erase	ms	ms
program		1,000—100,000 times	unlimited

NAND flash is just odd!

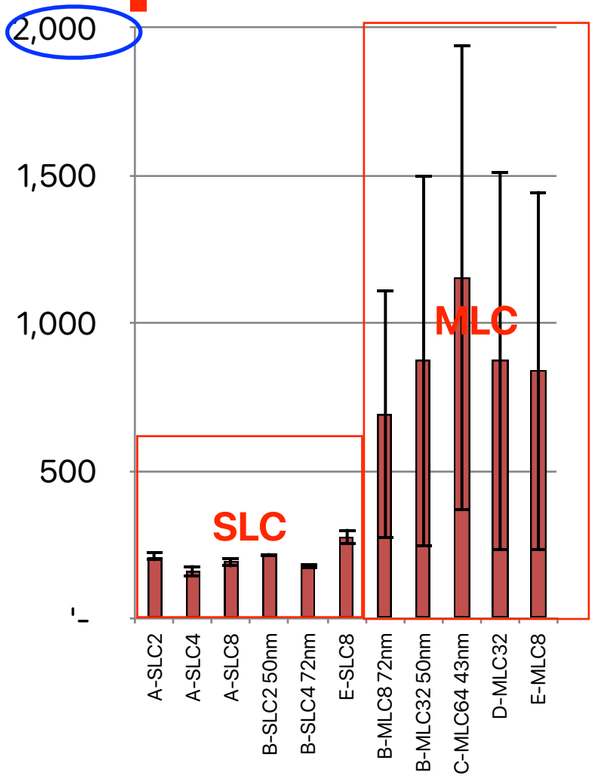
- Modern SSDs are based on NAND flash memory
- Different operation granularities
  - Read/Program in pages
  - Erase in blocks (64-384 pages)
- Performance of operation varies
  - Read — tens of us
  - Program — hundreds of us
  - Erase — ms
- Limited erase cycles
  - Only 1000 times for “QLC”

Flash performance

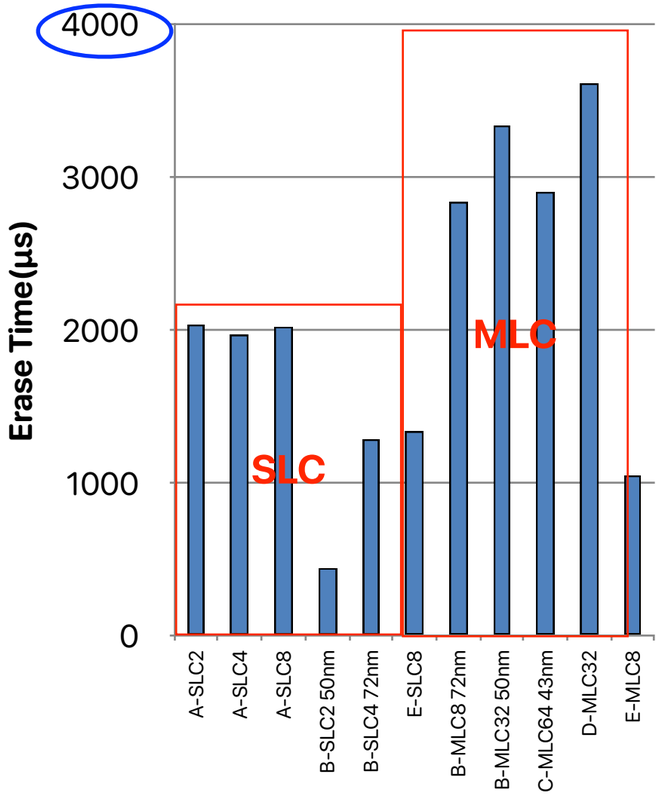
Not a good practice



Reads: less than 150us



Program/write: less than 2ms



Erase: less than 3.6ms

Similar relative performance for reads, writes and erases

Laura M. Grupp, Adrian M. Caulfield, Joel Coburn, Steven Swanson, Eitan Yaakobi, Paul H. Siegel, and Jack K. Wolf. Characterizing flash memory: anomalies, observations, and applications. In MICRO 2009.

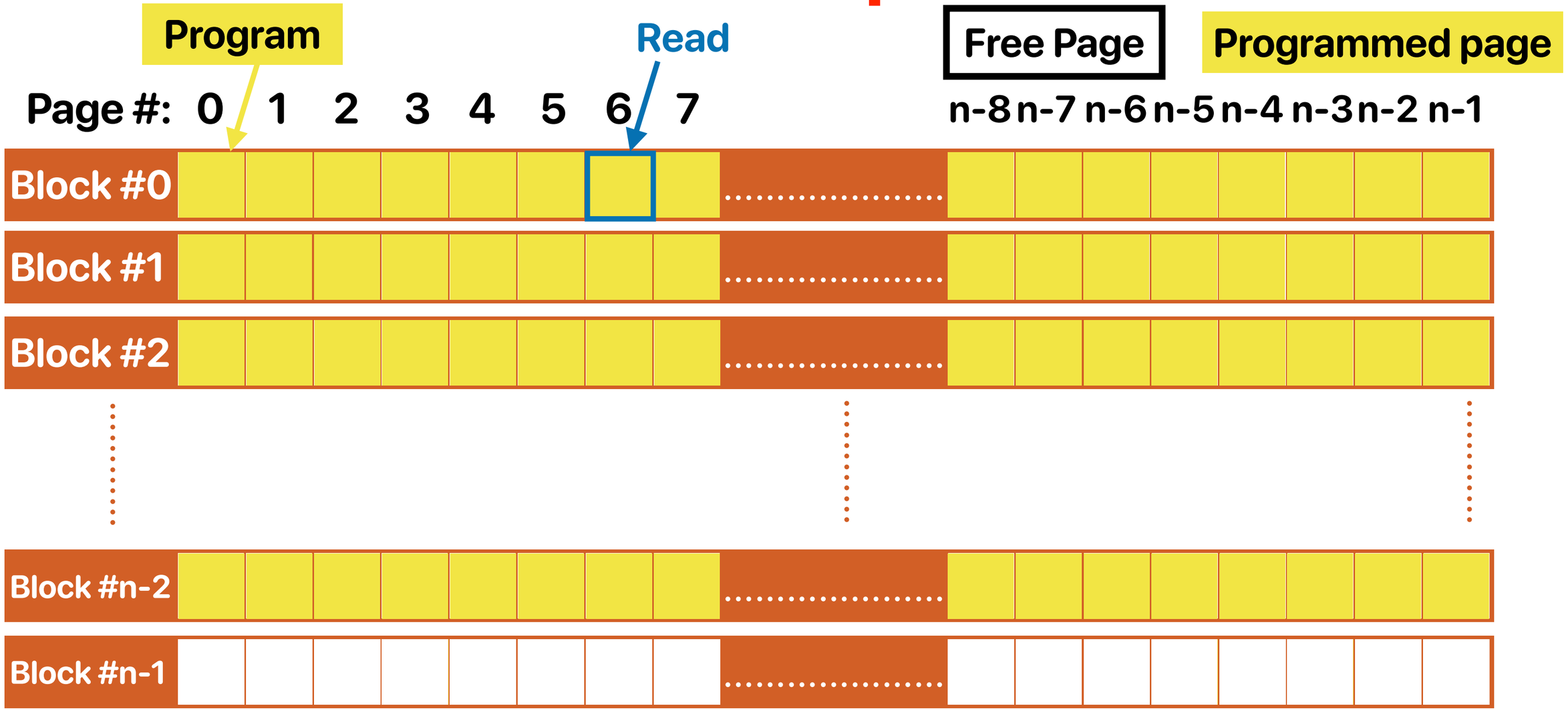
The architecture of an SSD

Controller + Registers

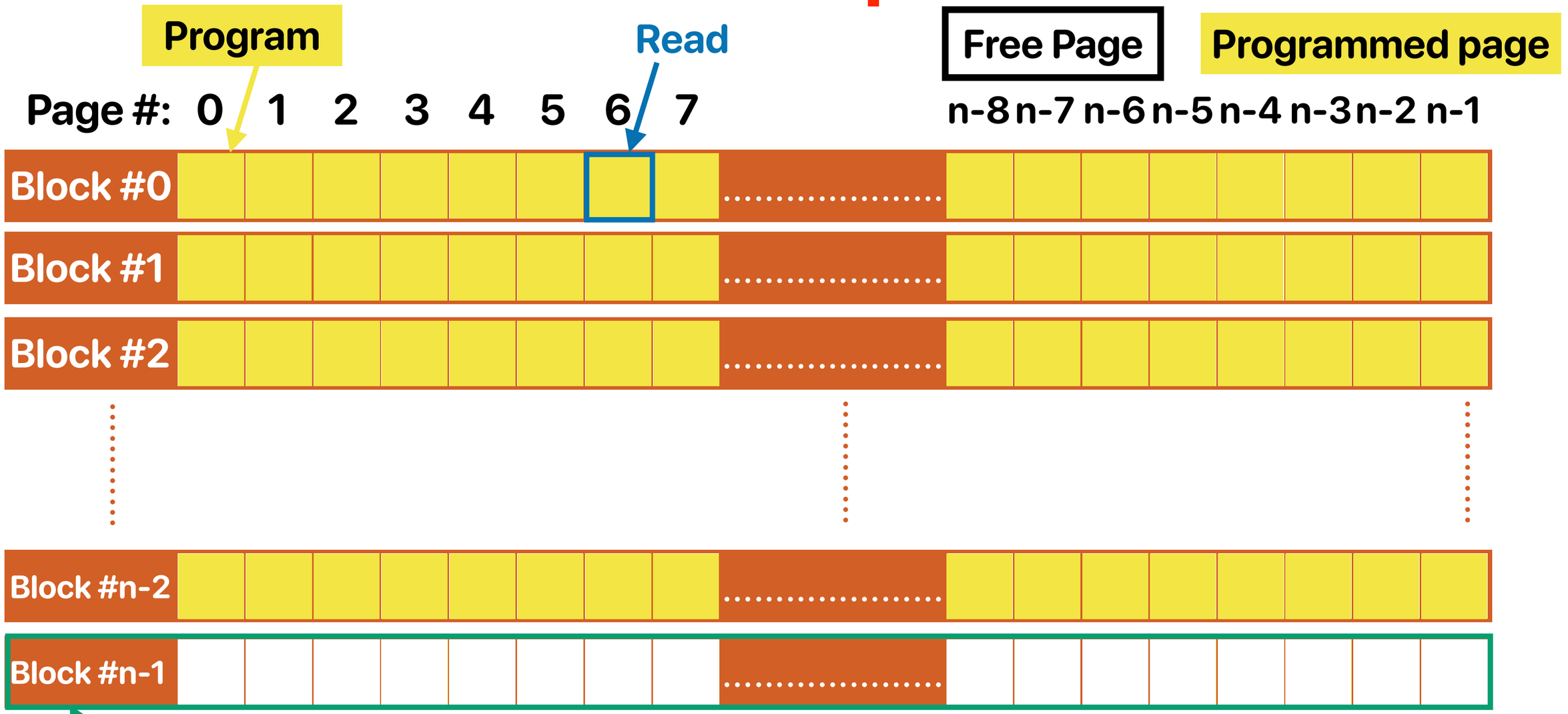
3:00

What’s the potential problem of writes in NAND-based SSDs?

