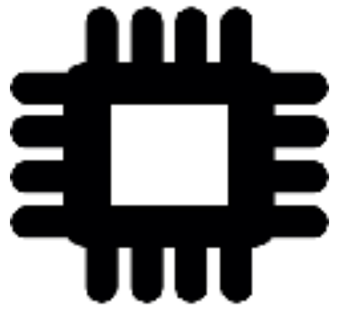
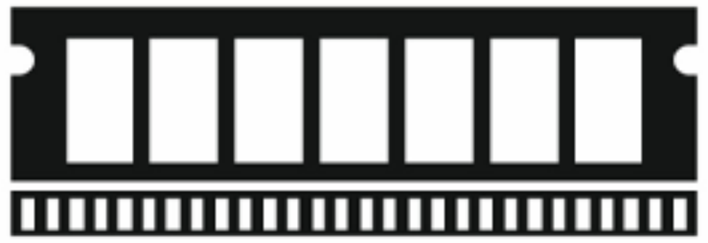
Storage

Database System Technology

The memory Hierarchy



CPU caches



memory

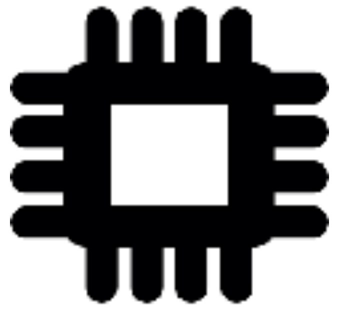
SSD

disk

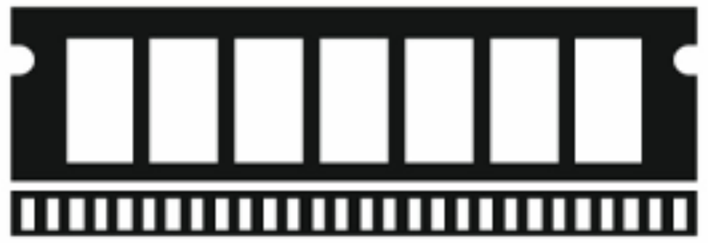
Expensive & fast

Slow & cheap

Volatility - Does data stay when power is off?



CPU caches



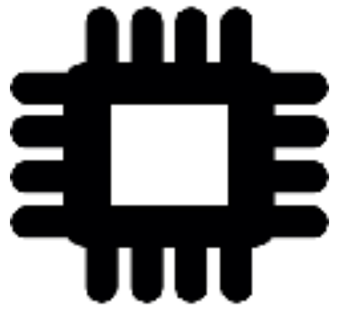
memory

SSD

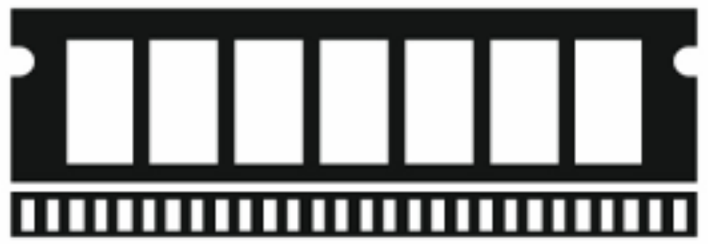
disk

Data disappears

Data persists



CPU caches



memory

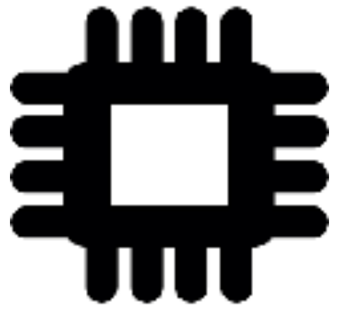
SSD

disk

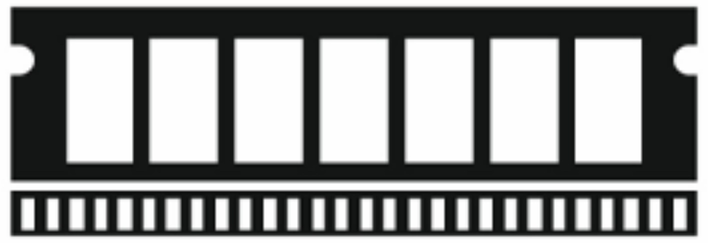
Data is brought here for processing

Data resides here

Access granularity



CPU caches



memory

Byte-addressable
64-128 B

SSD

disk

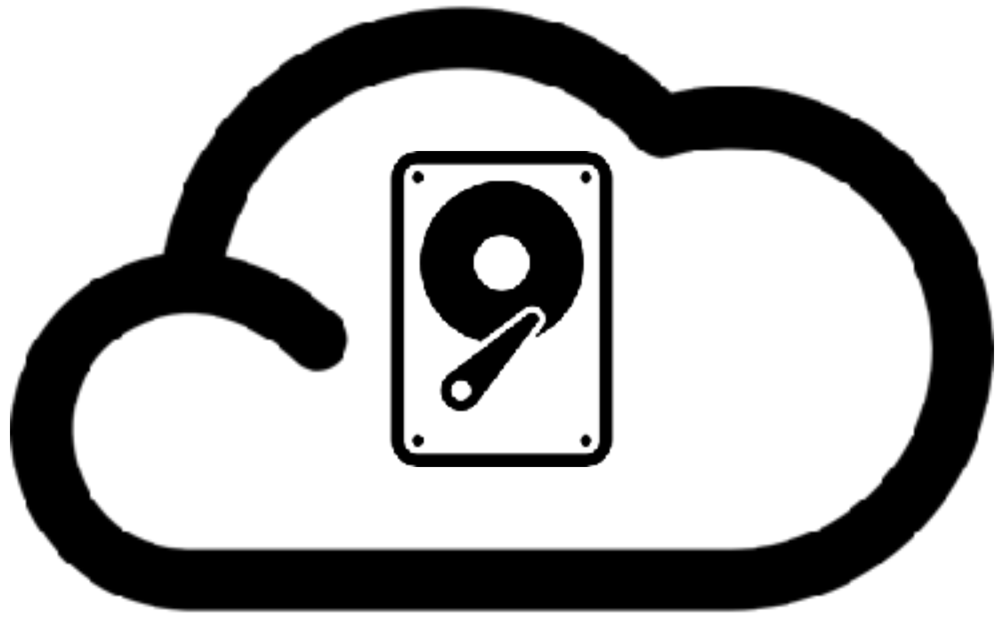
Block-addressable
4-16 KB

disk

Rarely used in laptops or PCs these days

disk

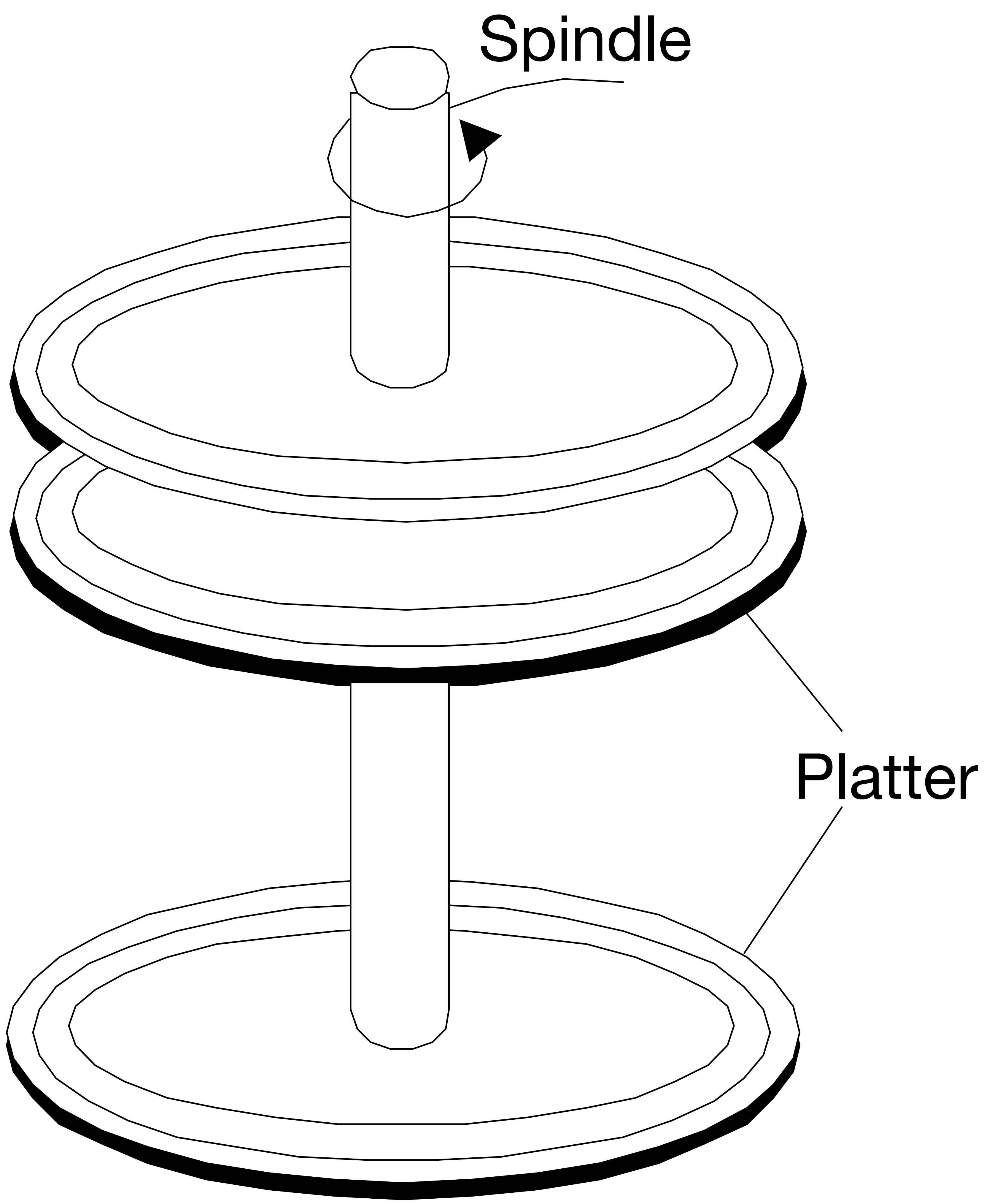
Rarely used in laptops or PCs these days



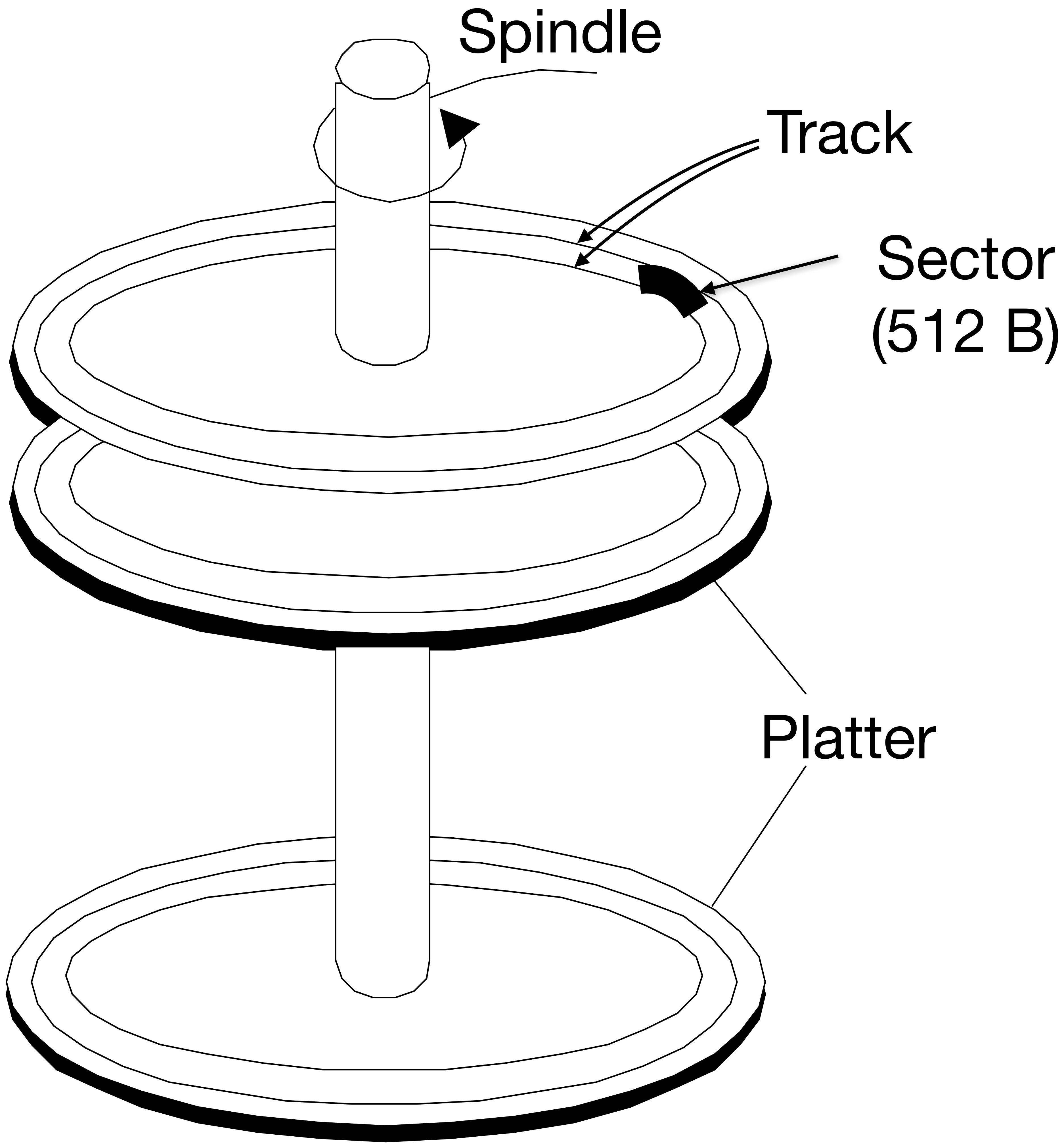
Used in data centers

disk

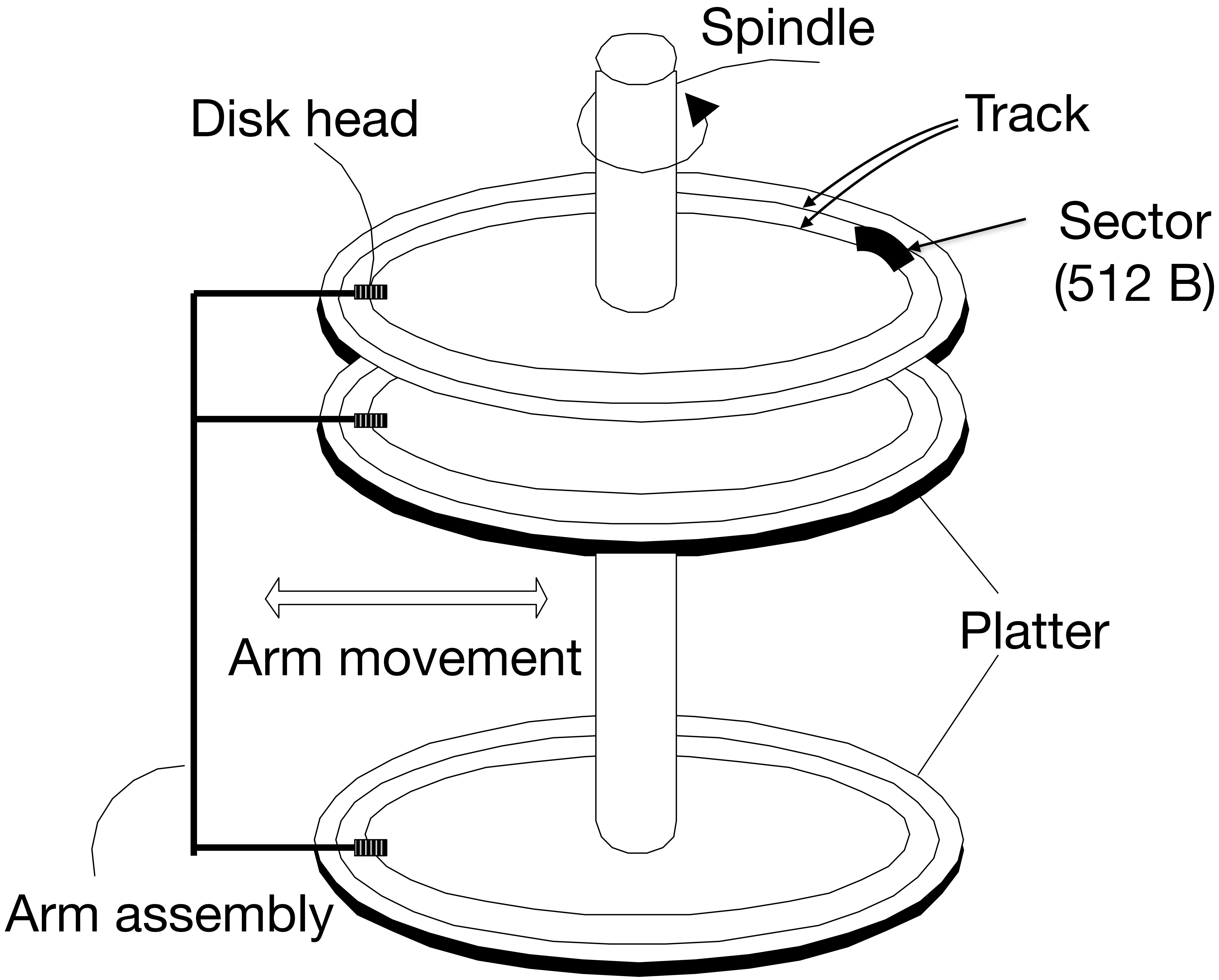
disk



disk



disk


Spindle
Disk head
Track
Sector
(512 B)
Arm movement
Platter
Arm assembly

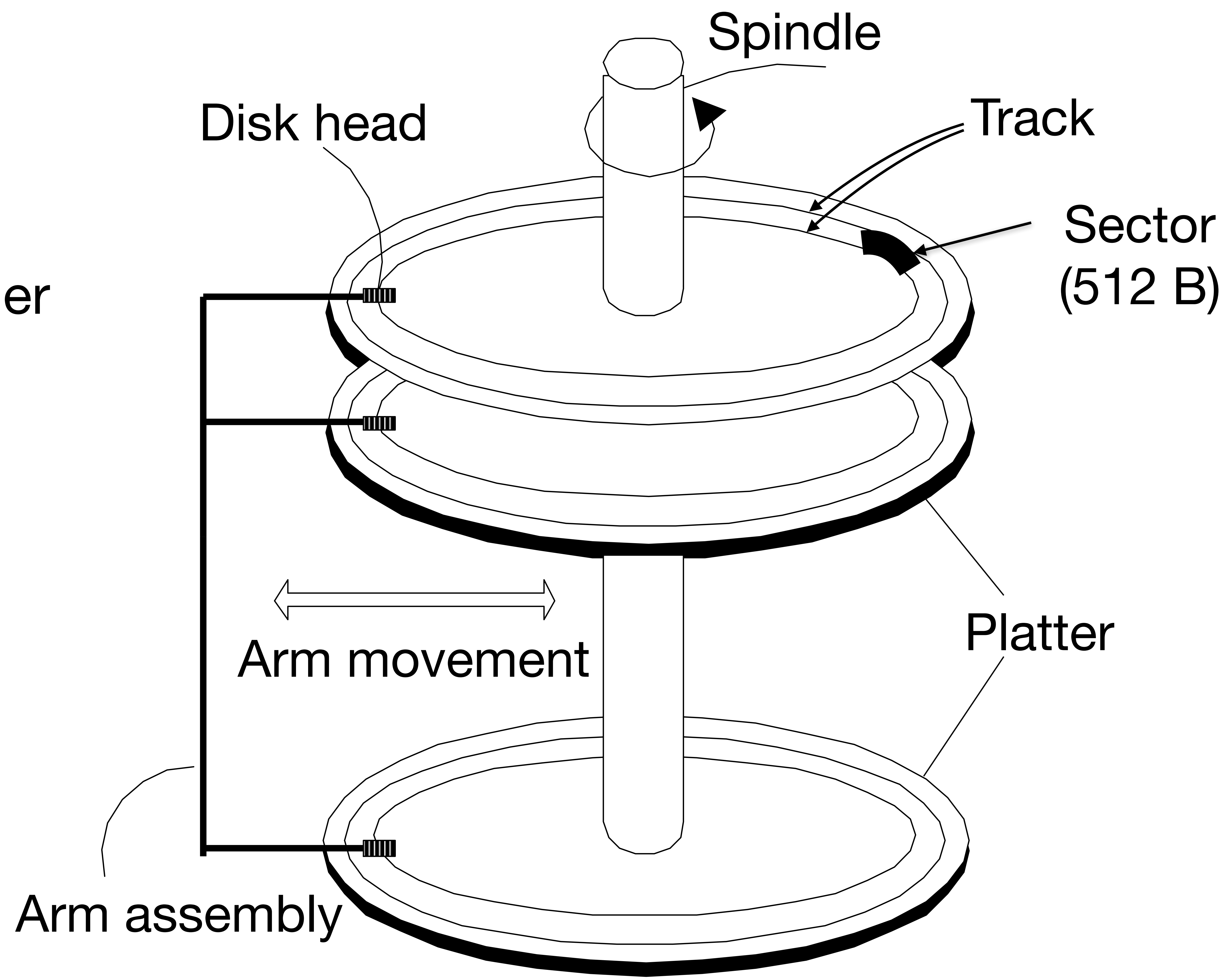
disk: access times

seek time - Move arms to position head on track (1-10 ms)

rotational delay - wait for sector to rotate under head (0-5 ms)

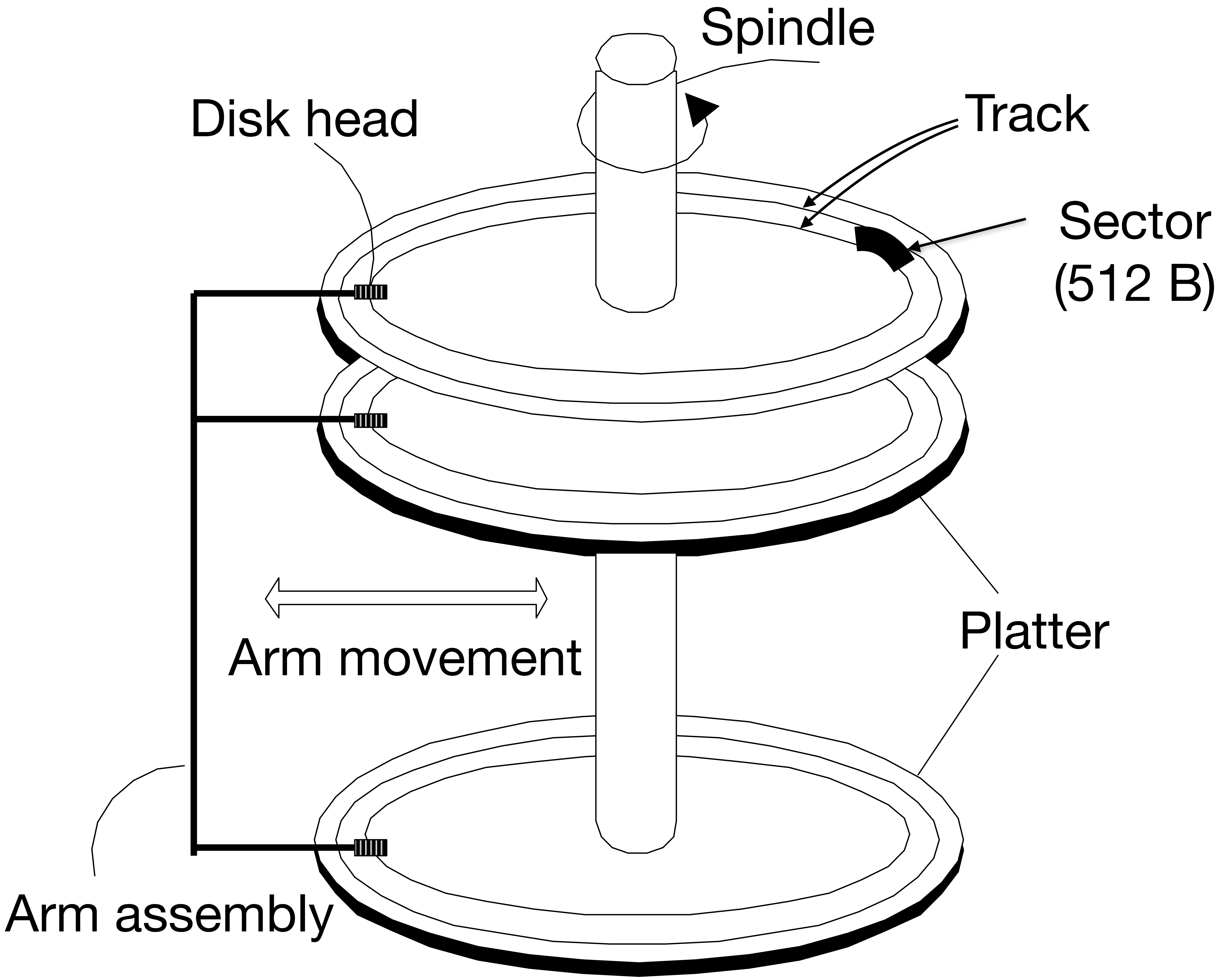
transfer time - moving data to/from disk surface
(> 0.01 ms)

seek/rotational delays dominate



disk access principles

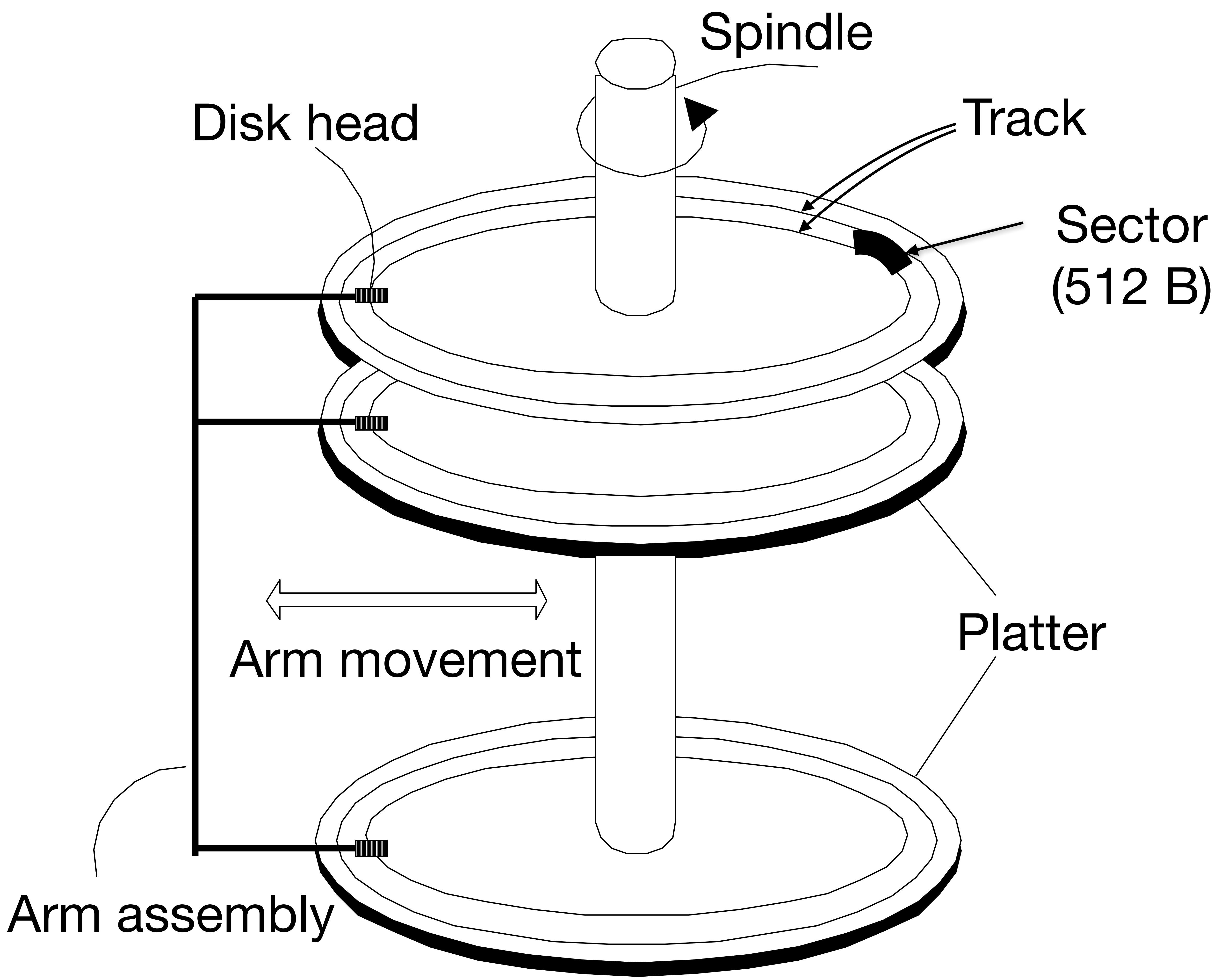
Sequential access?
Random access?



disk access principles

(1) Small random reads/writes are slow

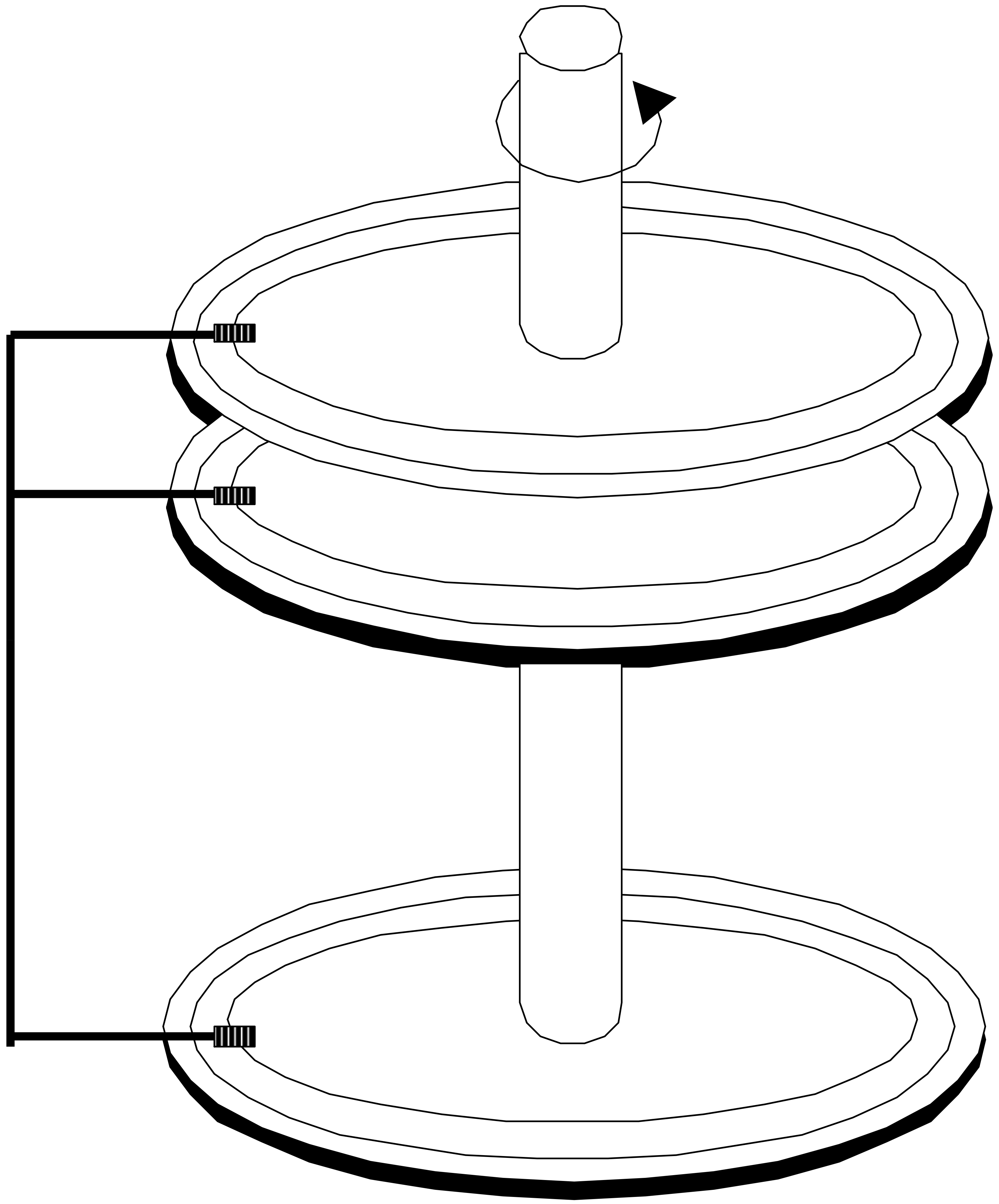
(2) Large sequential reads/writes of adjacent sectors and tracks are fast



Block device interface

Sector addresses

0	512 B
1	512 B
2	512 B
3	512 B
4	512 B
5	512 B
6	512 B
7	512 B
8	512 B
9	512 B
...	...