Basic flash operations

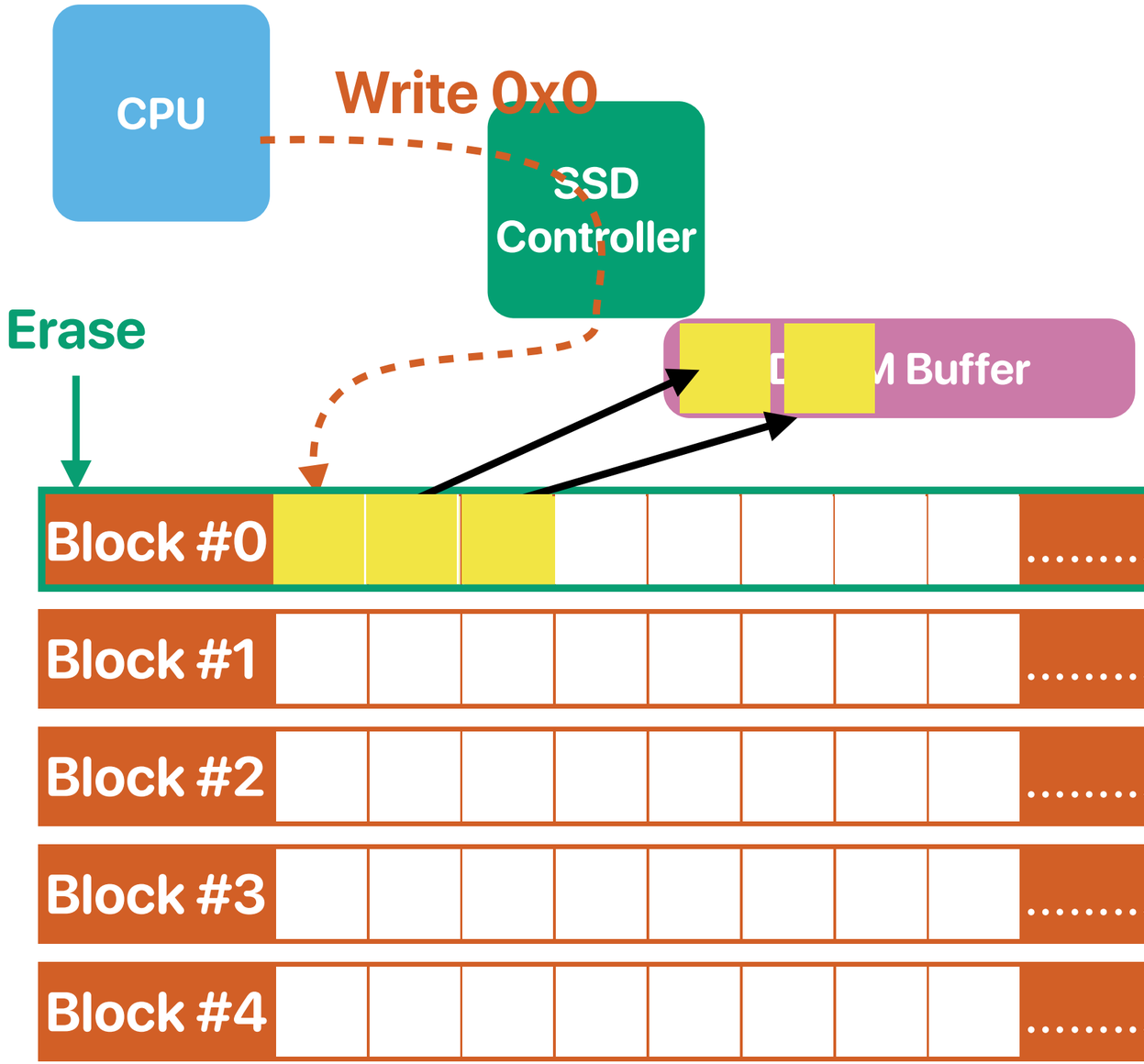


Basic flash operations



- Flash pages must be erased in "blocks"

What happens on a write if we use the same abstractions as H.D.D.



Can we write to page #0 directly? No.

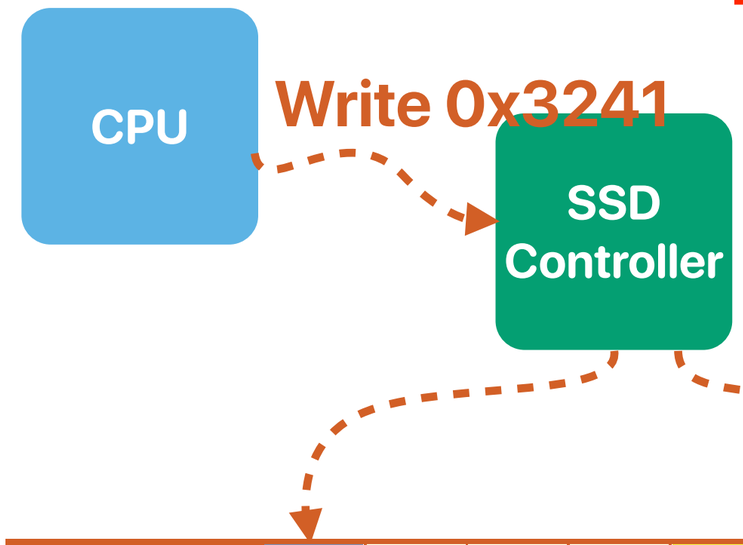
We have to copy page #1, page #2 in block #0 to somewhere (e.g. RAM buffer) and then erase the block

Write this the new 0 and the old 15 back to block #0 again!

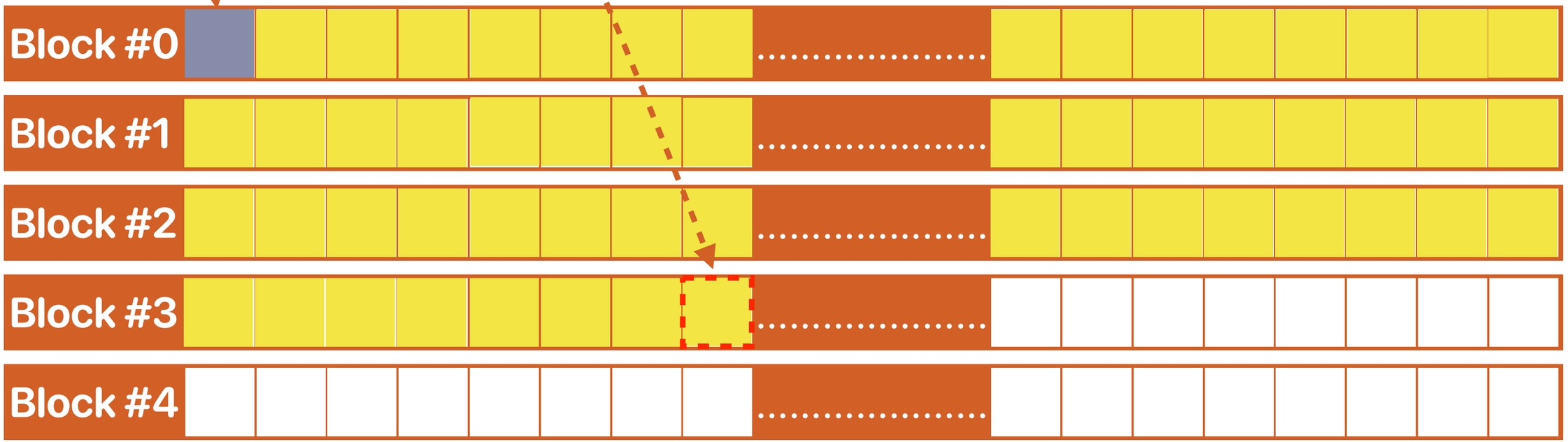
Read: 6*30us + Writing: 2ms*3 + Erasing 3ms ~ 9 ms

Not much faster than the H.D.D. — also hurts the lifetime

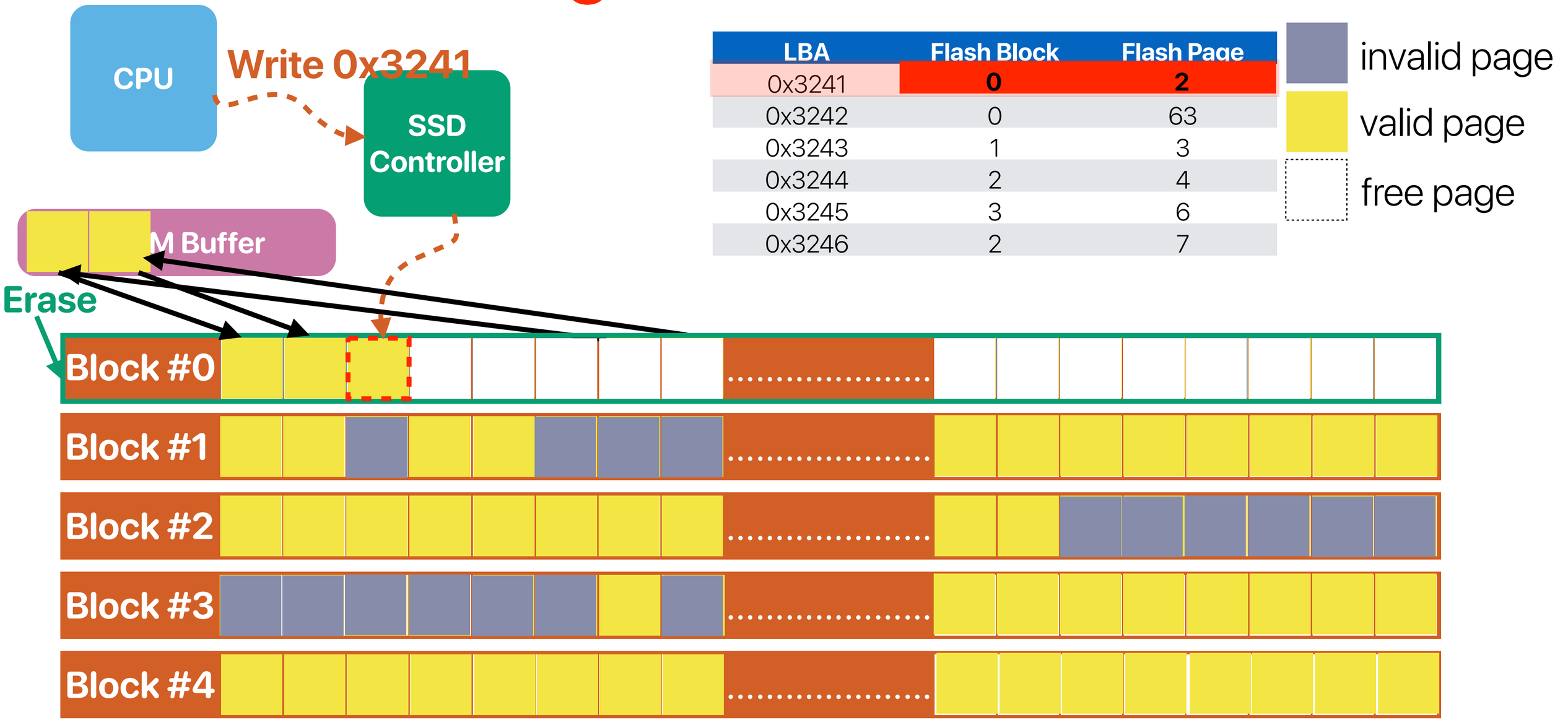
What happens on a write with FTL



LBA	Flash Block	Flash Page
0x3241	**3**	**7**
0x3242	0	63
0x3243	1	3
0x3244	2	4
0x3245	3	6
0x3246	2	7



Garbage Collection in FTL



LBA	Flash Block	Flash Page
0x3241	**0**	**2**
0x3242	0	63
0x3243	1	3
0x3244	2	4
0x3245	3	6
0x3246	2	7

invalid page

valid page

free page

The architecture of an SSD

Stores the mapping table

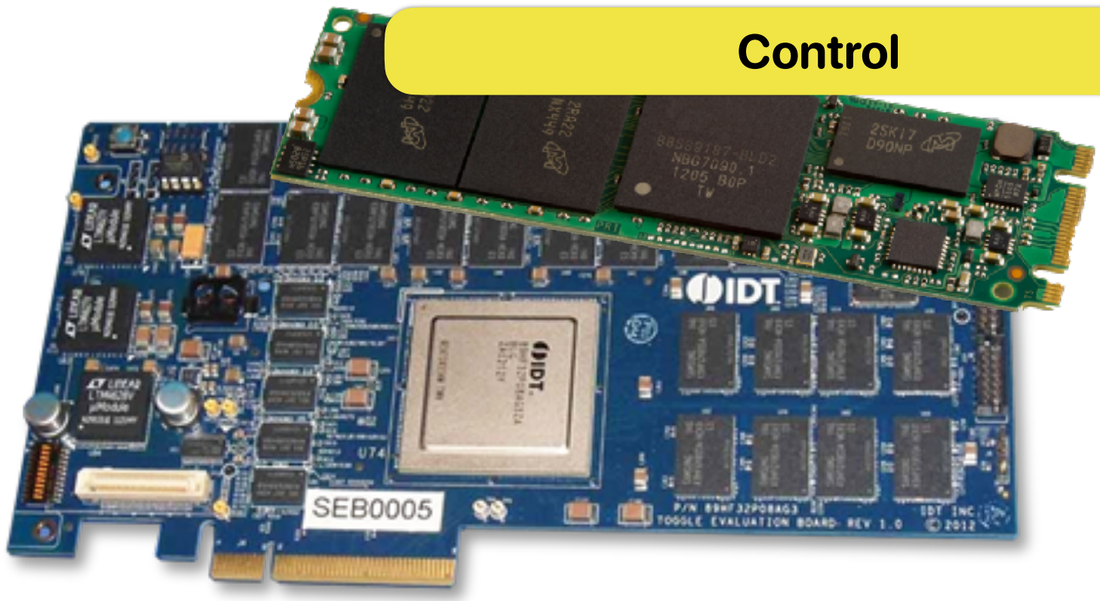
Controller + Registers

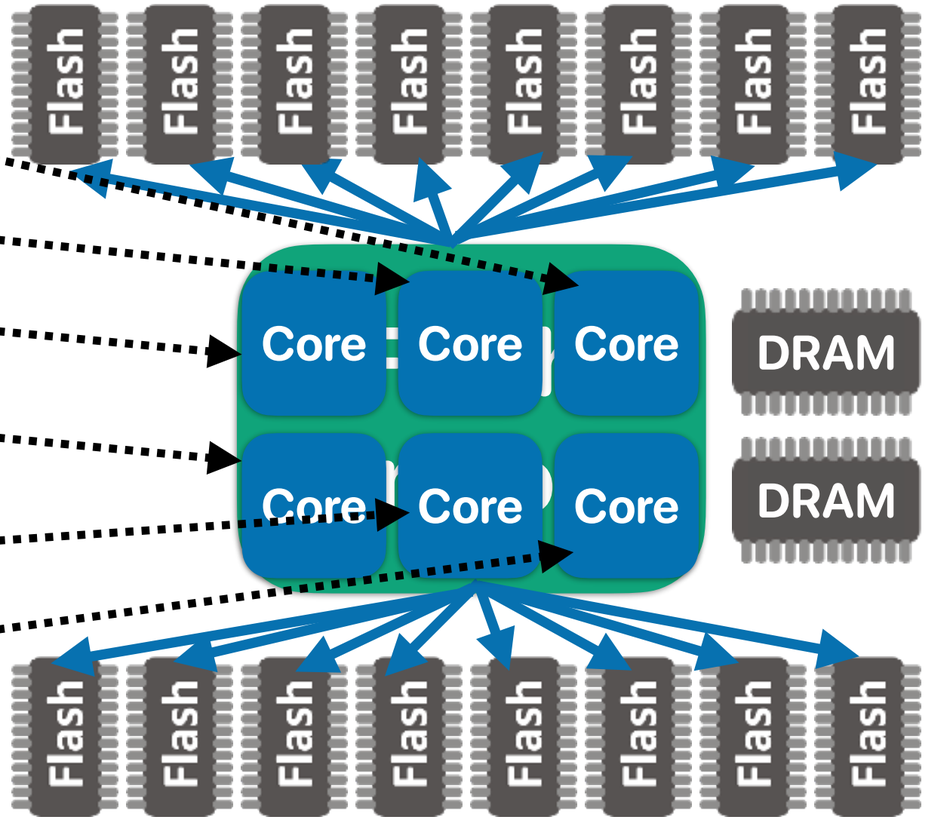
Perform FTL algorithms

- Maintaining the block device abstraction
- Dealing with the “weird” device characteristics
- Interfacing with the interconnect

The tasks of an SSD controller/firmware

- Parse NVMe/SATA/PCIe commands
- Address mapping/lookup
- Wear-leveling/Garbage Collection
- Read
- Program
- Control





Firmware is …

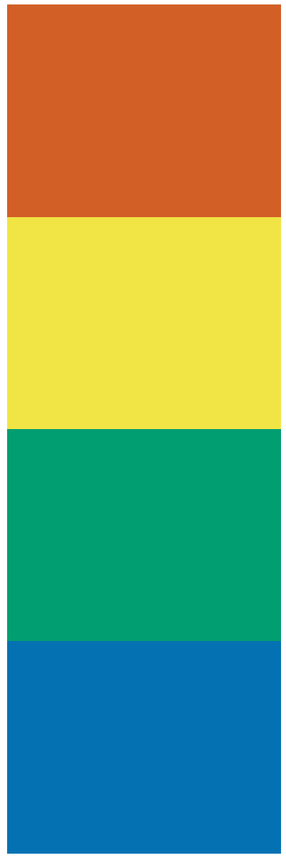
- A software program directly interacts with hardware without an operating system
- Typically a program works in between hardware and application “software”

Types of Flash Chips

2 voltage levels, 1-bit

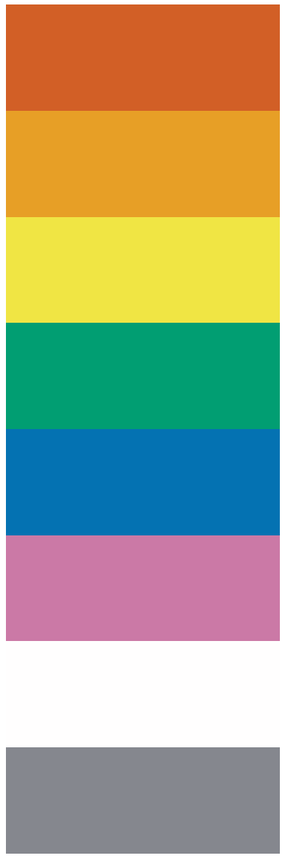
Single-Level Cell (SLC)

4 voltage levels, 2-bit



Multi-Level Cell (MLC)

8 voltage levels, 3-bit



Triple-Level Cell (TLC)

16 voltage levels, 4-bit

Quad-Level Cell (QLC)

Programming in MLC

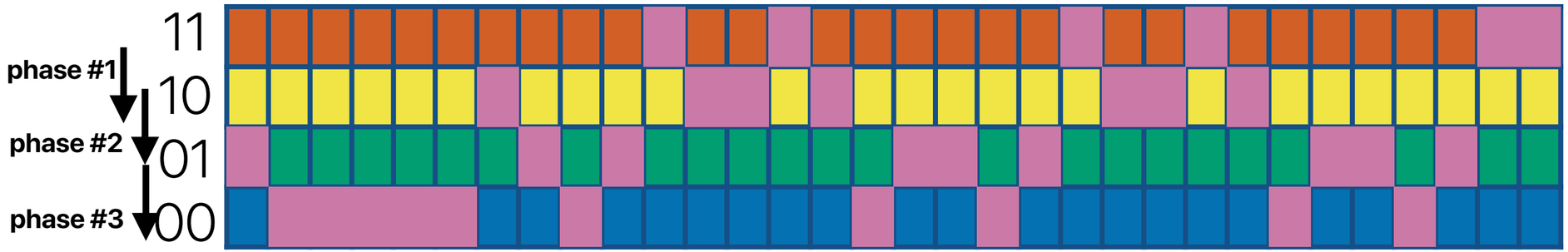
4 voltage levels,
2-bit

Multi-Level Cell (MLC)