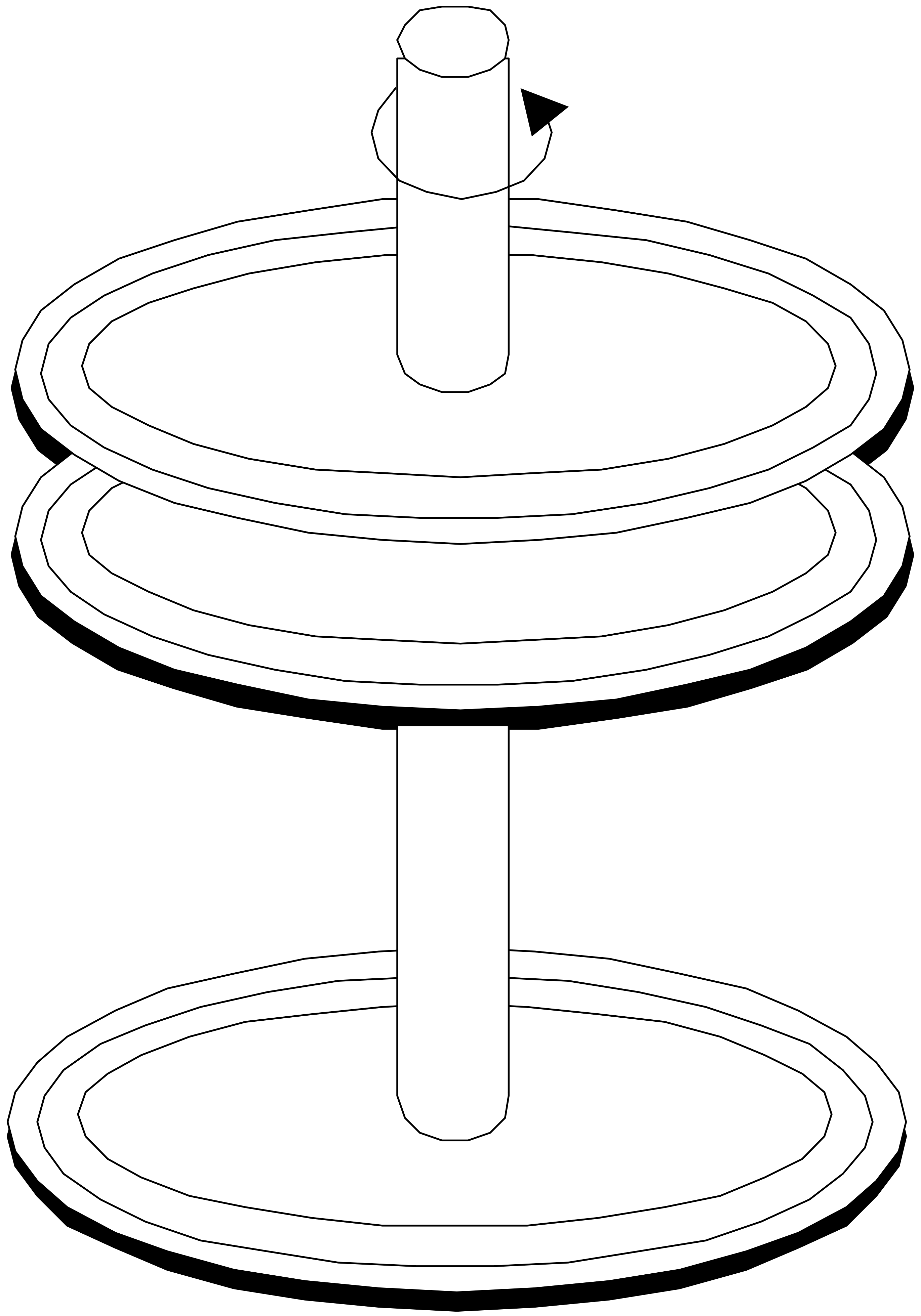
Block device interface

Operating System

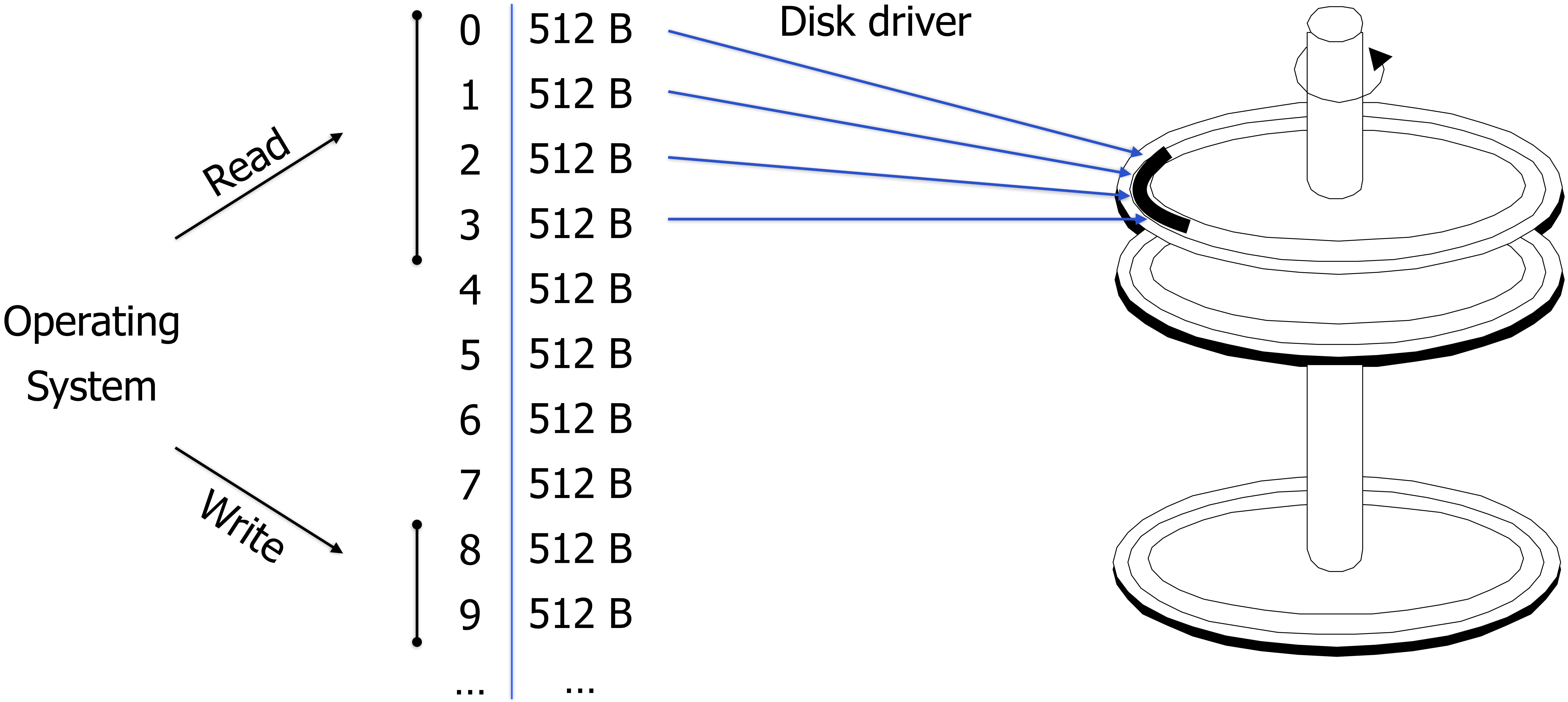
Read

Write

0	512 B
1	512 B
2	512 B
3	512 B
4	512 B
5	512 B
6	512 B
7	512 B
8	512 B
9	512 B
...	...



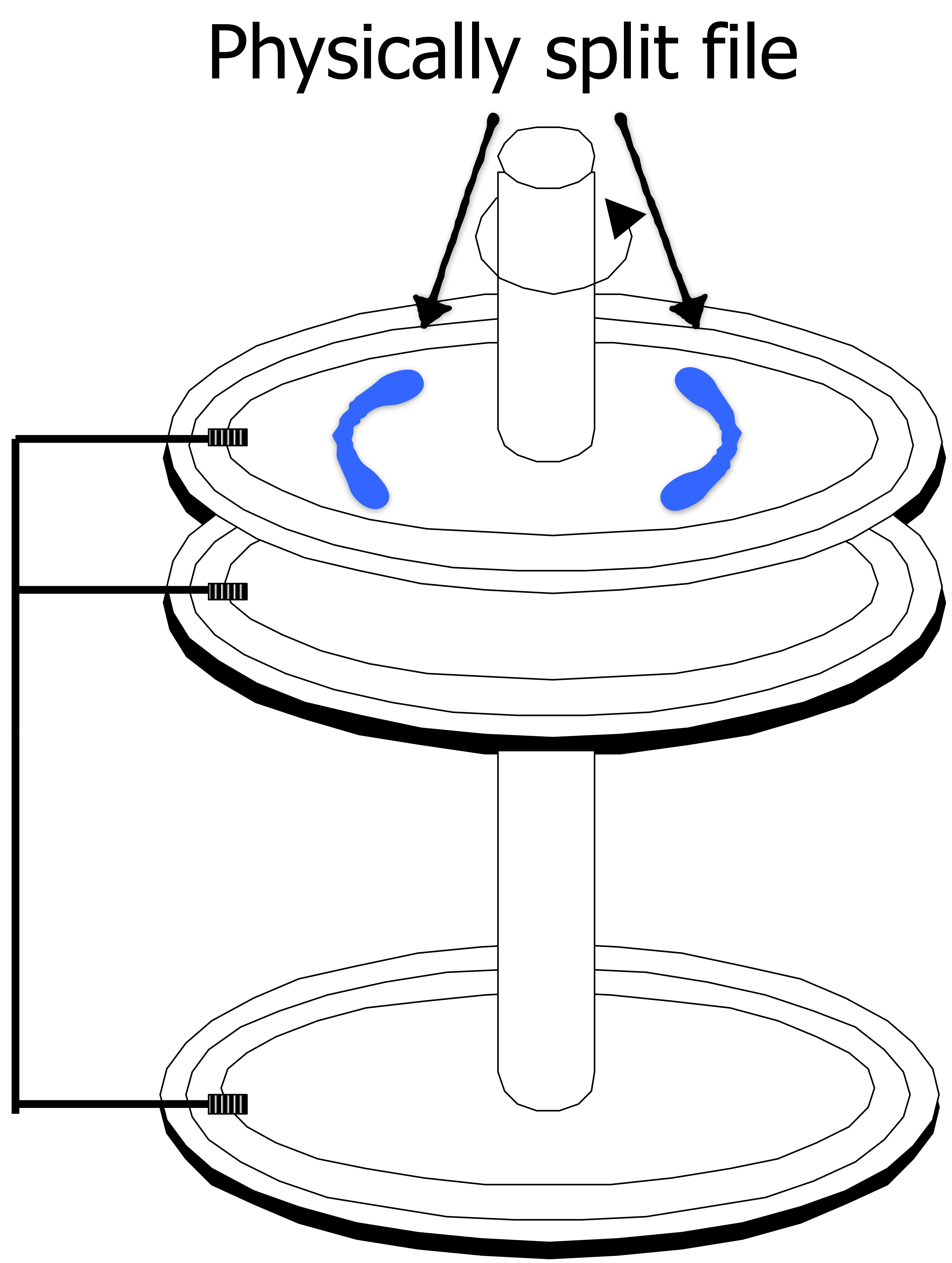
Block device interface



Problem

A file system or database may not initially find space to store data that's usually accessed together at the same place.

Solution?

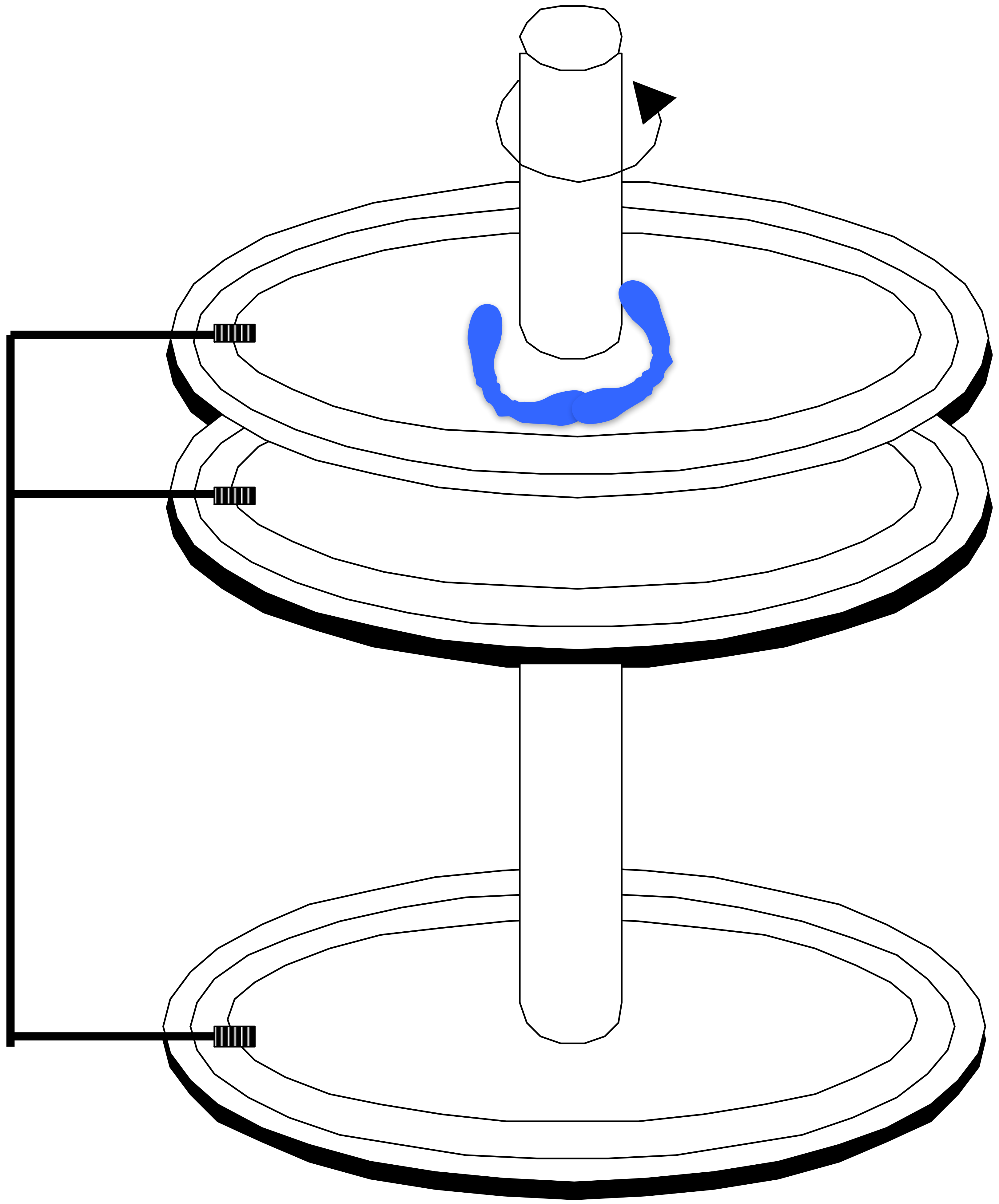


Defragmentation

A file system or database may not initially find space to store data that's usually accessed together at the same place.

Defragmentation fixes this by reorganizing disk so data belonging to the same file is close. This leads to fewer random accesses.

Databases do this too, as we'll see.