3.14000000000000012434497875800

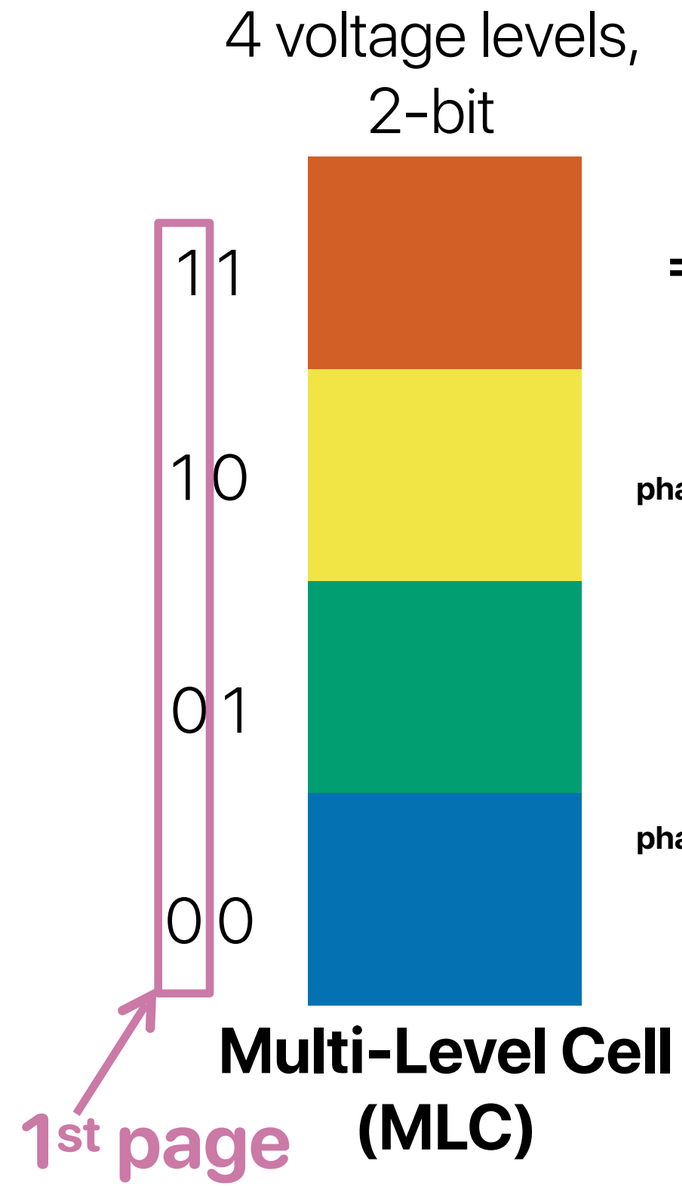
= 0x40091EB851EB851F

= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111



3 Cycles/Phases to finish programming

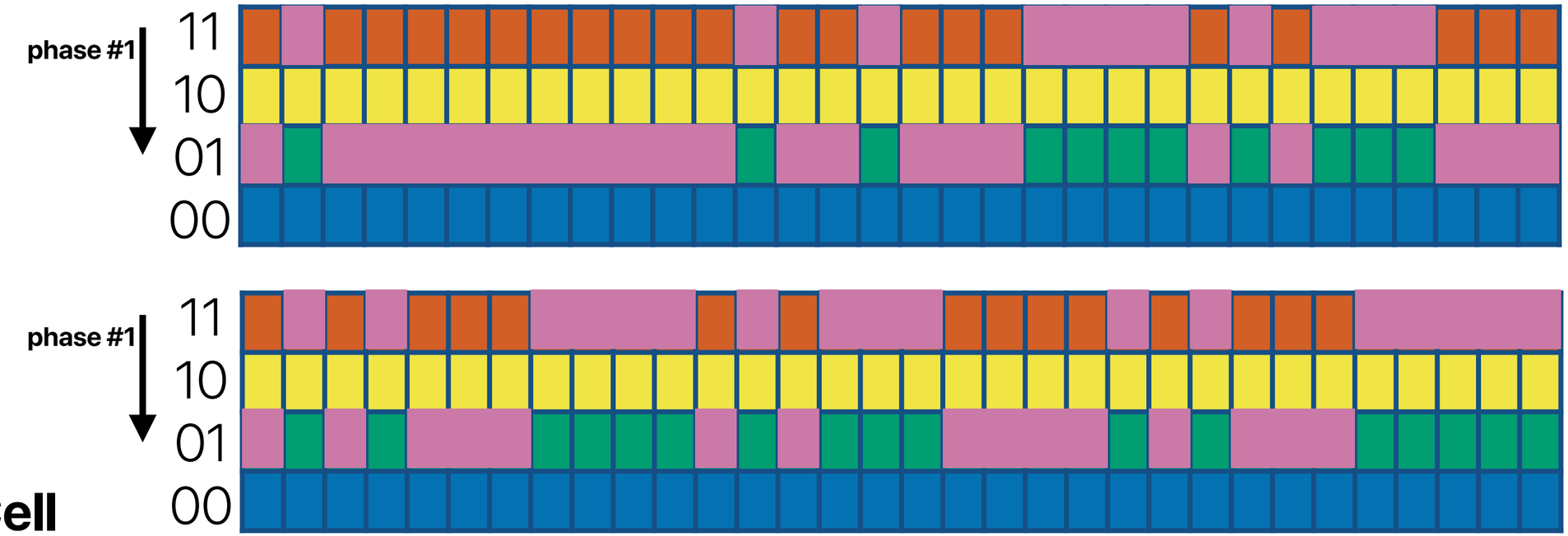
Programming in MLC



3.14000000000000012434497875800

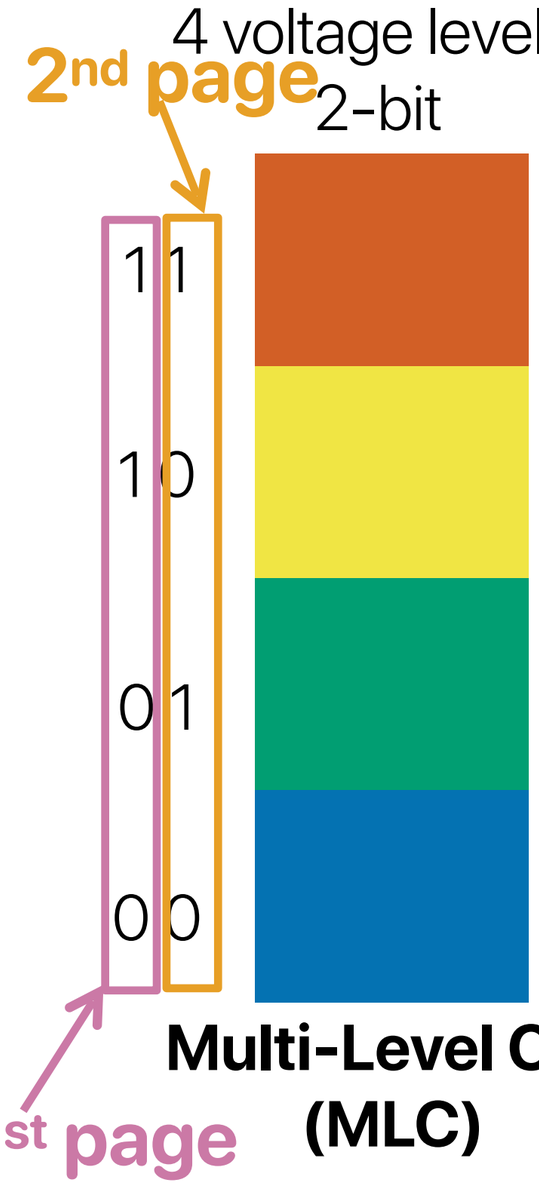
= 0x40091EB851EB851F

= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111



1 Phase to finish programming the first page!

Programming the 2nd page in MLC

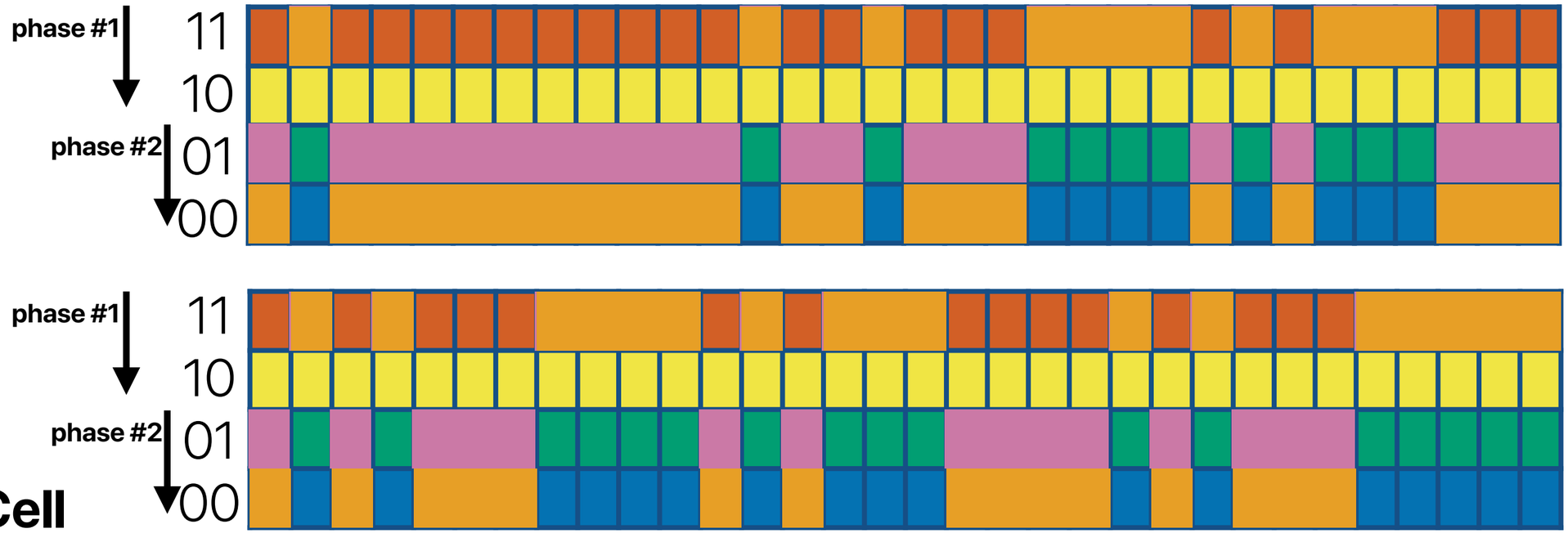


3.14000000000000012434497875800

= 0x40091EB851EB851F

= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111

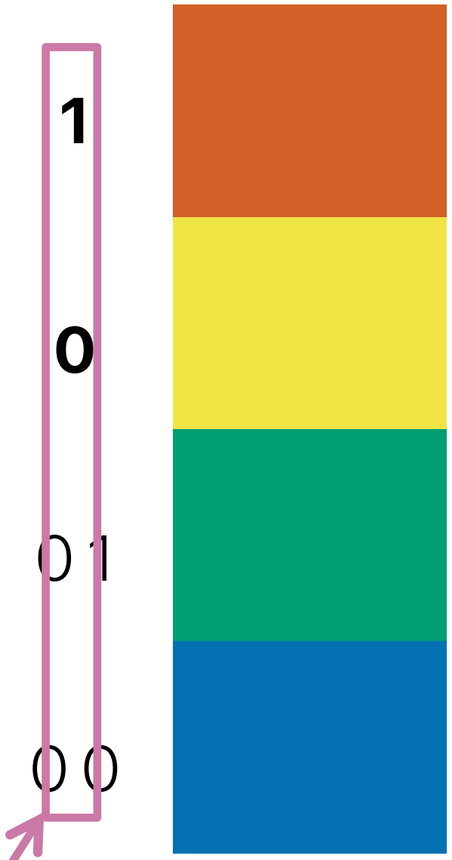
= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111



2 Phase to finish programming the second page!

Optimizing 1st Page Programming in MLC

4 voltage levels,
2-bit



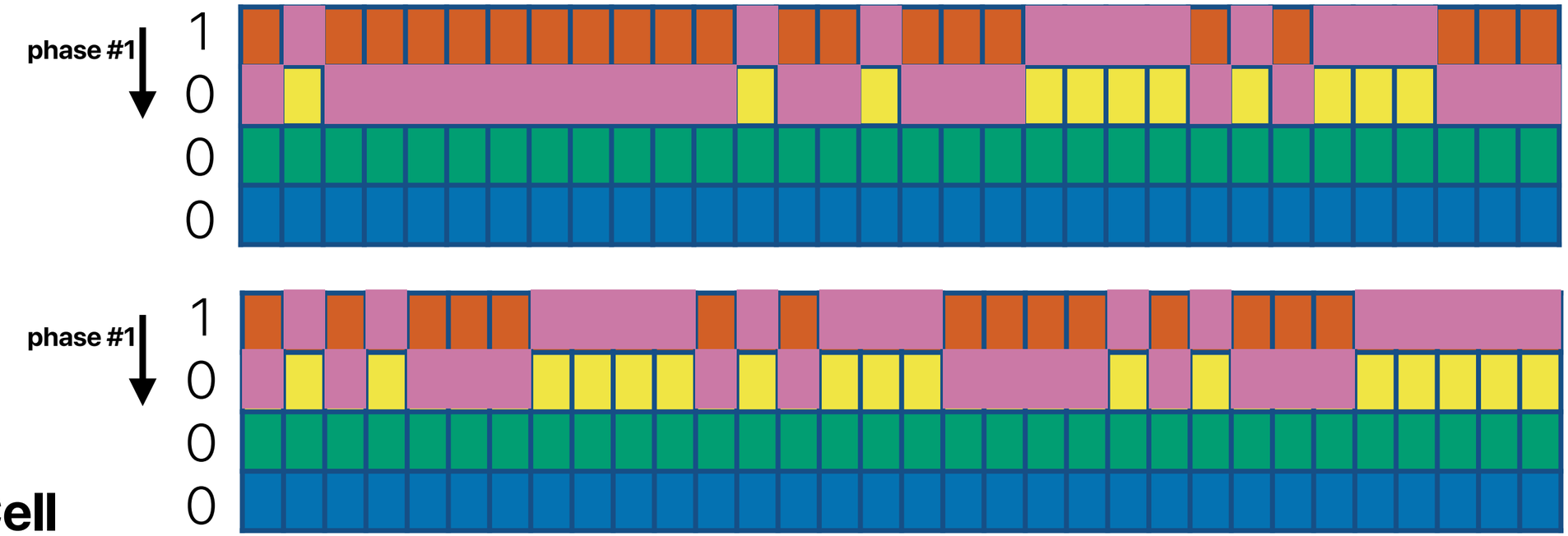
Multi-Level Cell (MLC)

1st page

3.140000000000001243449787580

= 0x40091EB851EB851F

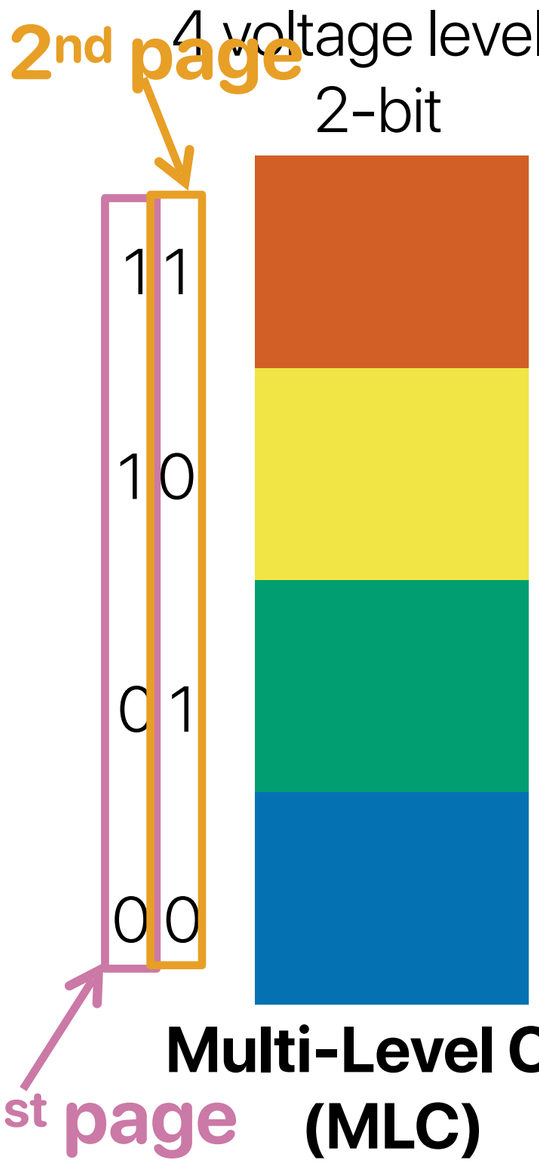
= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111



1 Phase to finish programming the first page!

— the phase is shorter now

2nd Page Programming in MLC

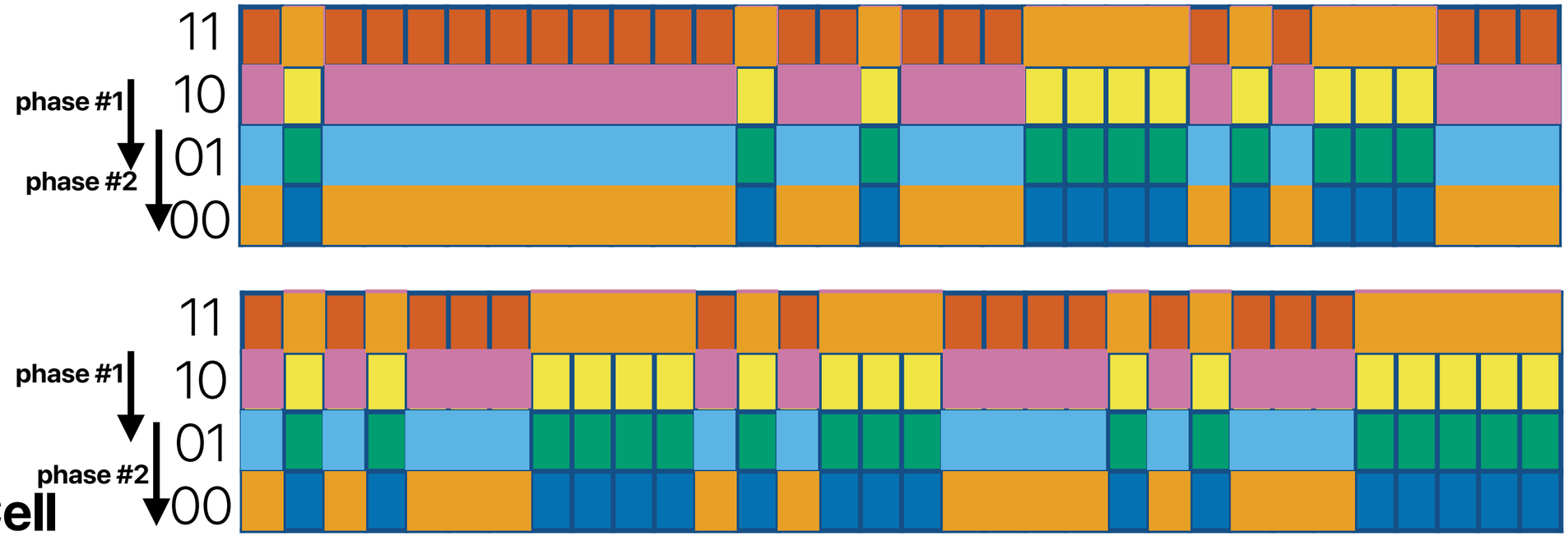


3.14000000000000012434497875800

= 0x40091EB851EB851F

= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111

= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111



2 Phase to finish programming the second page!

QLC = More Density Per NAND Cell

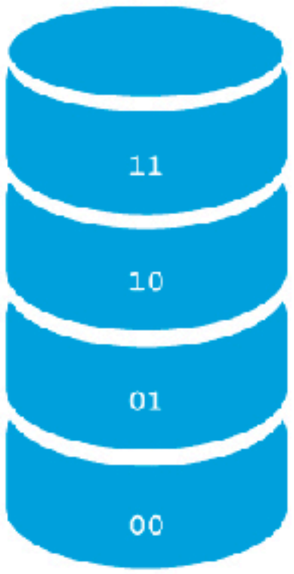
Lower $ per GB

SLC

1 Bit Per Cell

First SSD NAND technology

100K P/E Cycles
(at technology introduction)

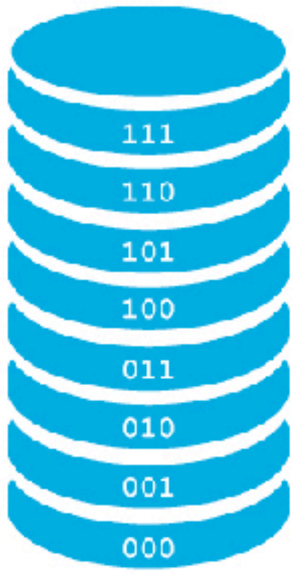


MLC

2 Bits Per Cell

100% increase

10K P/E Cycles

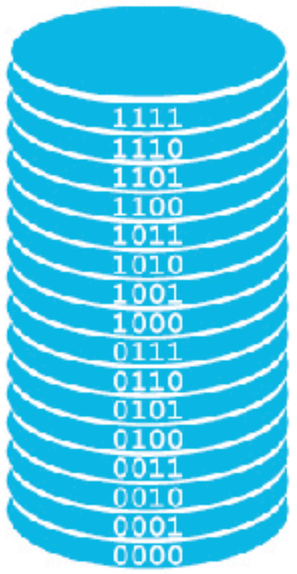


TLC

3 Bits Per Cell

50% increase

3K P/E Cycles



QLC

4 Bits Per Cell

33% increase

1K P/E Cycles

Fewer writes per cell

Micron

The bandwidth of a flash chip

- Organization
  - Page size x8: 18,592 bytes (16,384 + 2208 bytes)
  - Block size: 2304 pages, (36,864K + 4968K bytes)
  - Plane size: 4 planes x 504 blocks
  - Device size: 512Gb: 2016 blocks; 1Tb: 4032 blocks; 2Tb: 8064 blocks; 4Tb: 16,128 blocks; 8Tb: 32,256 blocks

is suspended or ongoing					
READ PAGE operation time without/with V_{PP}	^{t}R	57/41	60	µs	2, 12
SNAP READ operation time	$^{t}RSNAP$	27	37	µs	
Cache read busy time	$^{t}RCBSY$	11	60	µs	2, 8, 12