Disk Failure

Heat

Power surges

Dust

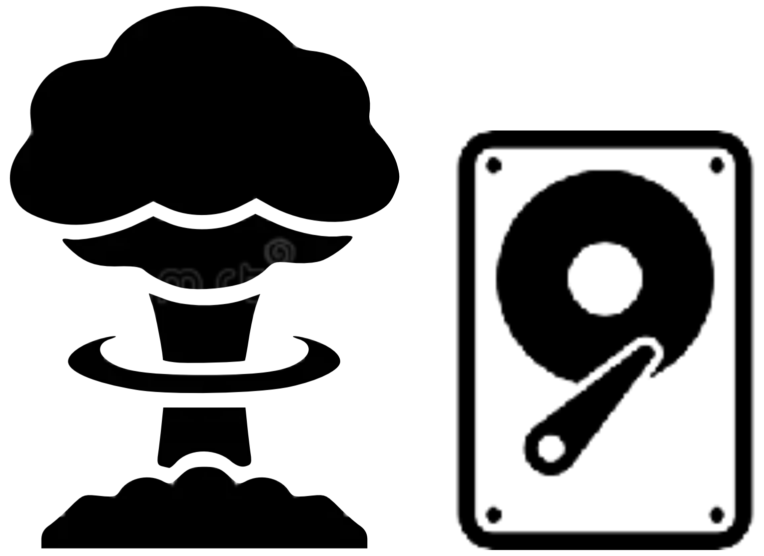
Time

Humidity

Disk Failure

Heat

Power surges



Dust

Time

Humidity

≈1% of a collection of disks will fail per year

disk

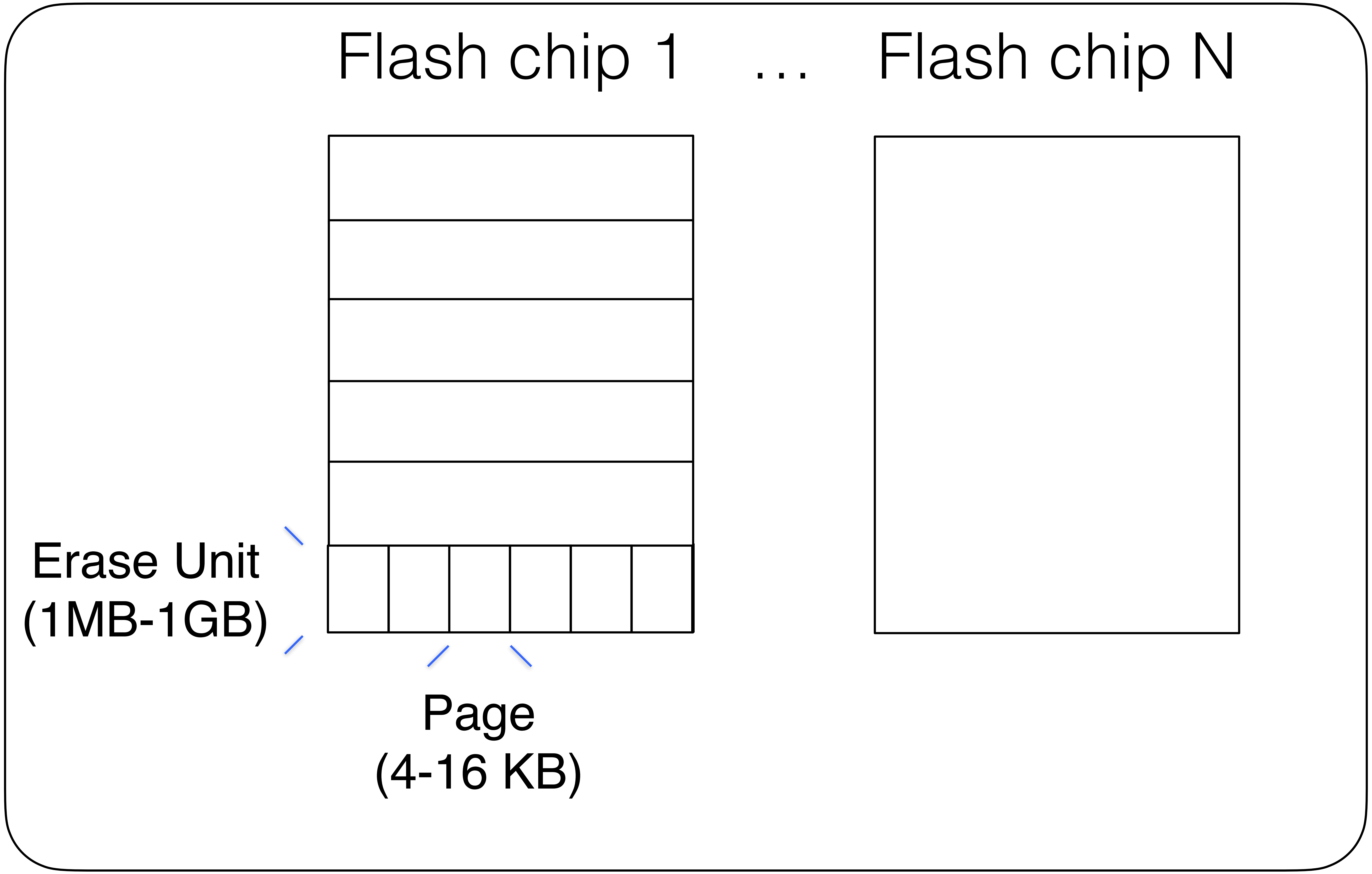
Traditional storage medium for decades

SSD

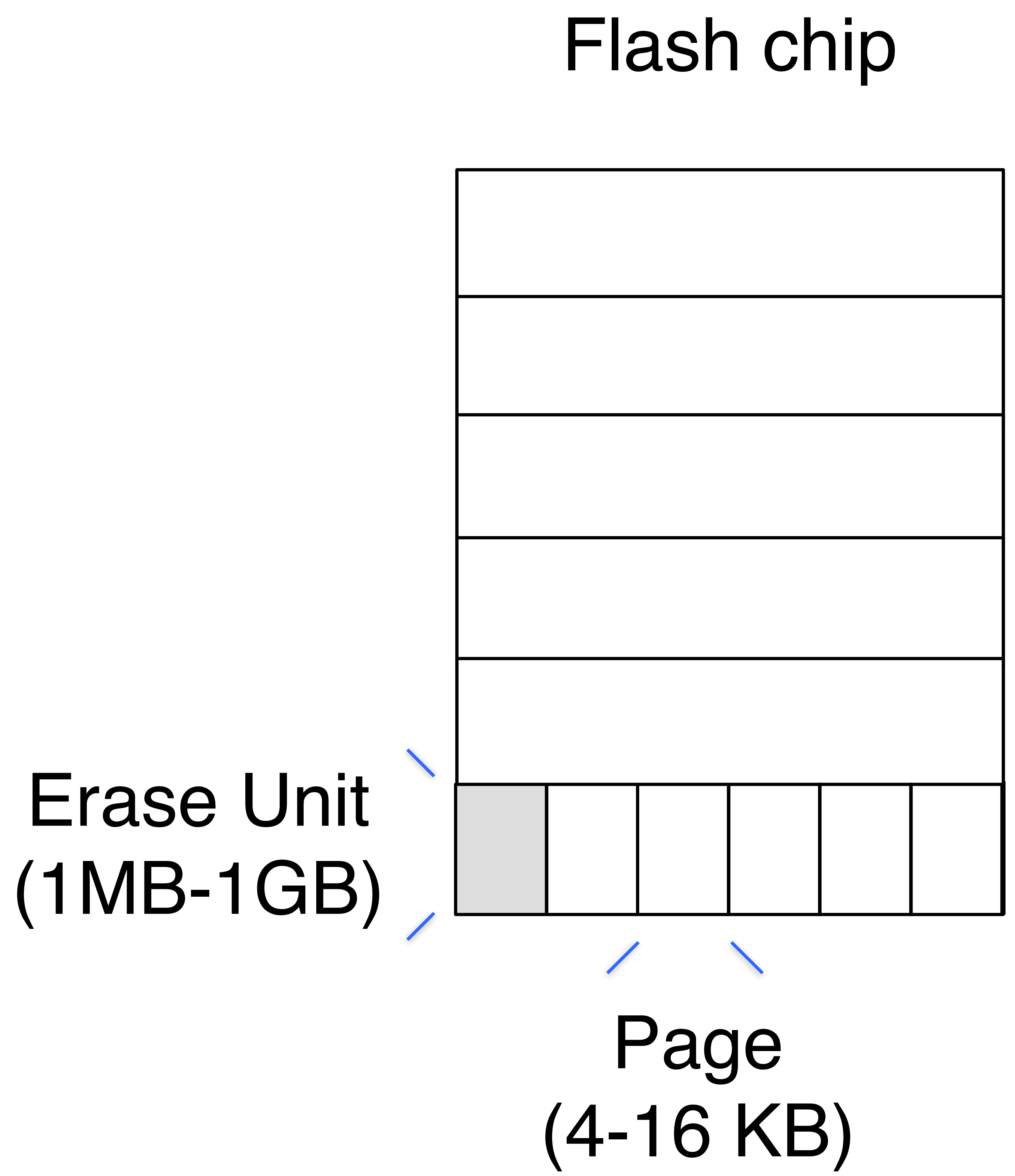
Became mainstream in the 2000s

SSD

SSD

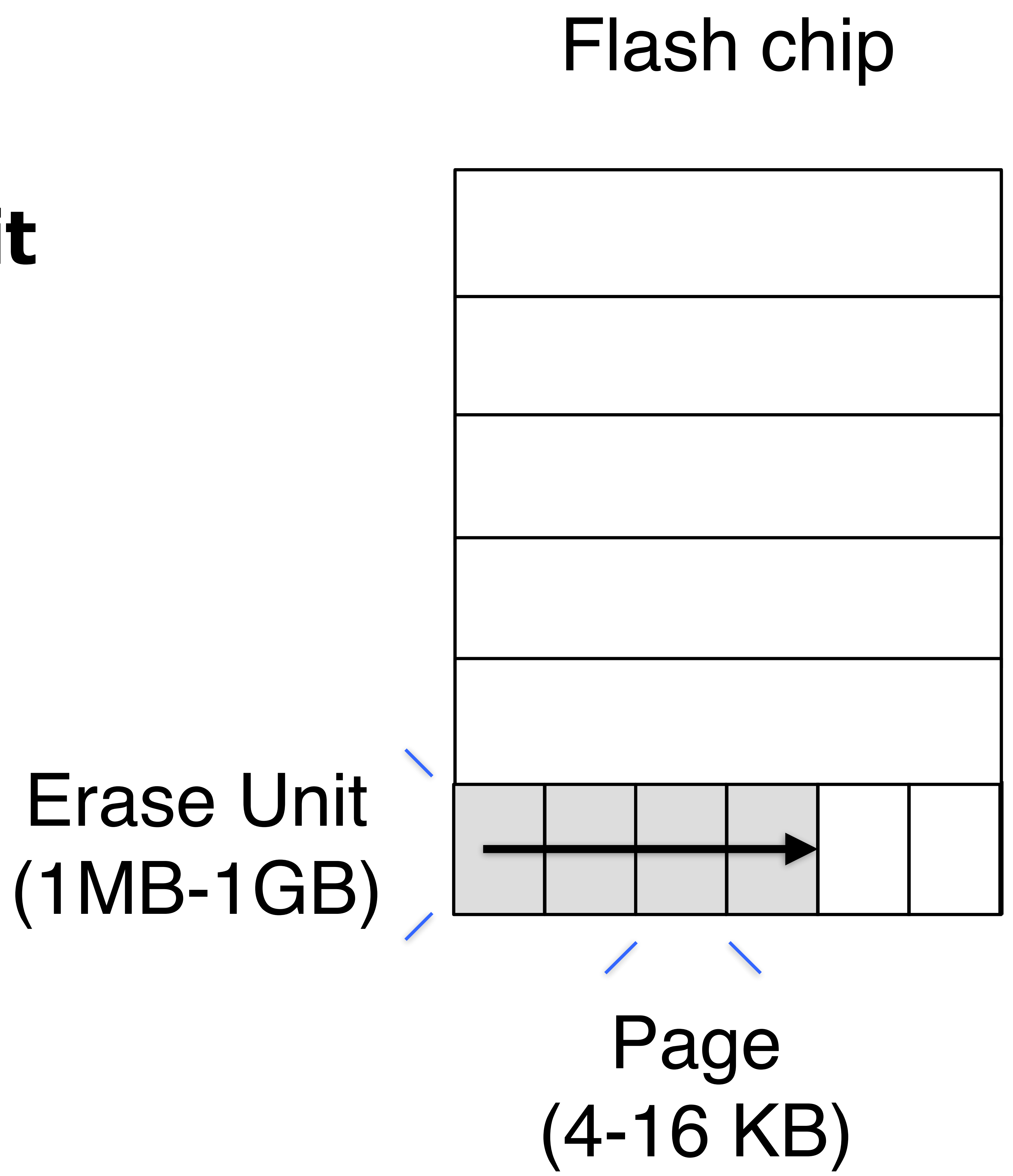


A page is the minimum read/write unit



A page is the minimum read/write unit

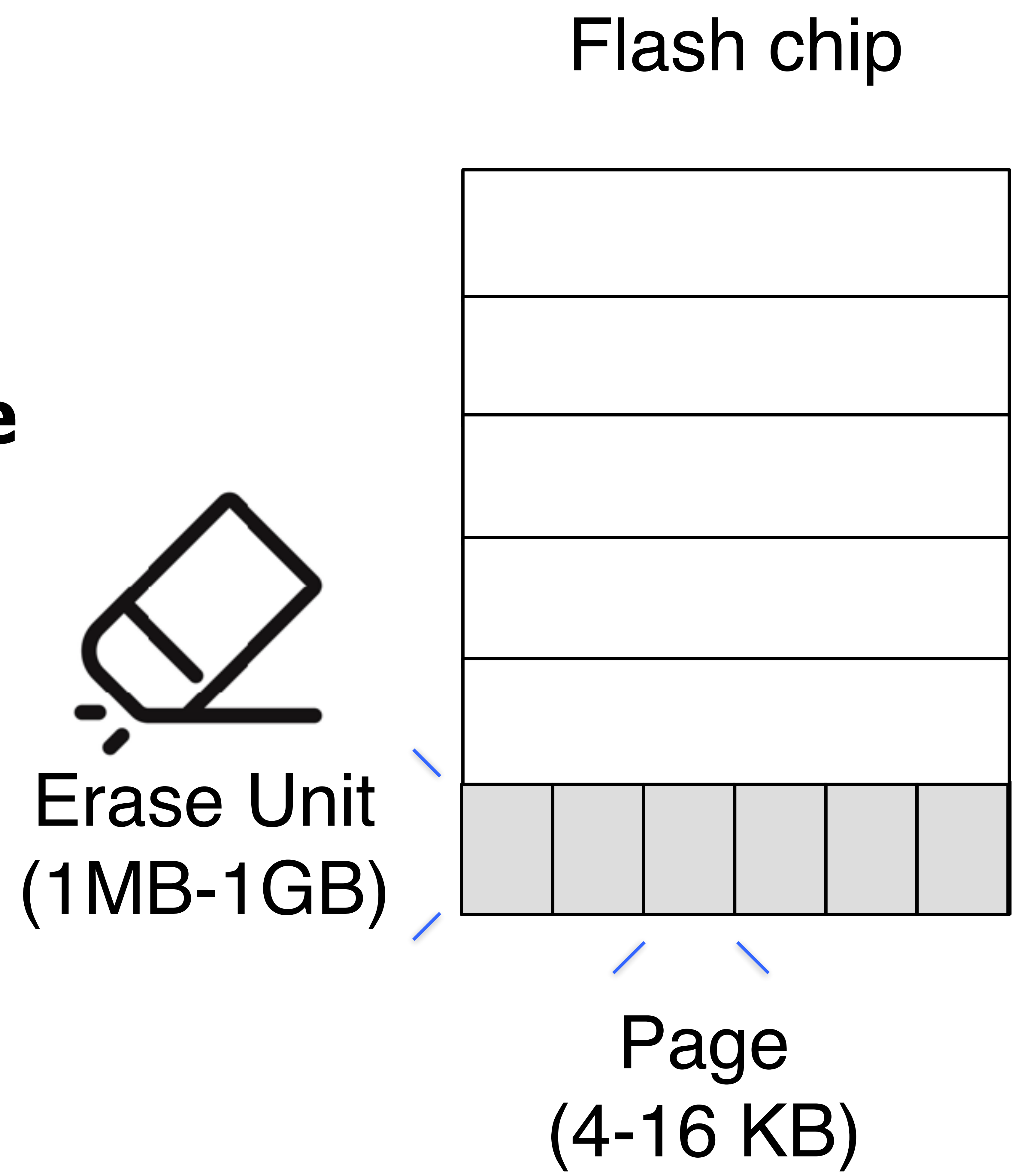
Pages must be written sequentially in an erase unit



A page is the minimum read/write unit

Pages must be written sequentially in an erase unit

All data in an erase unit is erased at the same time

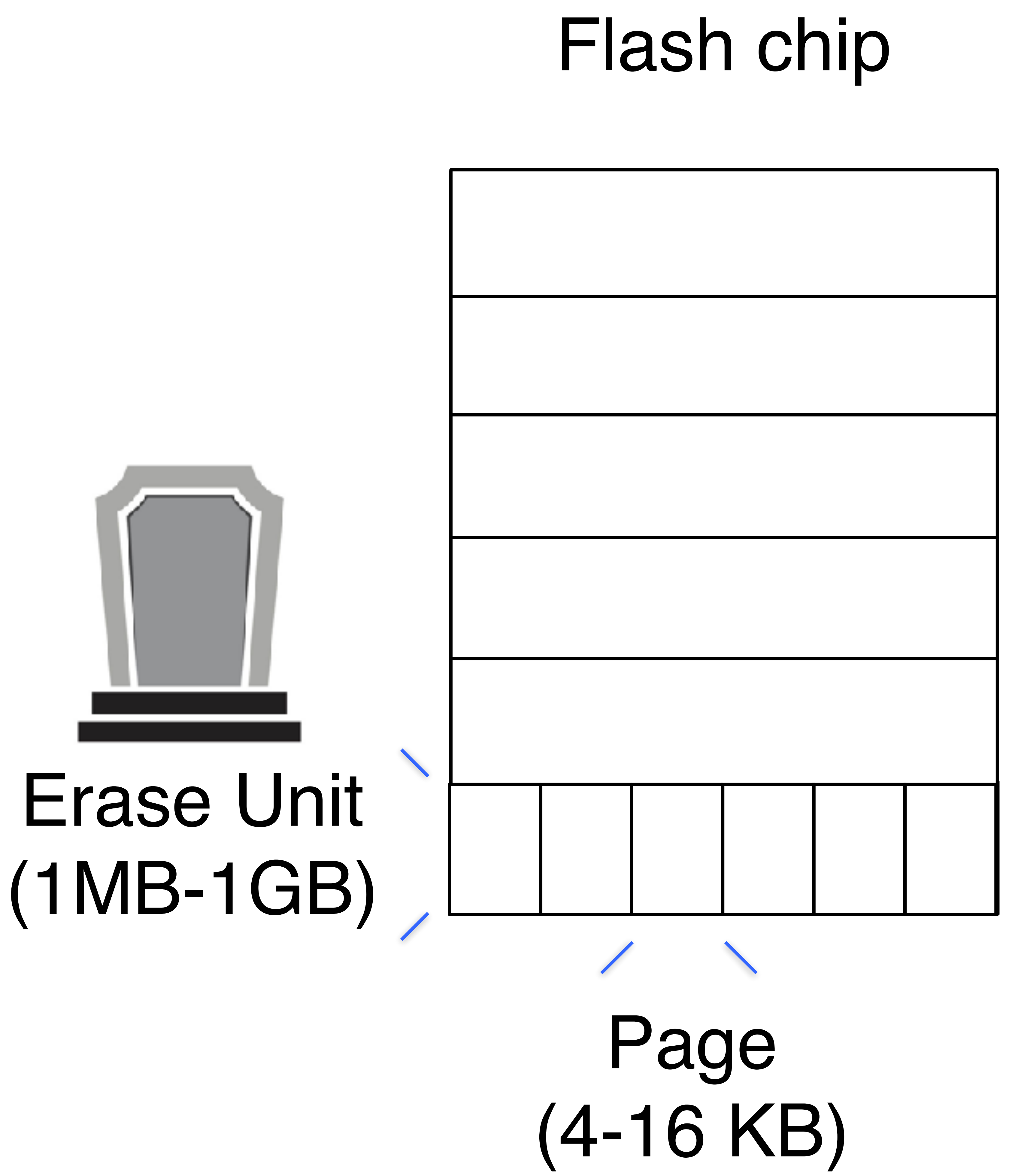


A page is the minimum read/write unit

Pages must be written sequentially in an erase unit

All data in an erase unit is erased at the same time

Each erase unit has a lifetime (1-10K erases)



A page is the minimum read/write unit

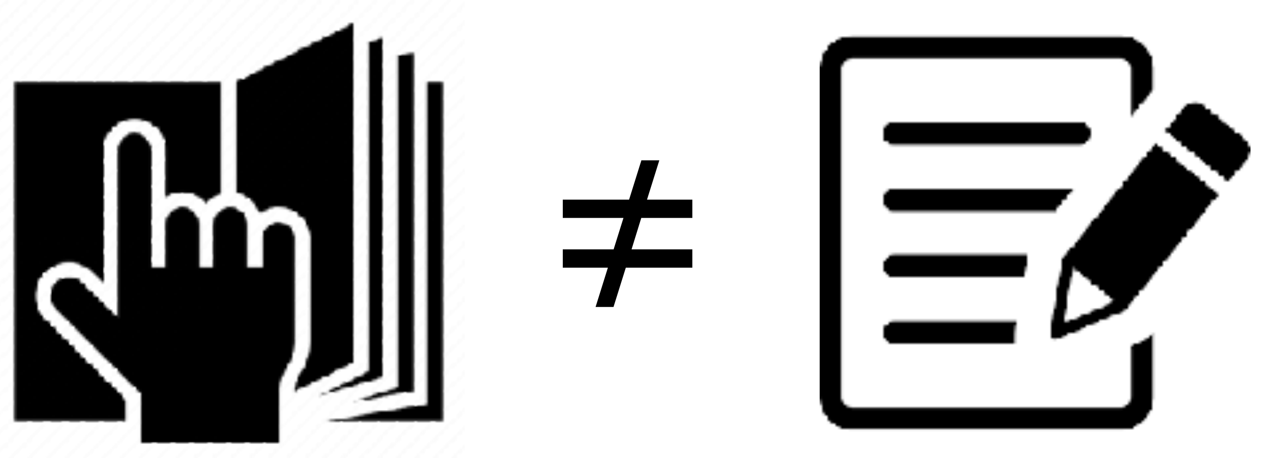
Pages must be written sequentially in an erase unit