$$\frac{16KB}{60us} = 267MB/sec$$

Multi-channel to optimize bandwidth

Flashtec™ NVMe2032 and NVMe2016 Controllers

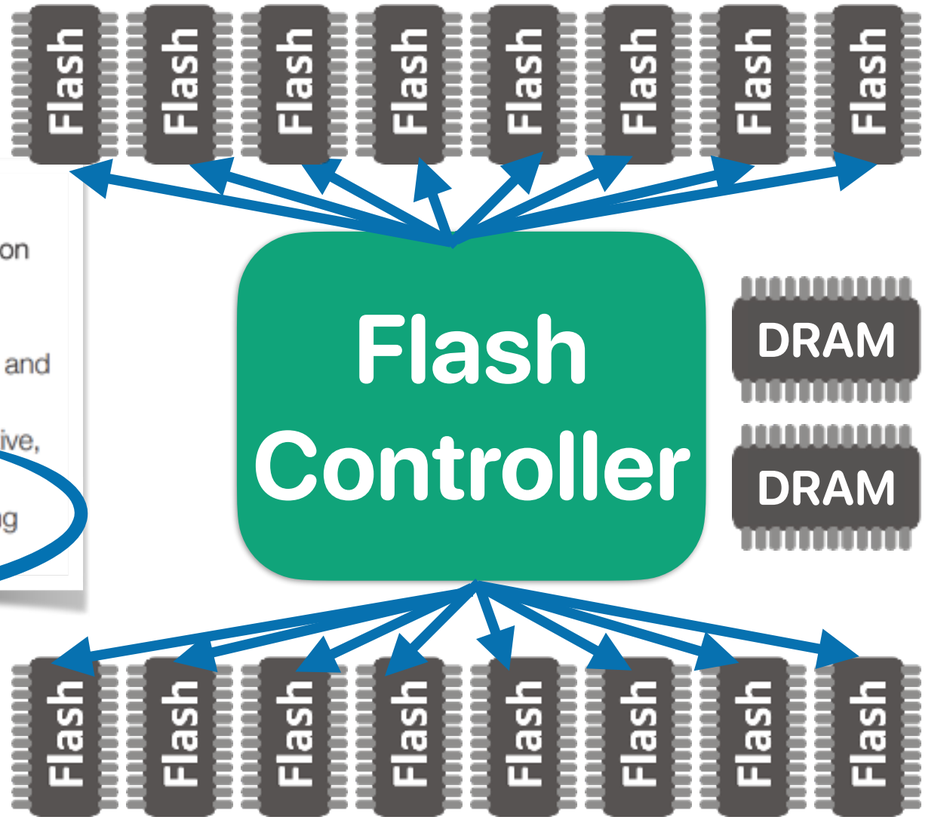
32- and 16-Channel PCIe Flash Controller Pr

Summary

The Flashtec™ 2nd generation NVMe Controller Family enables the
and data centers to realize the highest performance SSDs utilizing n
technologies. Combining world-class capacity and flexibility, the Flas
reliable choice. The Flashtec NVMe2032 and NVMe2016 controllers
Express (NVMe) host interface and are optimized for high-performan
operations, performing all flash management operations on-chip an
processing and memory resources.

Features

- Flashtec NVMe2032 controller can achieve up to 1 million random read IOPS on 4 KB operations
- Up to 20 TB Flash capacity using 256 GB Flash
- SLC, MLC, Enterprise MLC, and TLC Flash with toggle and ONFI interface
- PCIe Gen 3 x8 or dual independent PCIe Gen 3 x4 (active, active/standby) host interface
- 16 and 32 independent Flash channels, each supporting up to 8 CE



Each ~267MB/sec
16 channels == 4.2 GB/sec

Micron

- Organization
  - Page size x8: 18,592 bytes (16,384 + 2208 bytes)
  - Block size: 2304 pages, (36,864K + 4968K bytes)
  - Plane size: 4 planes x 504 blocks
  - Device size: 512Gb: 2016 blocks; 1Tb: 4032 blocks; 2Tb: 8064 blocks; 4Tb: 16,128 blocks; 8Tb: 32,256 blocks

TLC 512Gb-8Tb NAND – Array Characteristics

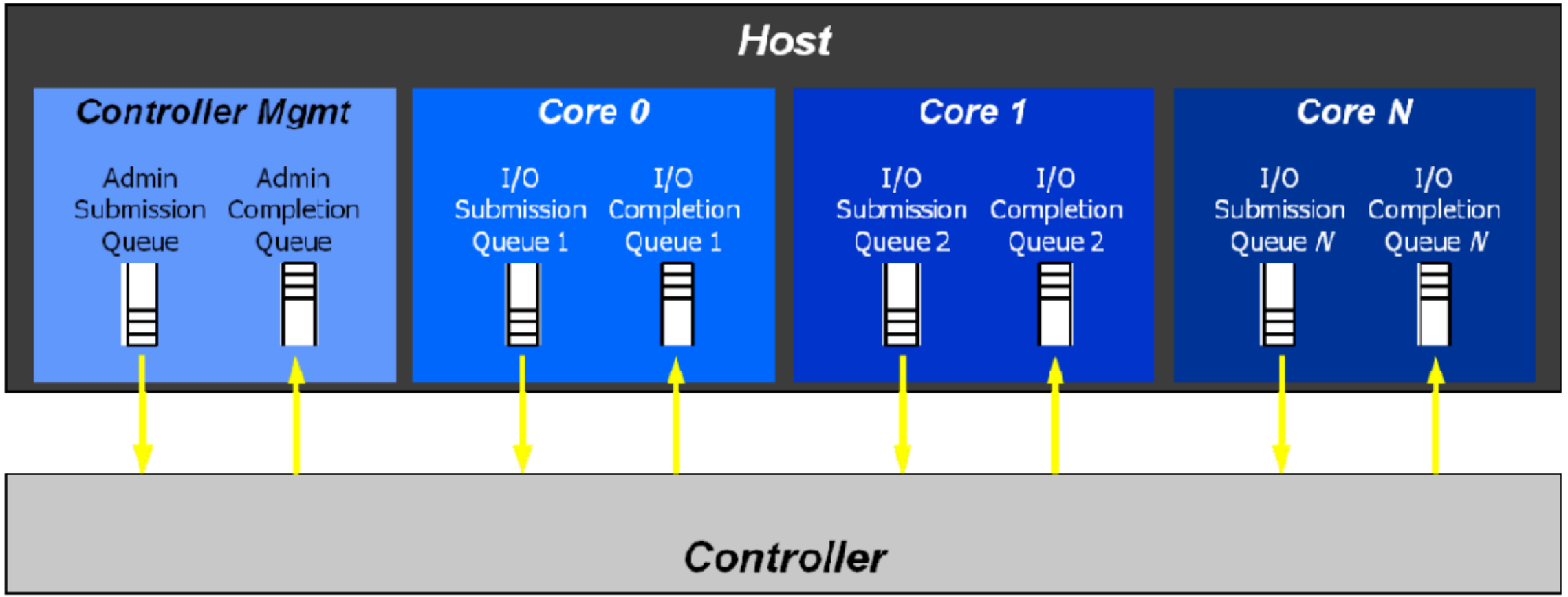
is suspended or ongoing					
READ PAGE operation time without/with V_{PP}	^{t}R	57/41	60	µs	2, 12
SNAP READ operation time	$^{t}RSNAP$	27	37	µs	
Cache read busy time	$^{t}RCBSY$	11	60	µs	2, 8, 12

3:00

What can you do to take advantage of multiple access channels in SSDs?

NVMe Model

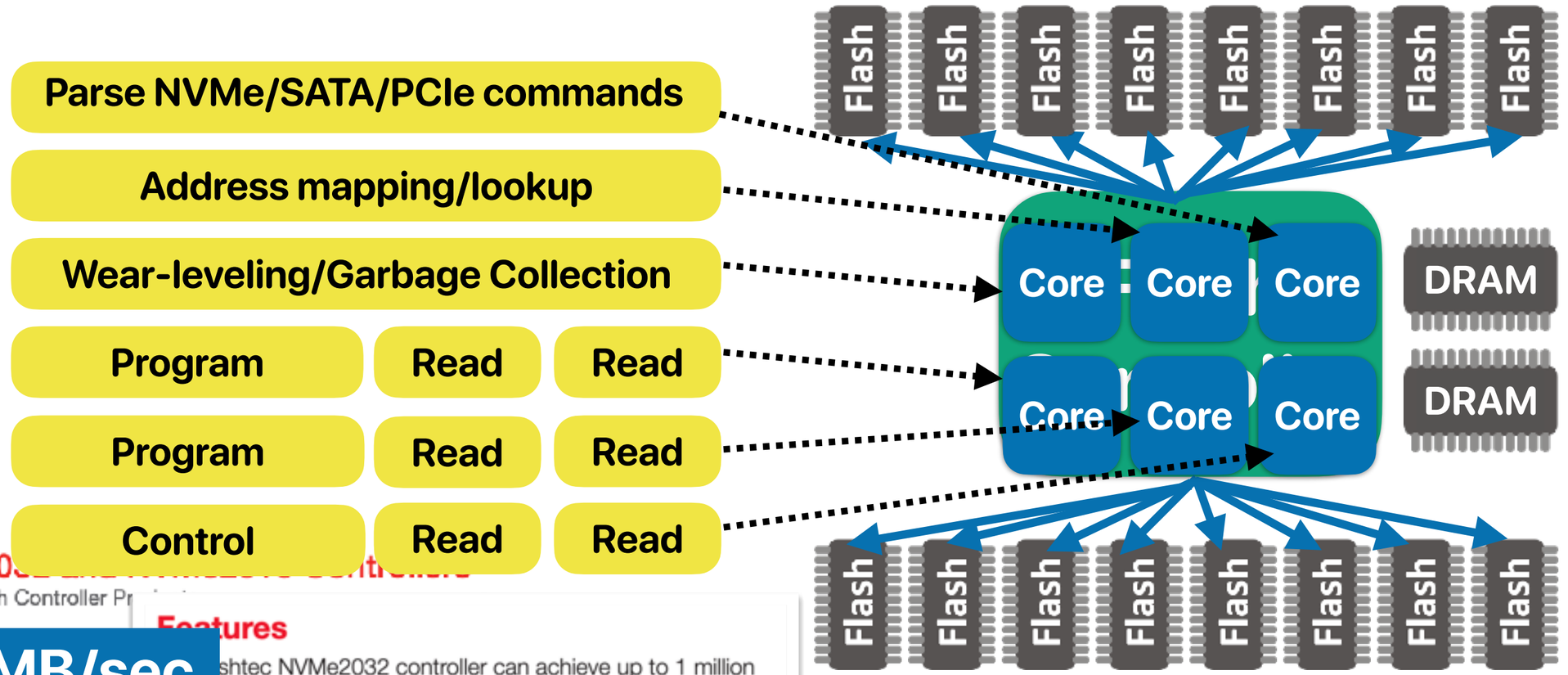
Figure 3: Queue Pair Example, 1:1 Mapping



https://nvmexpress.org/wp-content/uploads/2013/04/NVM_whitepaper.pdf

Why does MQ make sense?

Multi-channel to optimize bandwidth



Each ~267MB/sec
16 channels ==
4.2GB/sec

Flashtec™ NVMe20

32- and 16-Channel PCIe Flash Controller Pr

Features

shtec NVMe2032 controller can achieve up to 1 million
dom read IOPS on 4 KB operations
to 20 TB Flash capacity using 256 GB Flash
C, MLC, Enterprise MLC, and TLC Flash with to
FI interface
le Gen 3 x8 or dual independent PCIe Gen 3 x4 (active,
tive/standby) host interface
- 16 and 32 independent Flash channels, each suppo
up to 8 CE

Micron

- Organization
  - Page size x8: 18,592 bytes (16,384 + 2208 bytes)
  - Block size: 2304 pages, (36,864K + 4968K bytes)
  - Plane size: 4 planes x 504 blocks
  - Device size: 512Gb: 2016 blocks; 1Tb: 4032 blocks; 2Tb: 8064 blocks; 4Tb: 16,128 blocks; 8Tb: 32,256 blocks

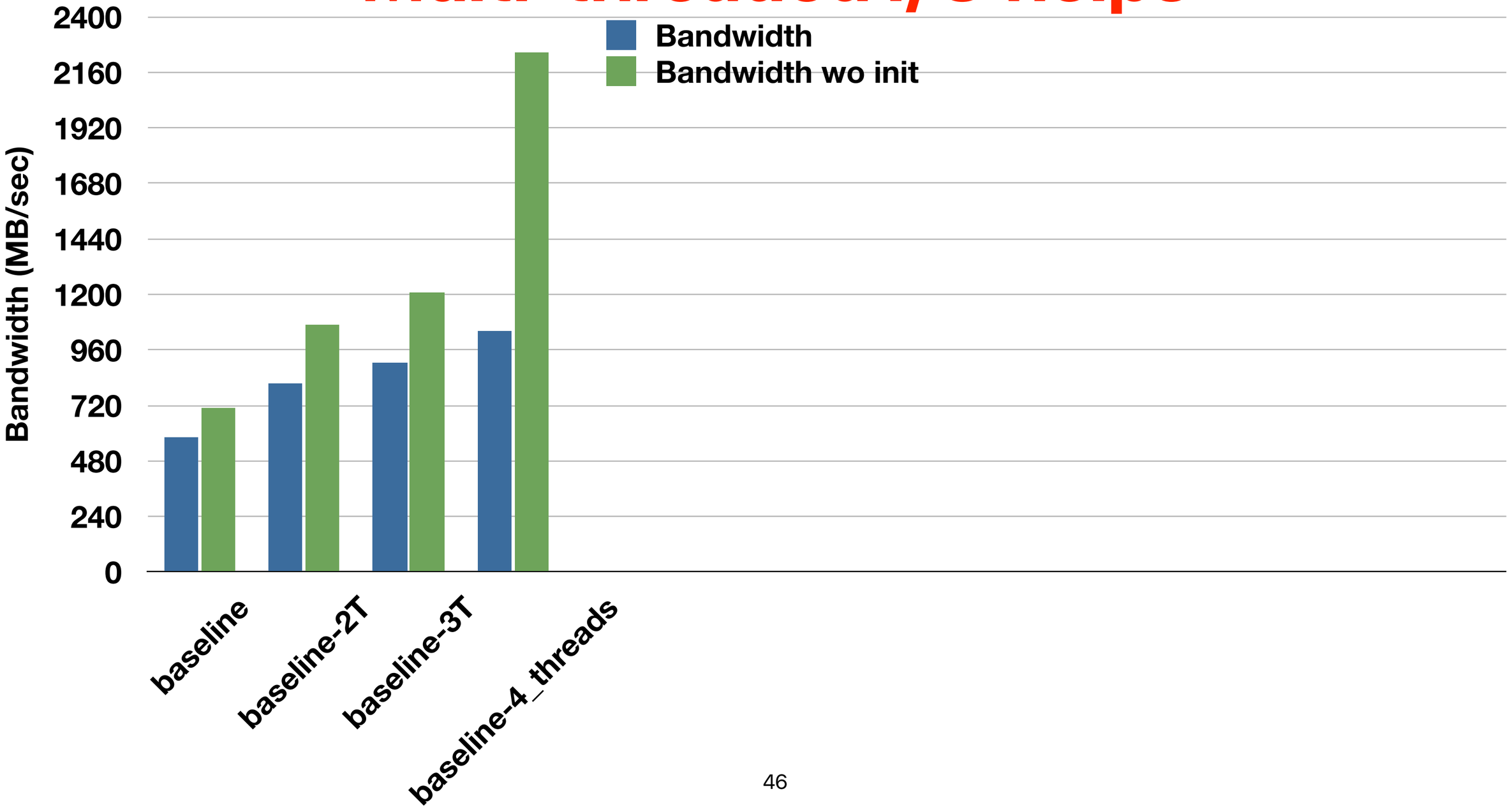
TLC 512Gb-8Tb NAND
– Array Characteristics

is suspended or ongoing					
REA PAGE operation time without/with V_{PP}	^{t}R	57/41	60	µs	2, 12
SNAP READ operation time	$^{t}RSNAP$	27	37	µs	
ache read busy time	$^{t}RCBSY$	11	60	µs	2, 8, 12

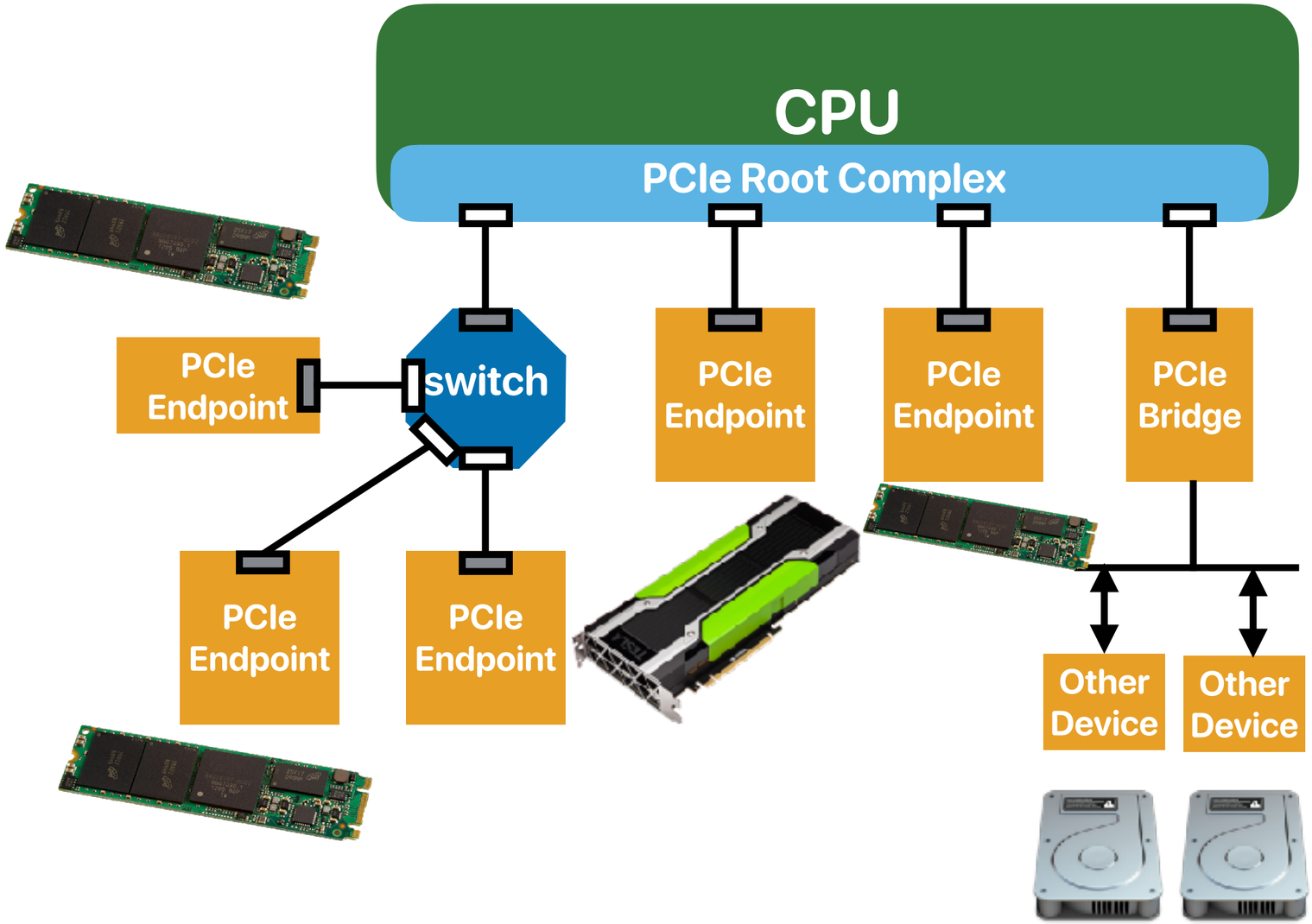
Why does MQ makes sense?

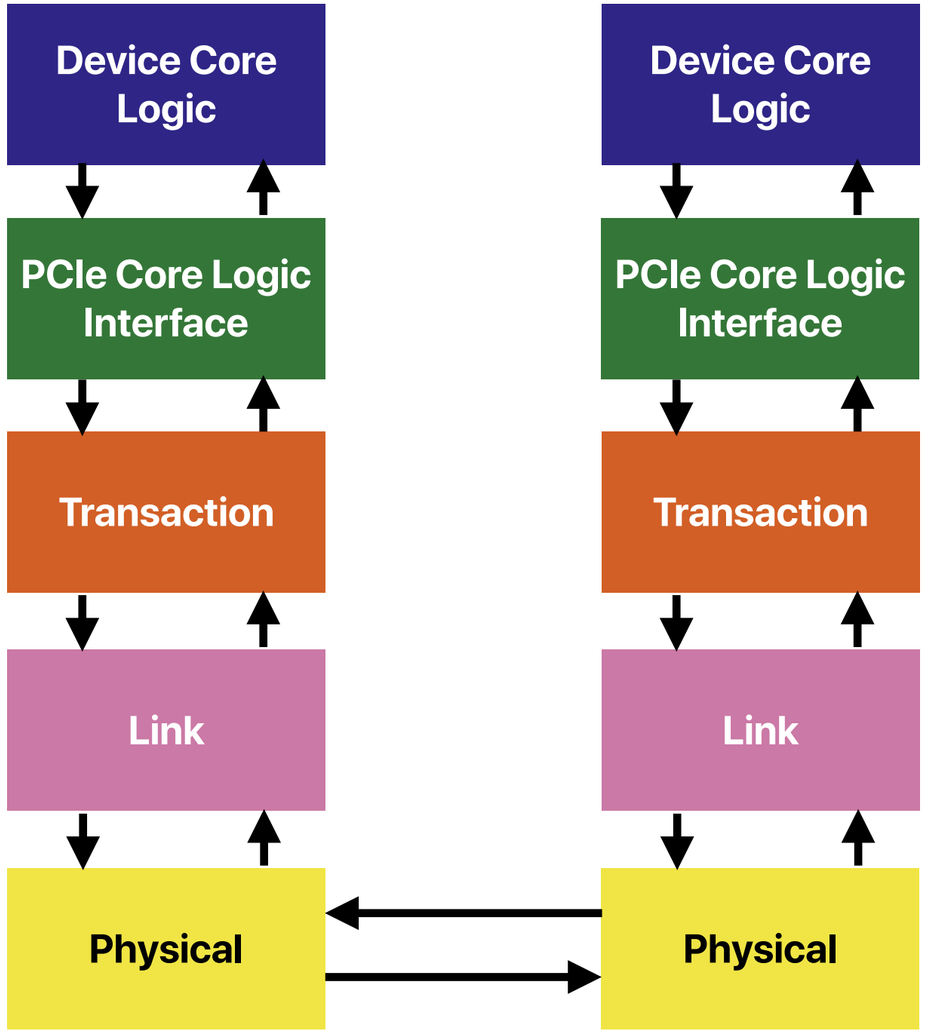
- Flash SSDs have multiple channels inside!
  - You need multiple requests to fill the requests
- The same design cannot work on HDDs — HDDs have only one head