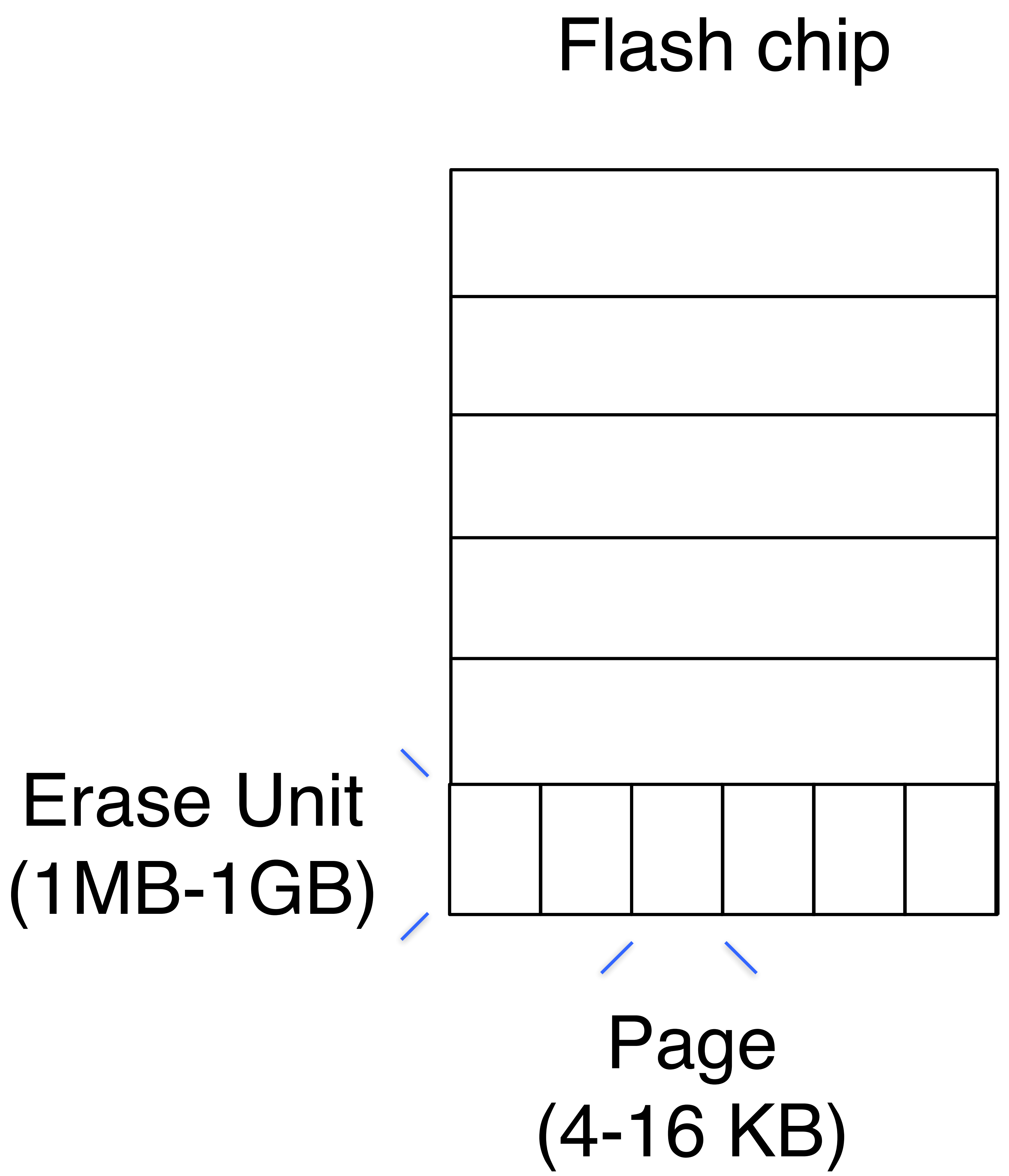
All data in an erase unit is erased at the same time

Each erase unit has a lifetime (1-10K erases)

Reading a page takes approx 50 us

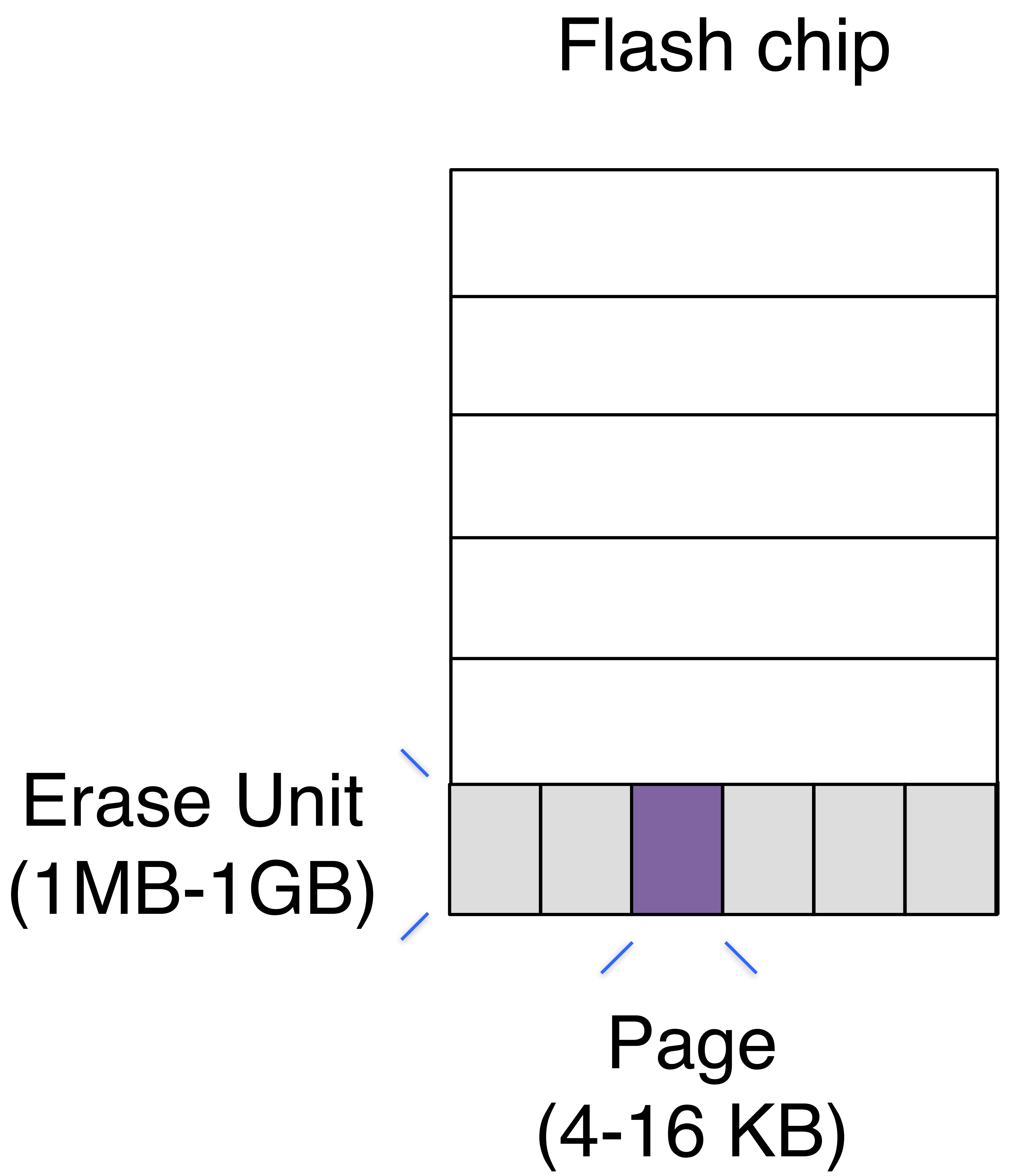
Writing a page takes approx 100-200 us





Updating a Page in-Place

What's the simplest way to update a page?

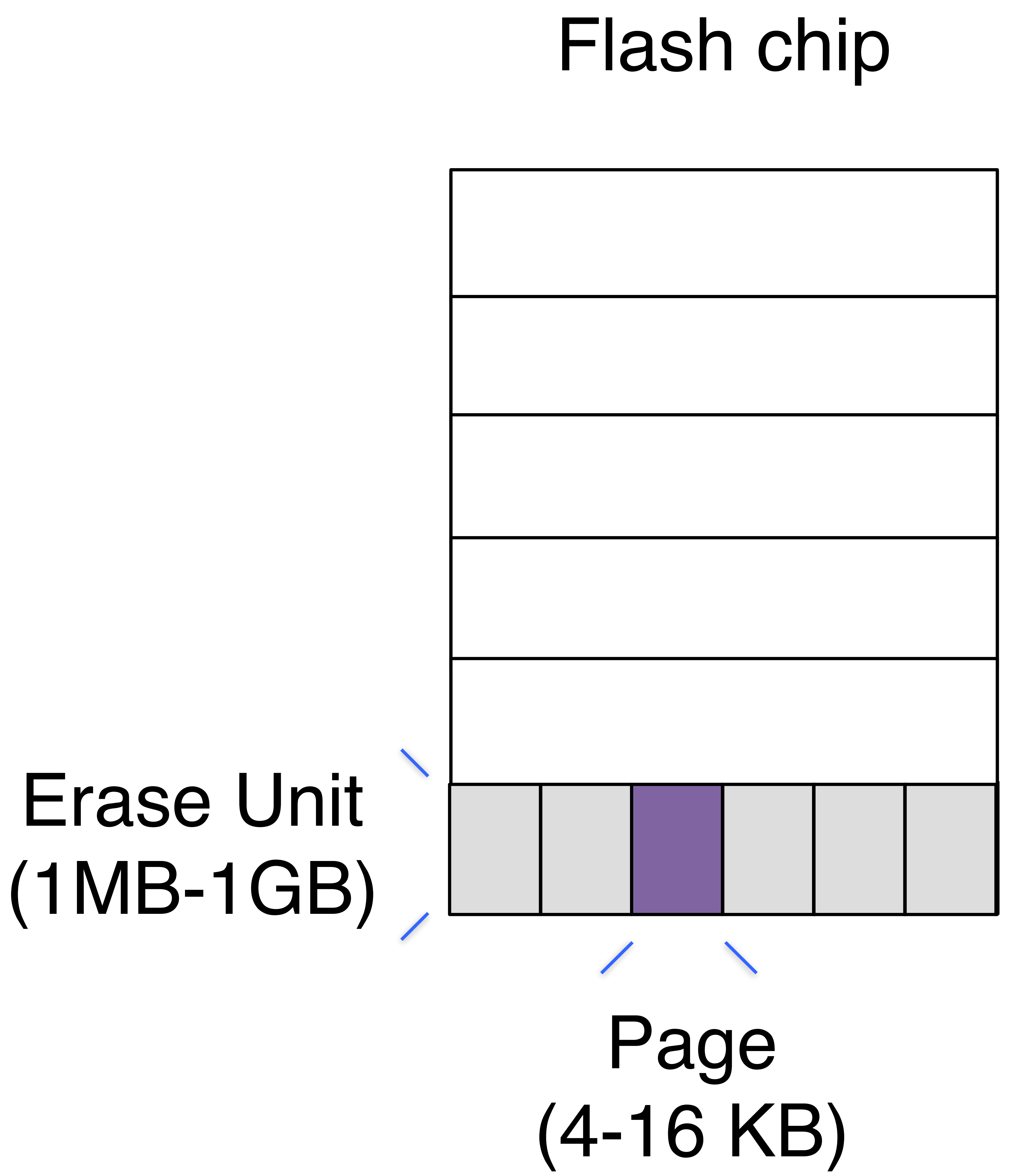


Updating a Page in-Place

What's the simplest way to update a page?

Read & rewrite the entire erase block

Why is this bad?



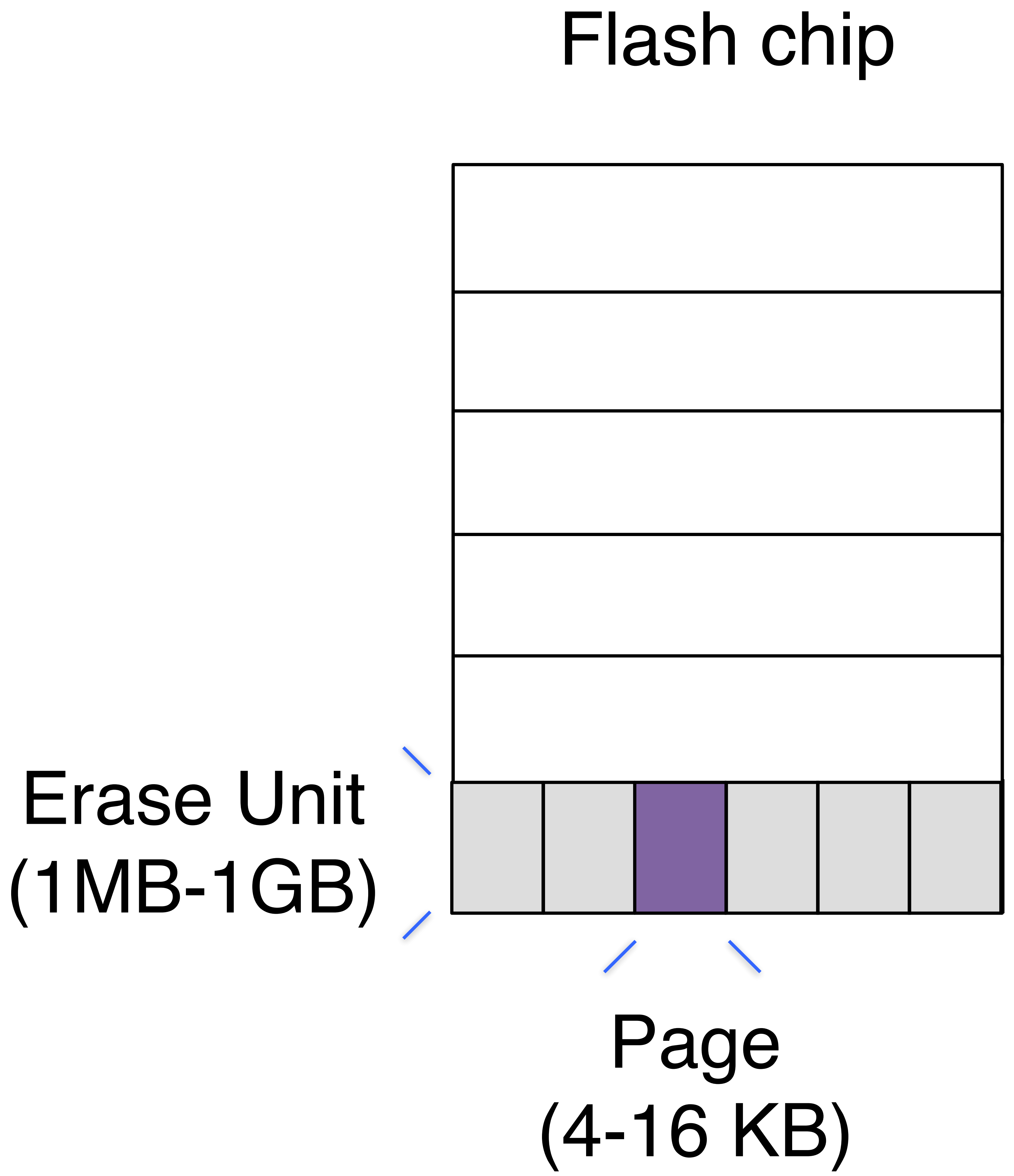
Updating a Page in-Place

What's the simplest way to update a page?

Read & rewrite the entire erase block

Why is this bad?

Suppose the purple page is repeatedly updated, while other data in the erase unit stays static.