Multi-threaded I/O helps



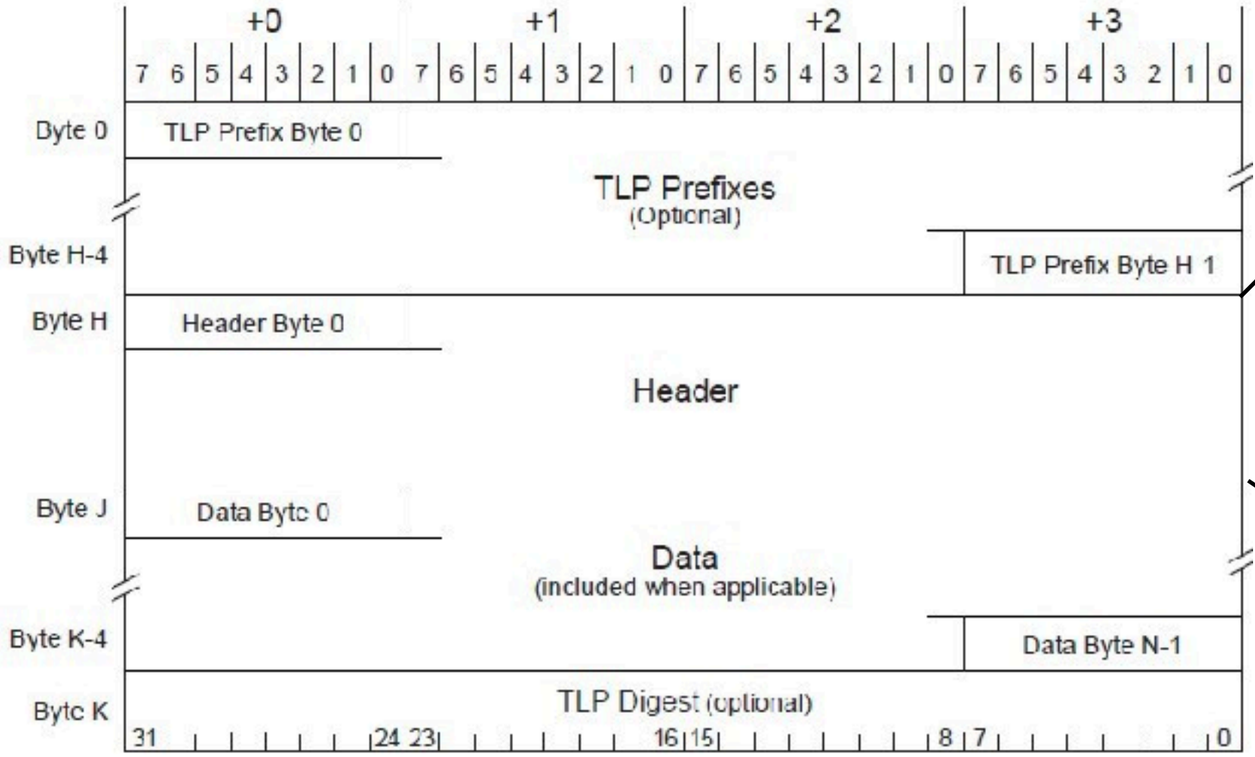
PCIe "Interconnect"

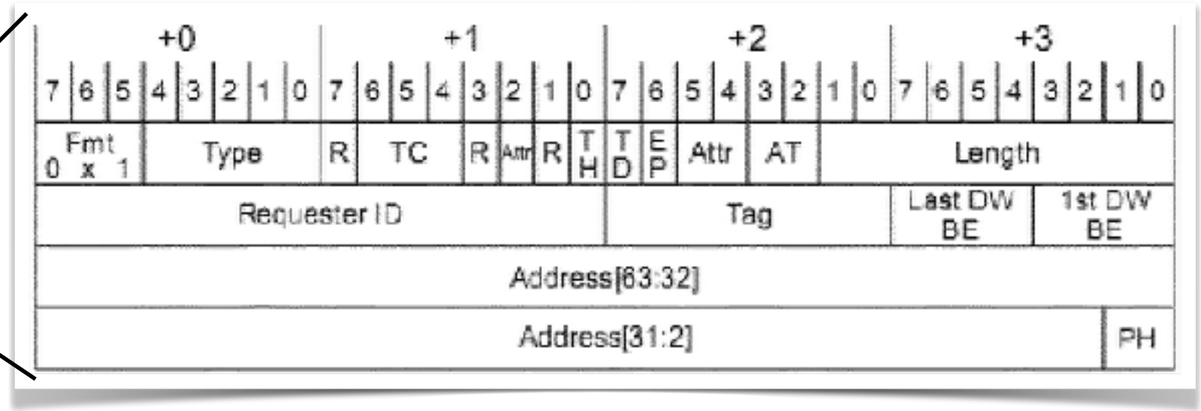




PCIe interconnect

- Very similar to computer networks
- Use “memory addresses” as the identifier for routing
- Peer-to-peer communication is possible





NVMe

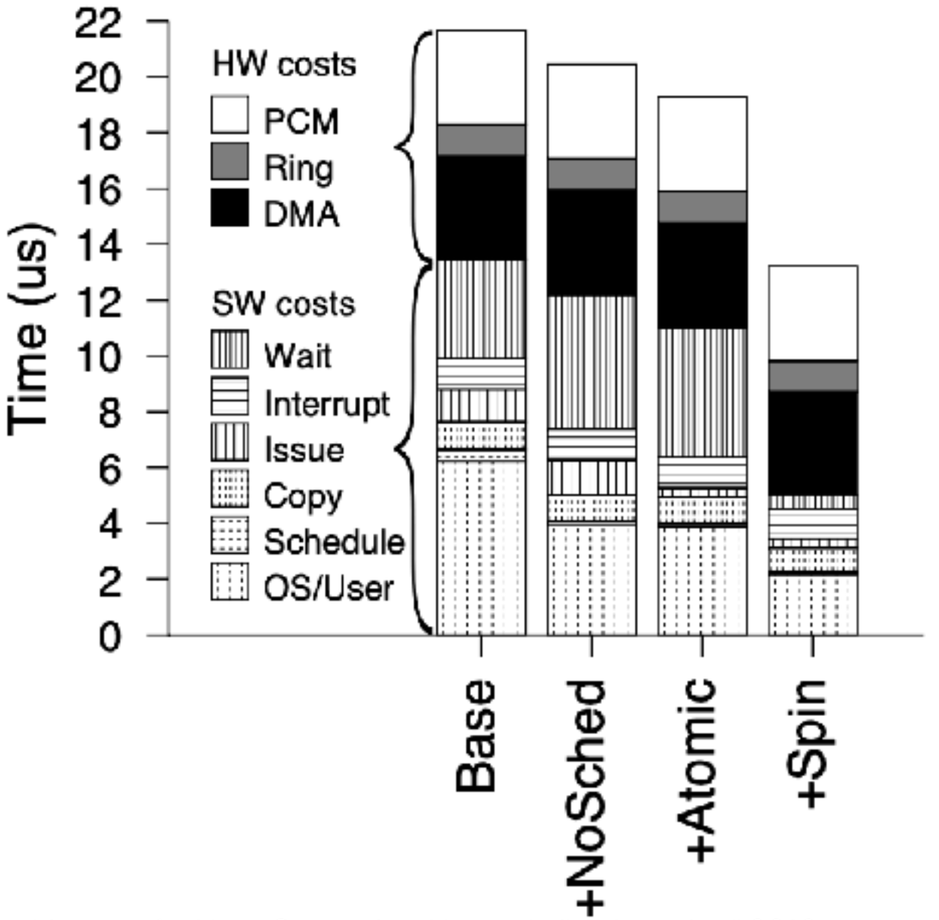
- The standard of PCIe SSD devices now
  - Provides multiple command queues to better support multithreading hardware
  - Allows more parallelism inside the SSD
- The “payload” of a PCIe packet

0	8	16	32	48	63
OPCODE	**FLAGS**	**command id**	**Namespace ID**		
reserved					
metadata					
PRP1					
PRP2					
Start LBA					
length		**control**	**Dataset management**		
Reference tag			**App tag**	**App mask**	

Let’s walk through an NVMe driver

Software overhead

Label	Description	Baseline latency (μs) Write	Read
OS/User	OS and userspace overhead	1.98	1.95
OS/User	Linux block queue and no-op scheduler	2.51	3.74
Schedule	Get request from queue and assign tag	0.44	0.51
Copy	Data copy into DMA buffer	0.24/KB	-
Issue	PIO command writes to Moneta	1.18	1.15
DMA	DMA from host to Moneta buffer	0.93/KB	-
Ring	Data from Moneta buffer to mem ctrl	0.28/KB	-
PCM	4 KB PCM memory access	4.39	5.18
Ring	Data from mem ctrl to Moneta buffer	-	0.43/KB
DMA	DMA from Moneta buffer to host	-	0.65/KB
Wait	Thread sleep during hw	11.8	12.3
Interrupt	Driver interrupt handler	1.10	1.08
Copy	Data copy from DMA buffer	-	0.27/KB
OS/User	OS return and userspace overhead	1.98	1.95
Hardware total for 4 KB (accounting for overlap)		8.2	8.0
Software total for 4 KB (accounting for overlap)		13.3	12.2
File system additional overhead		5.8	4.2



A. M. Caulfield, A. De, J. Coburn, T. I. Mollow, R. K. Gupta and S. Swanson, "Moneta: A High-Performance Storage Array Architecture for Next-Generation, Non-volatile Memories," 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture, 2010, pp. 385-395, doi: 10.1109/MICRO.2010.33.

Electrical
Computer Science
Engineering

277

つづく