Updating a Page in-Place

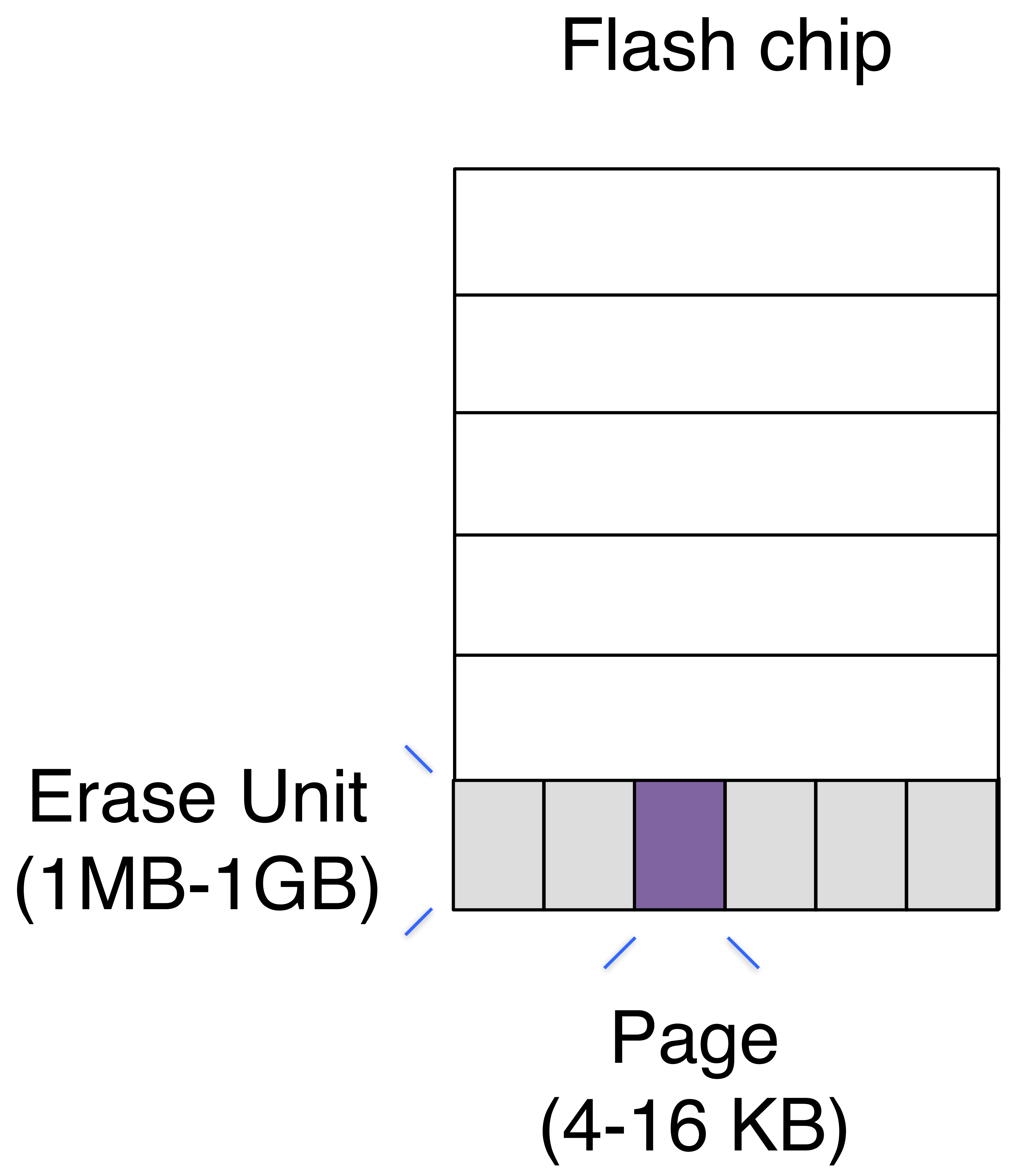
What's the simplest way to update a page?

Read & rewrite the entire erase block

Why is this bad?

Suppose the purple page is repeatedly updated, while other data in the erase unit stays static.

Physical work done >> work needed



Updating a Page in-Place

What’s the simplest way to update a page?

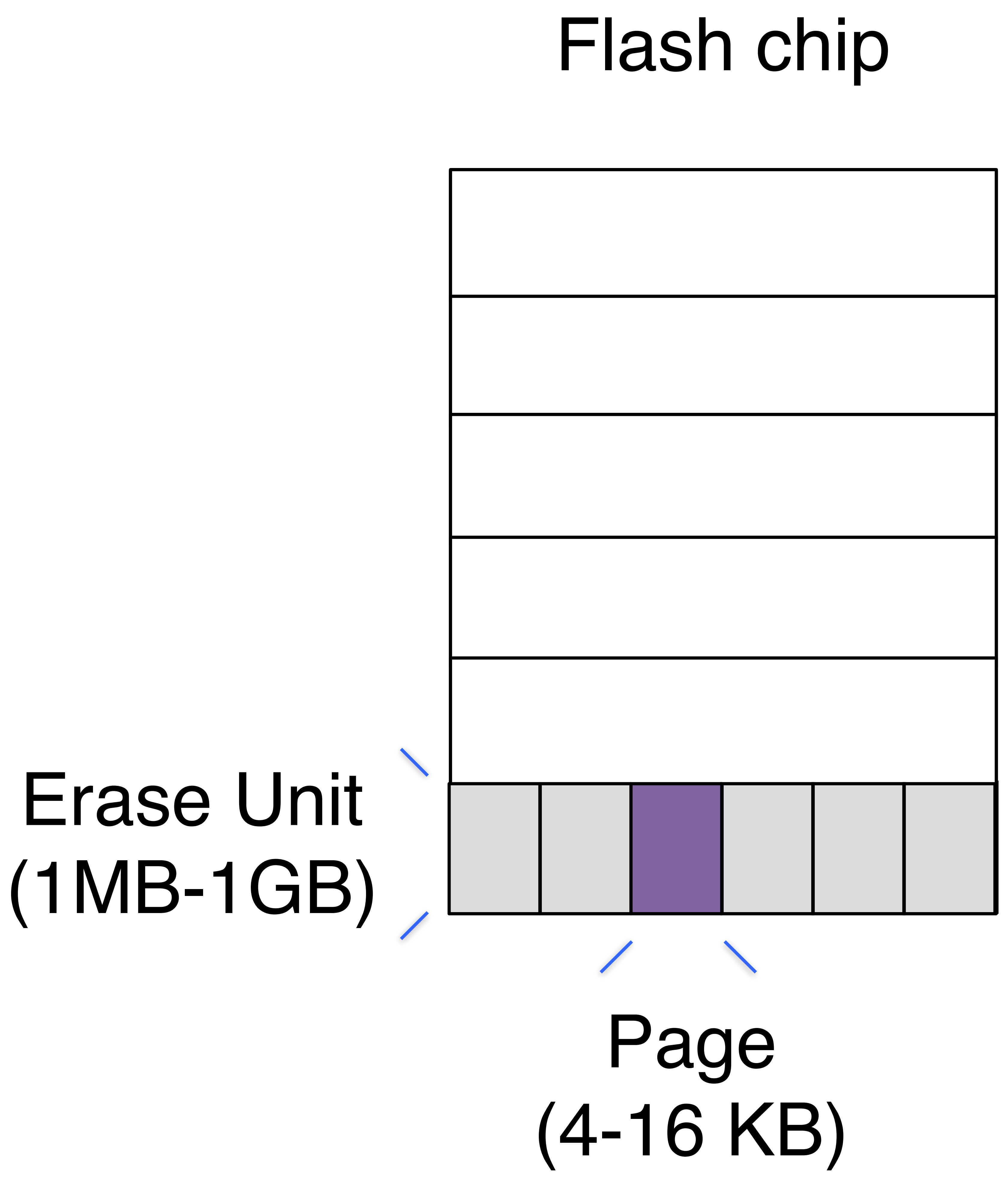
Read & rewrite the entire erase block

Why is this bad?

Suppose the purple page is repeatedly updated,
while other data in the erase unit stays static.

Physical work done >> work needed

$$\textbf{Write-amplification} = \frac{\textbf{Erase unit size}}{\textbf{Page size}}$$



Updating a Page in-Place

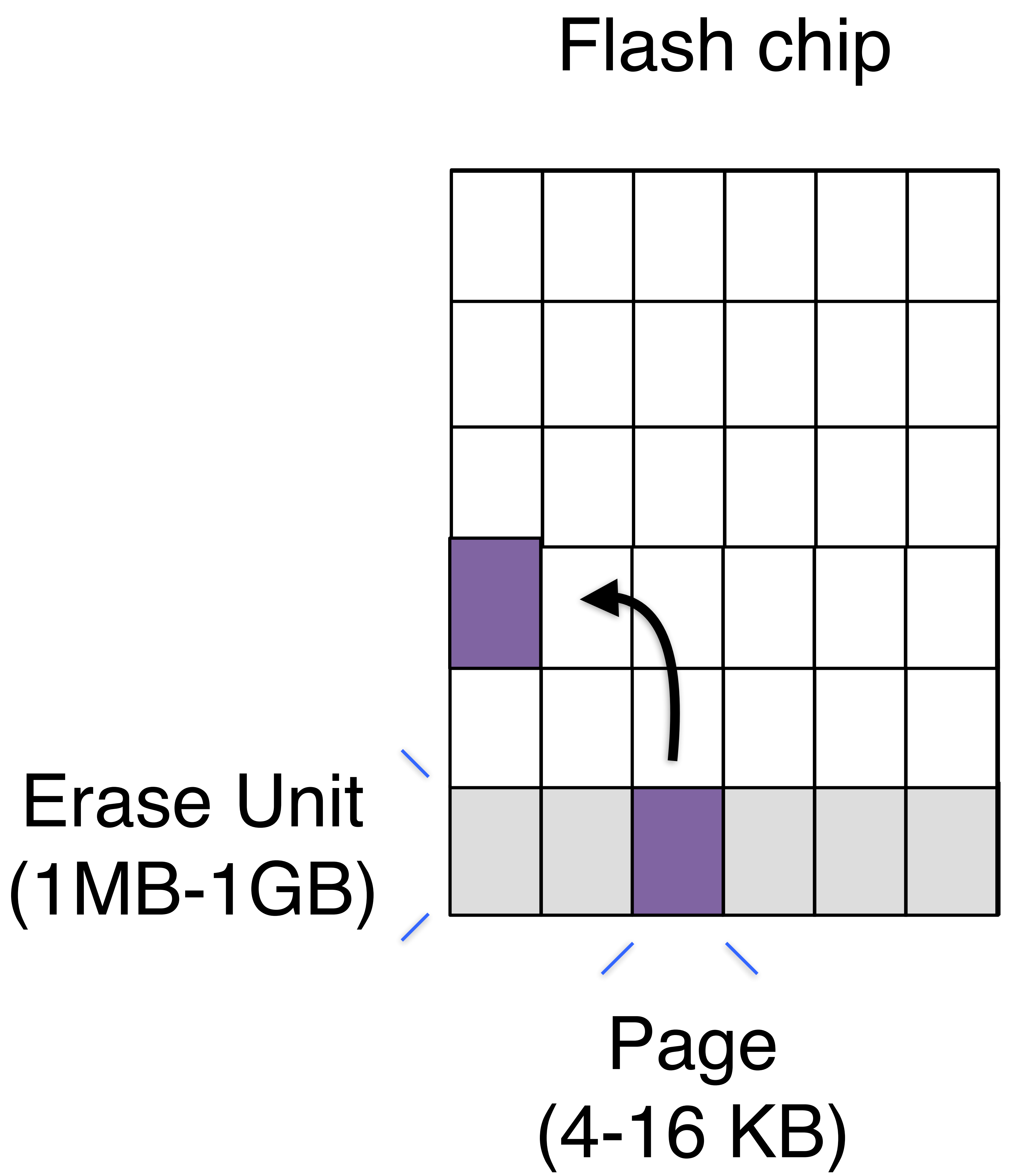
Better solution?

Updating a Page Out-of-Place

Better solution?

Copy the page to an erase unit with free space

“Copy-on-write” or “out-of-place update”



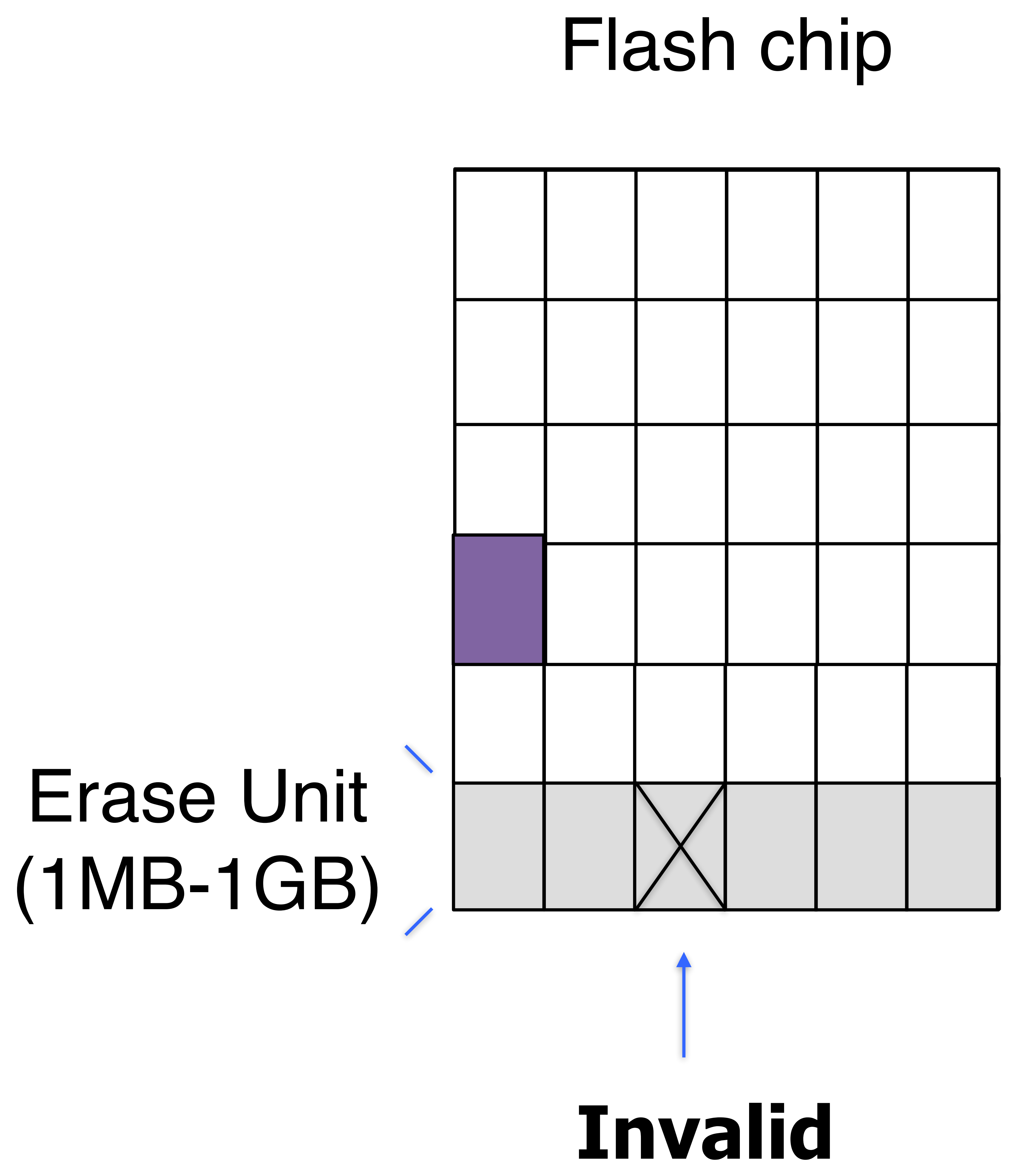
Updating a Page Out-of-Place

Better solution?

Copy the page to an erase unit with free space

"Copy-on-write" or "out-of-place update"

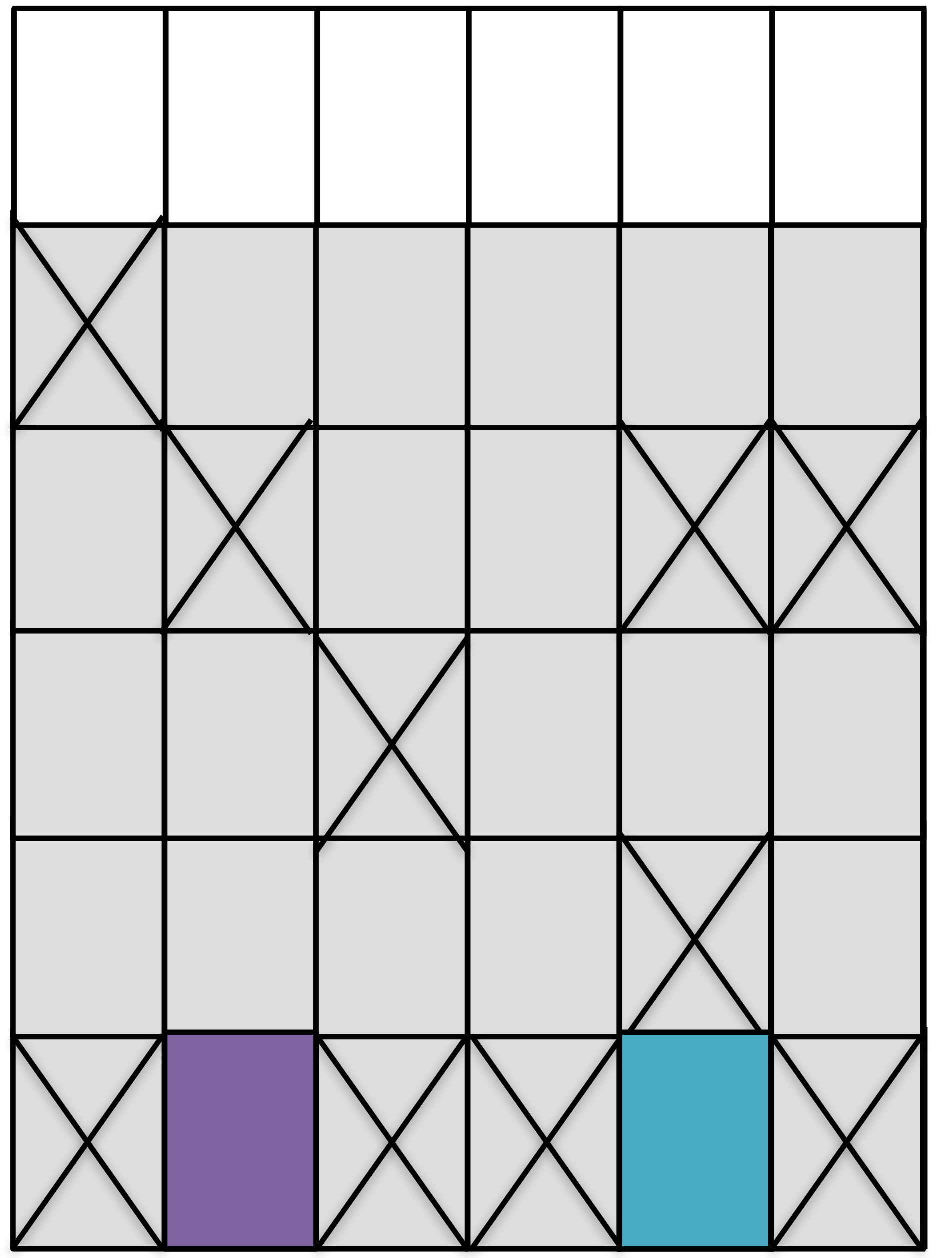
Mark the original page as invalid (using a bitmap)



Updating a Page Out-of-Place

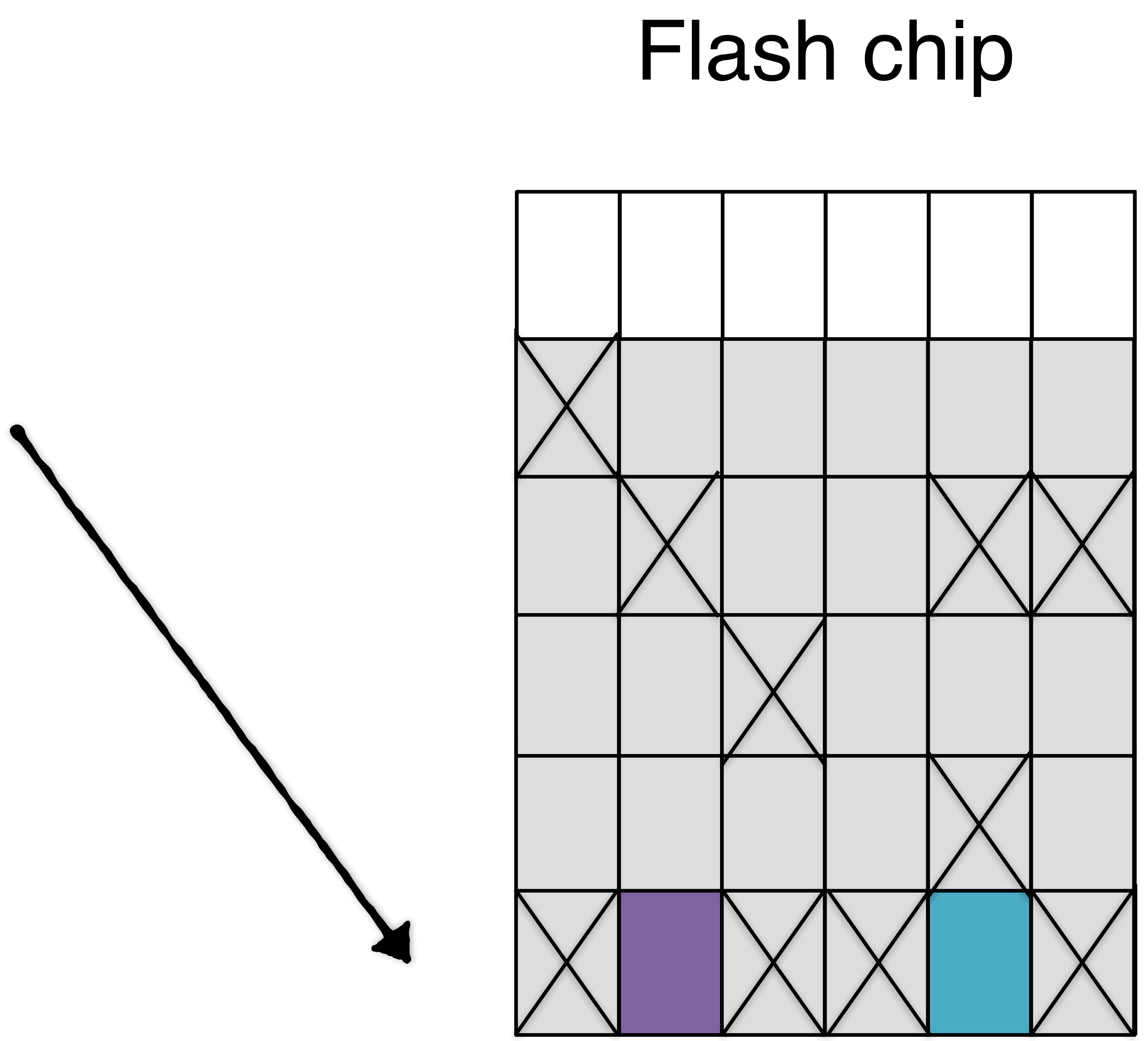
Eventually many invalid pages accumulate.

How do we reclaim space to support more writes?



Garbage-Collection

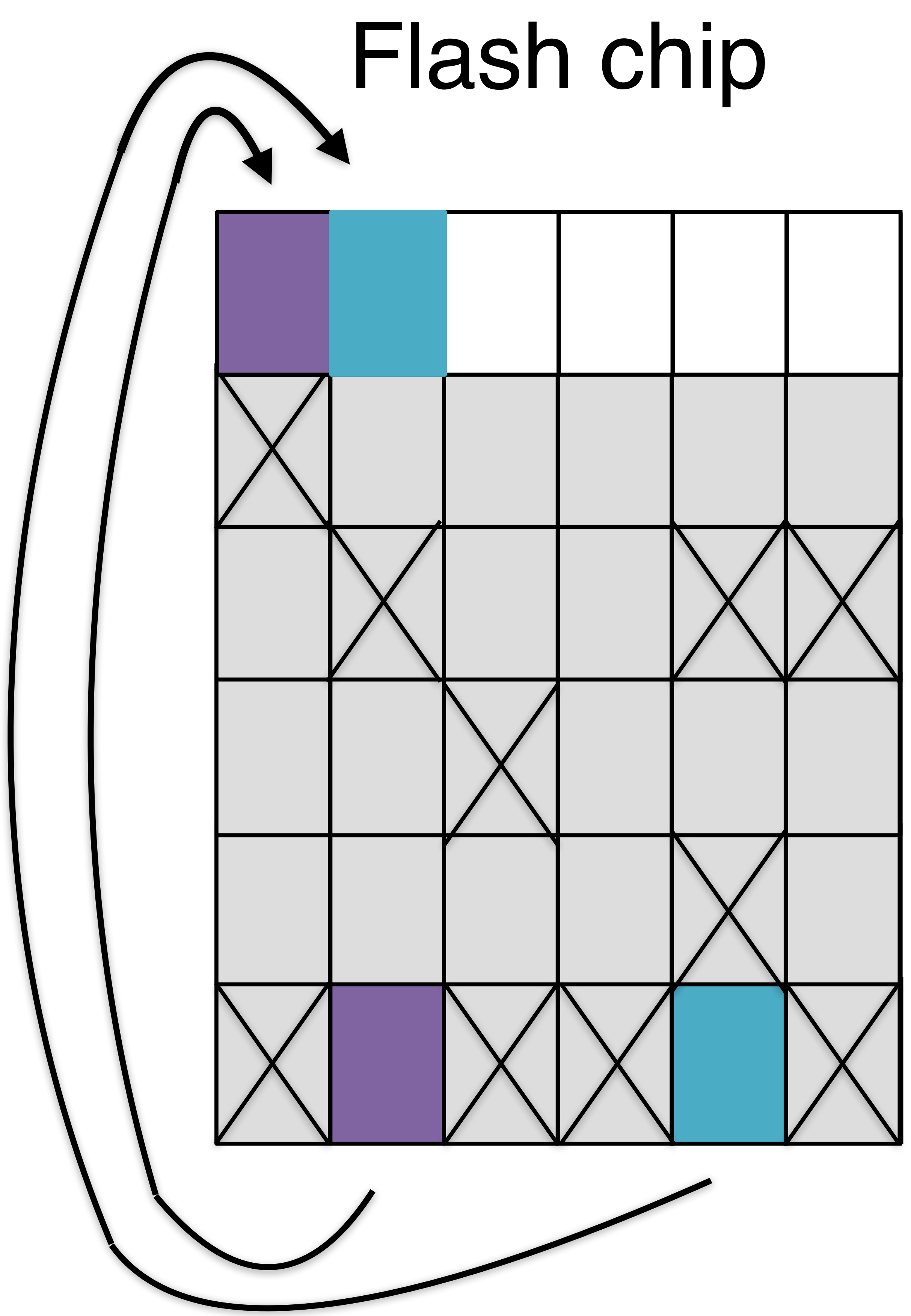
Find the erase unit with the least live data left.



Garbage-Collection

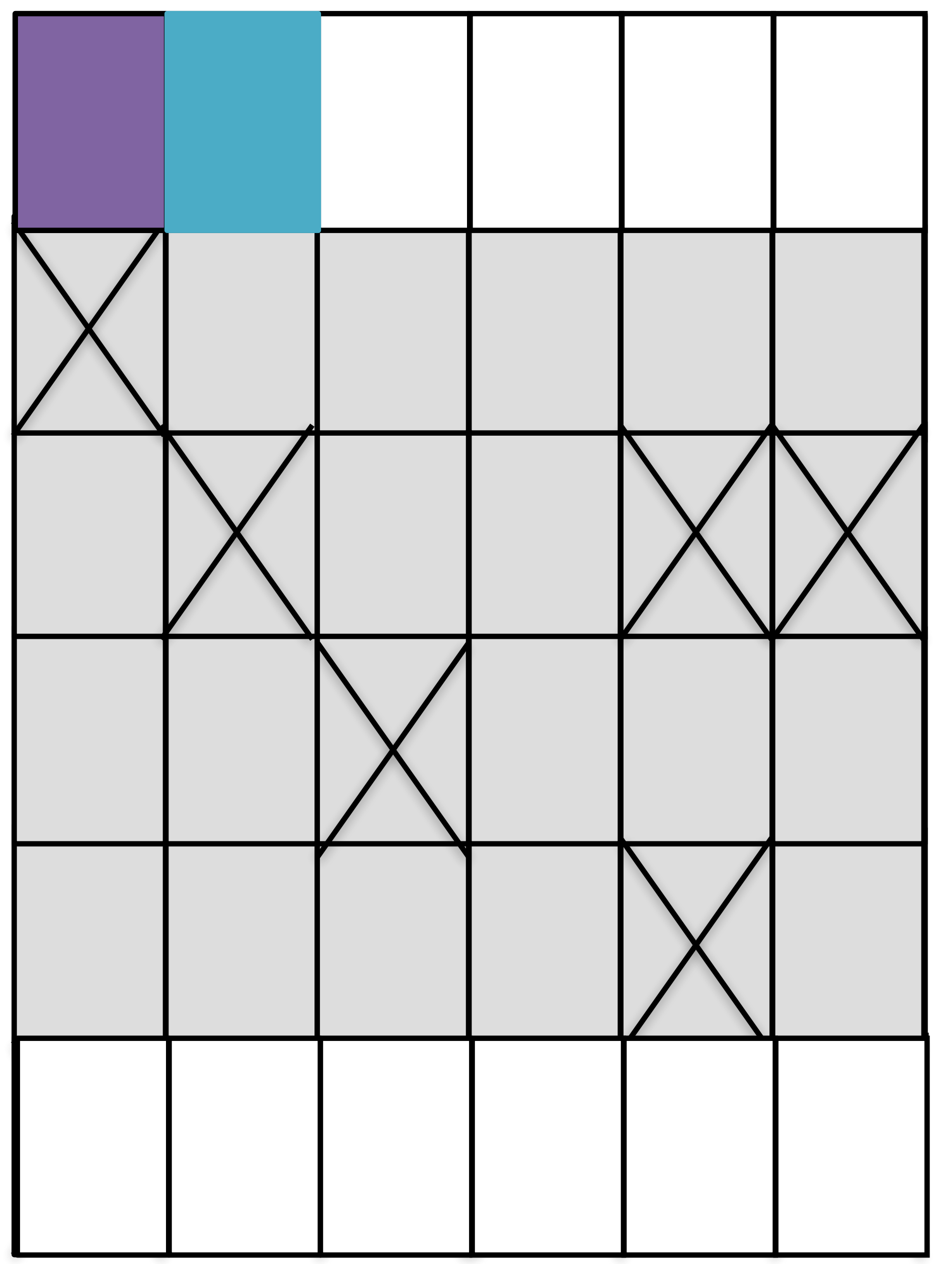
Find the erase unit with the least live data left.

Copy live pages to an erase unit with free space.



Garbage-Collection

Flash chip



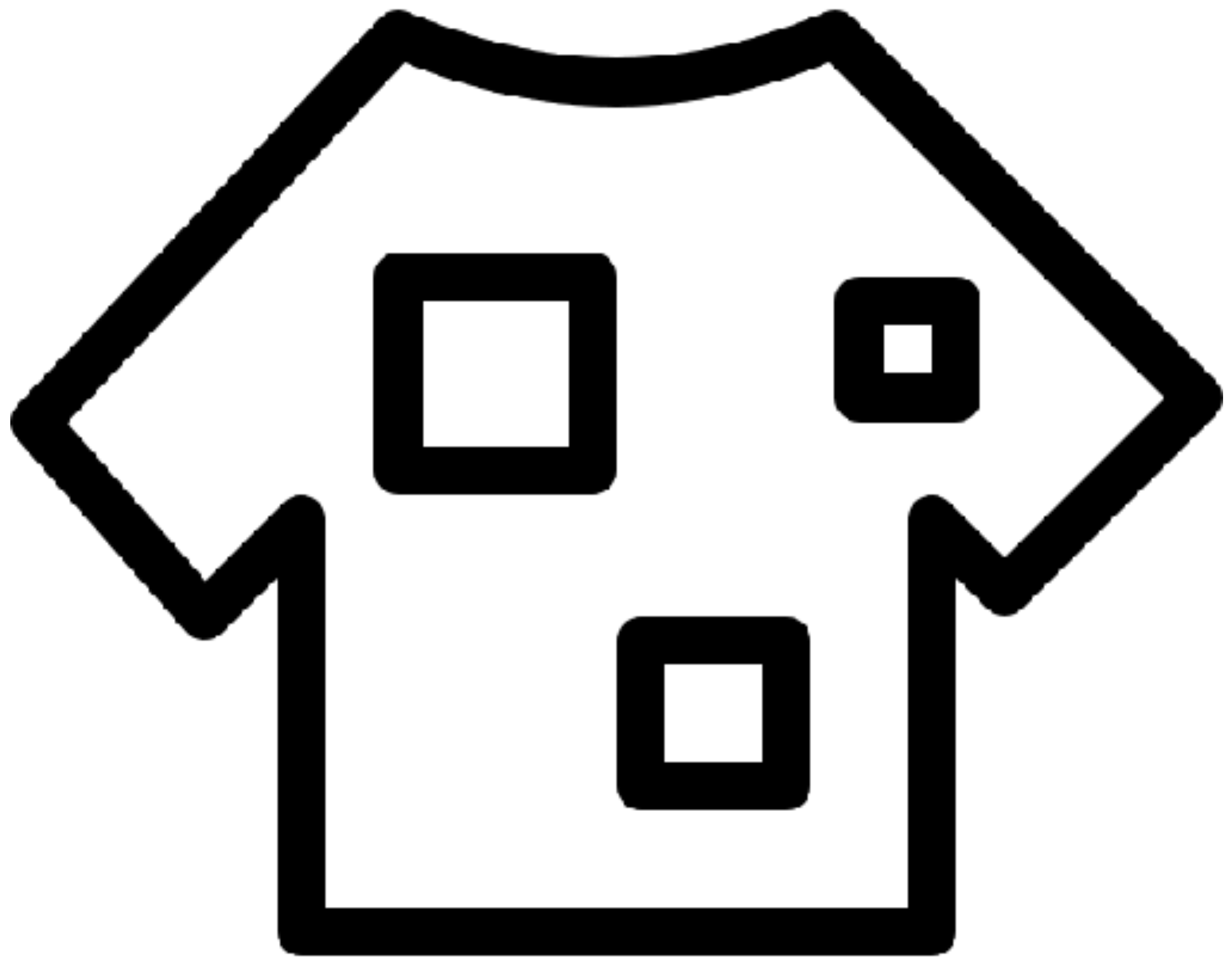
Find the erase unit with the least live data left.

Copy live pages to an erase unit with free space.

Erase the block

SSD Wearing Out

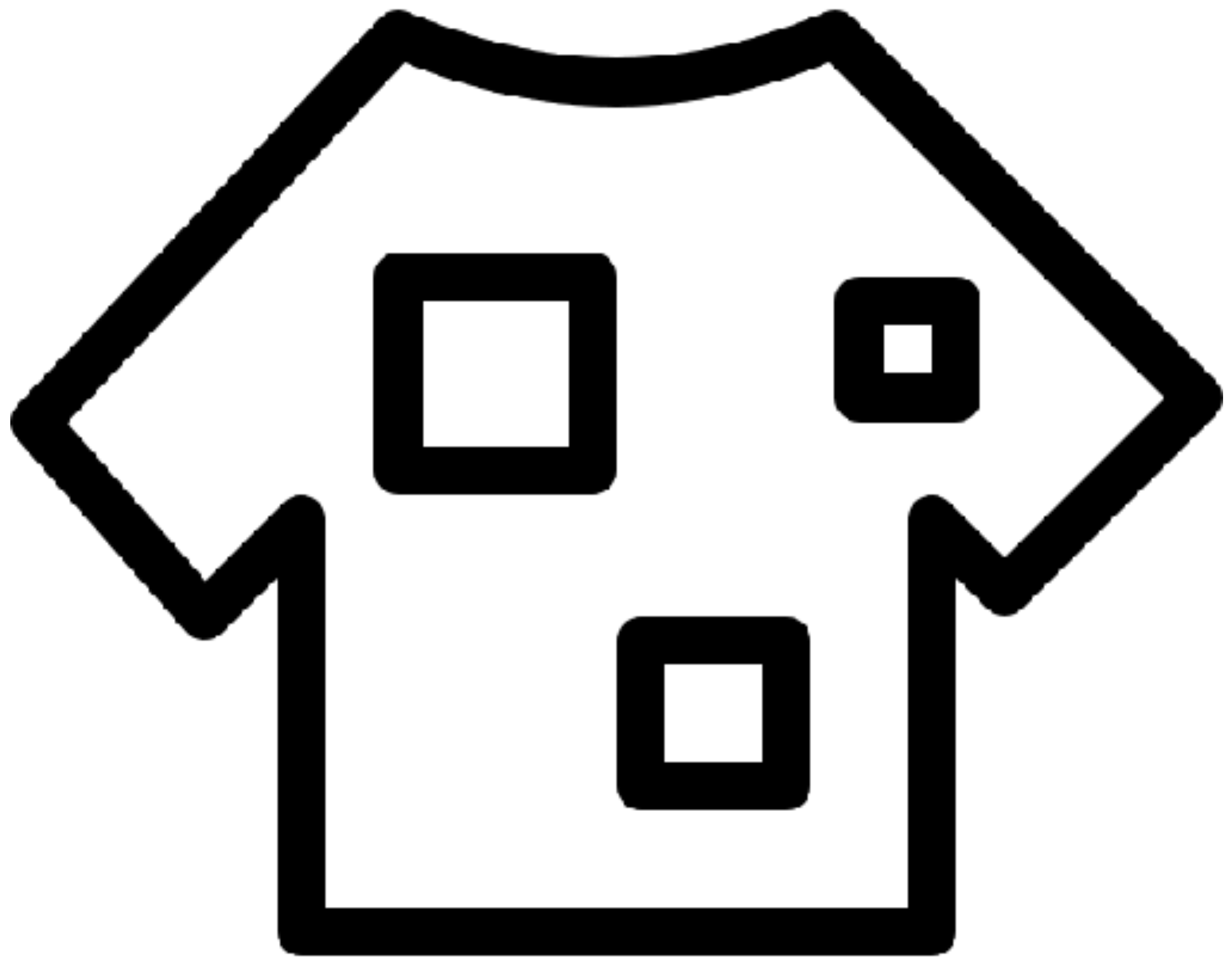
Cells become more error-prone as we write to them



SSD Wearing Out

Cells become more error-prone as we write to them

There are error correction codes internally, but eventually there is so much error they cannot correct it.

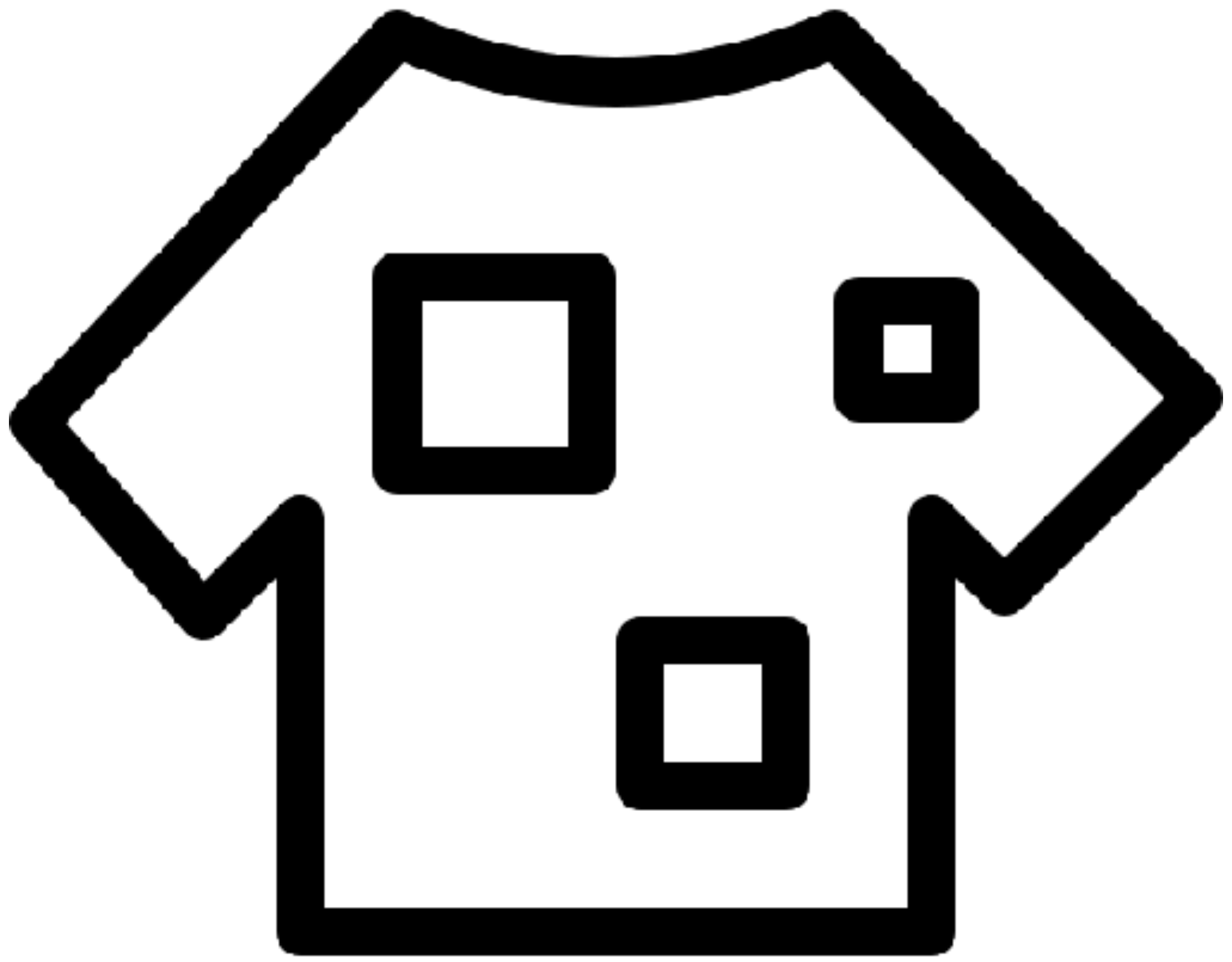


SSD Wearing Out

Cells become more error-prone as we write to them

There are error correction codes internally, but eventually there is so much error they cannot correct it.

An SSD will report in the specs how many sequential or random writes it can take before failing

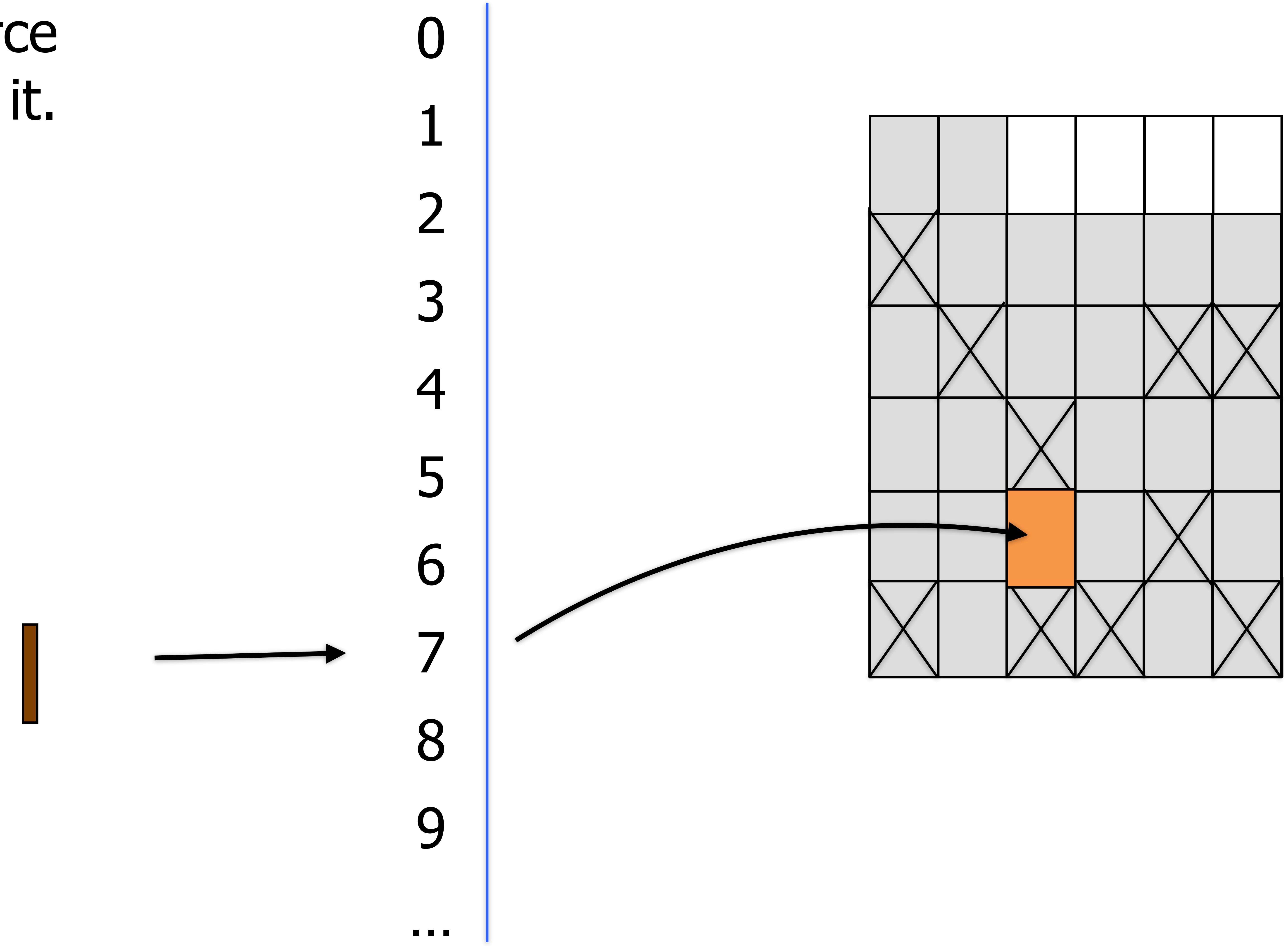


6 Principles for Making the Most of your SSD

Principle 1: Avoid Small Updates

Small Updates are terrible - they force reading a whole page and rewriting it.

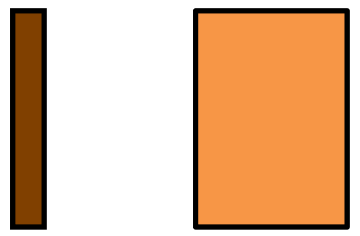
Application
Update smaller than page size

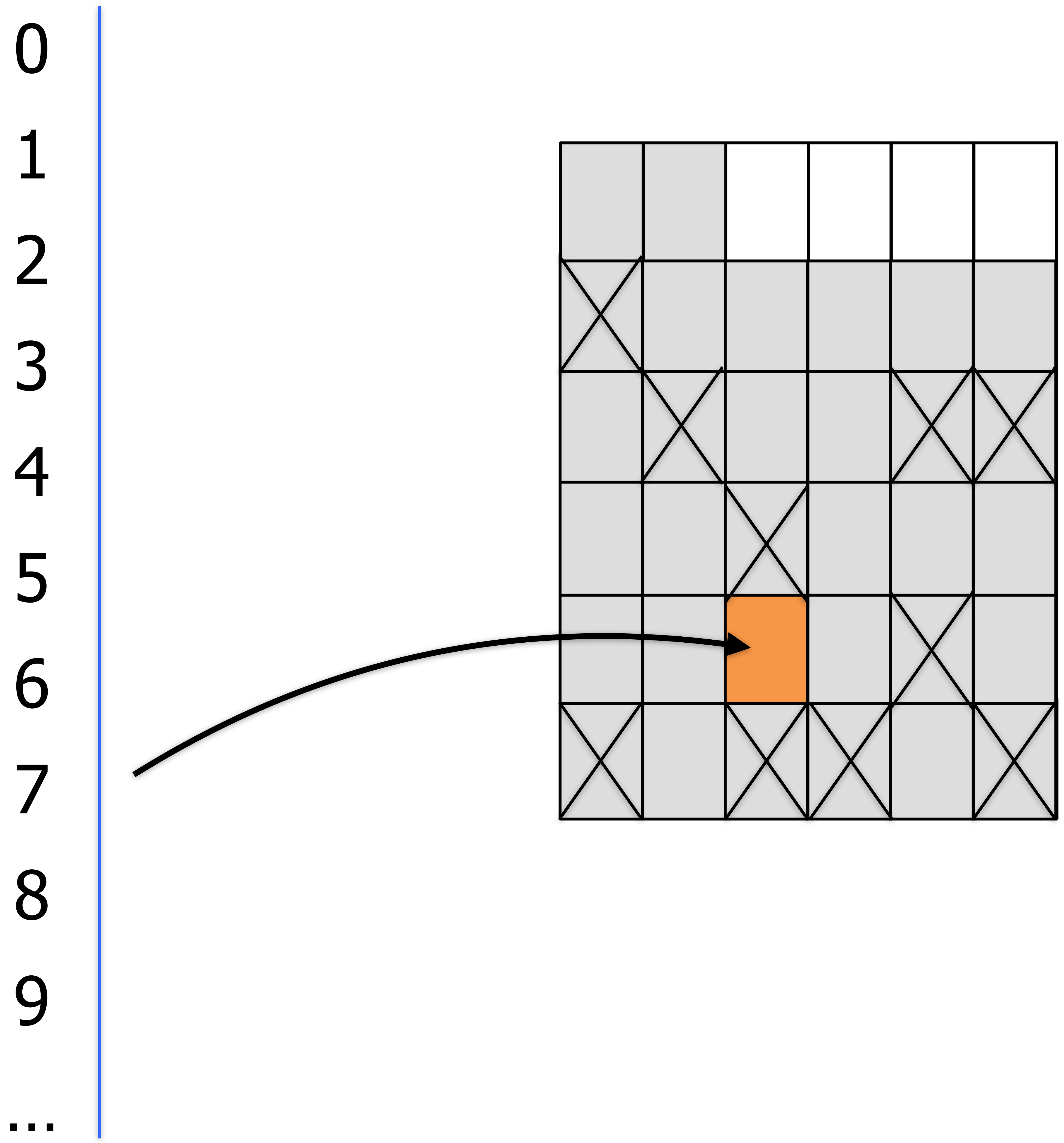


Principle 1: Avoid Small Updates

Small Updates are terrible - they force reading a whole page and rewriting it.

Application
Update smaller than page size

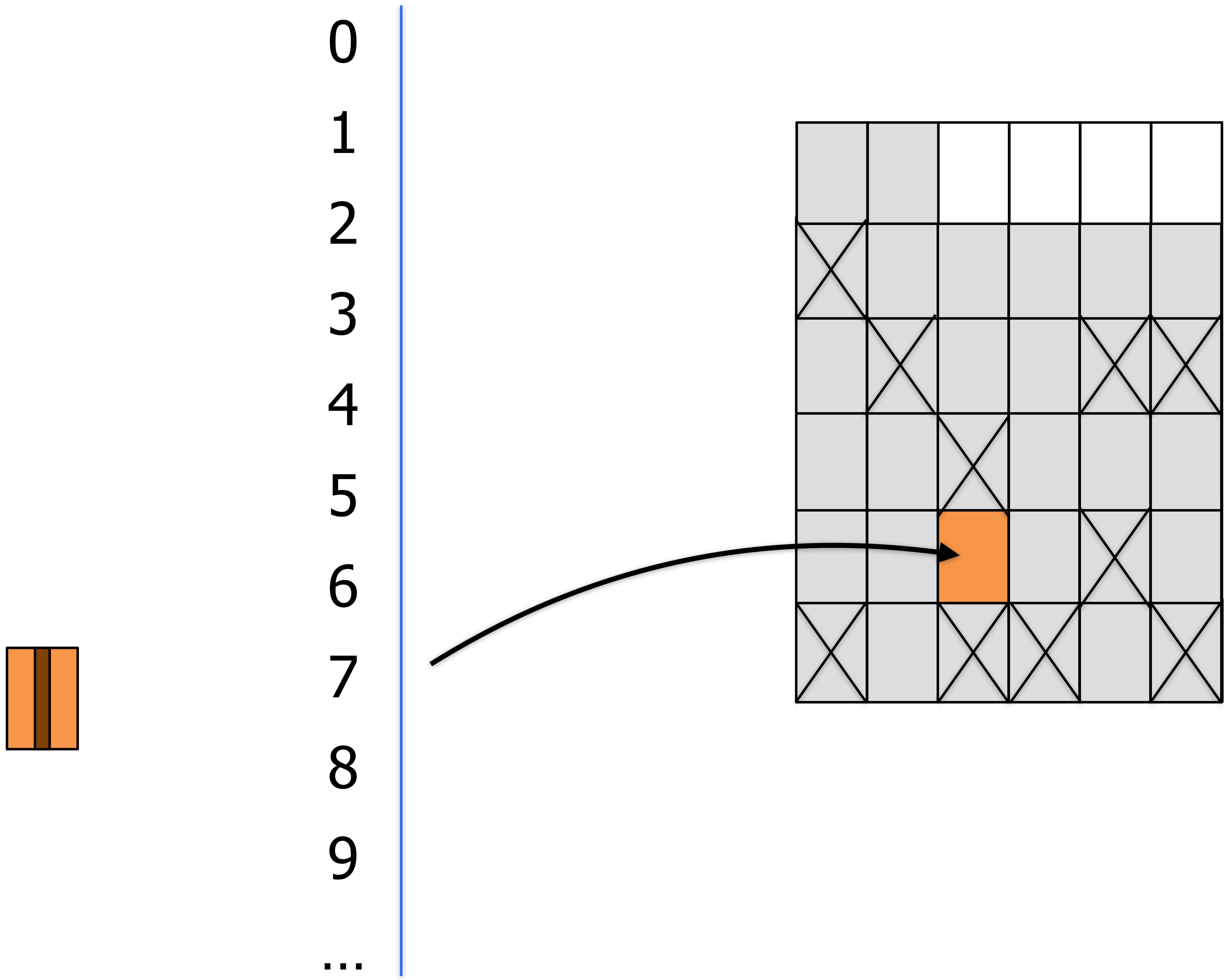




Principle 1: Avoid Small Updates

Small Updates are terrible - they force reading a whole page and rewriting it.

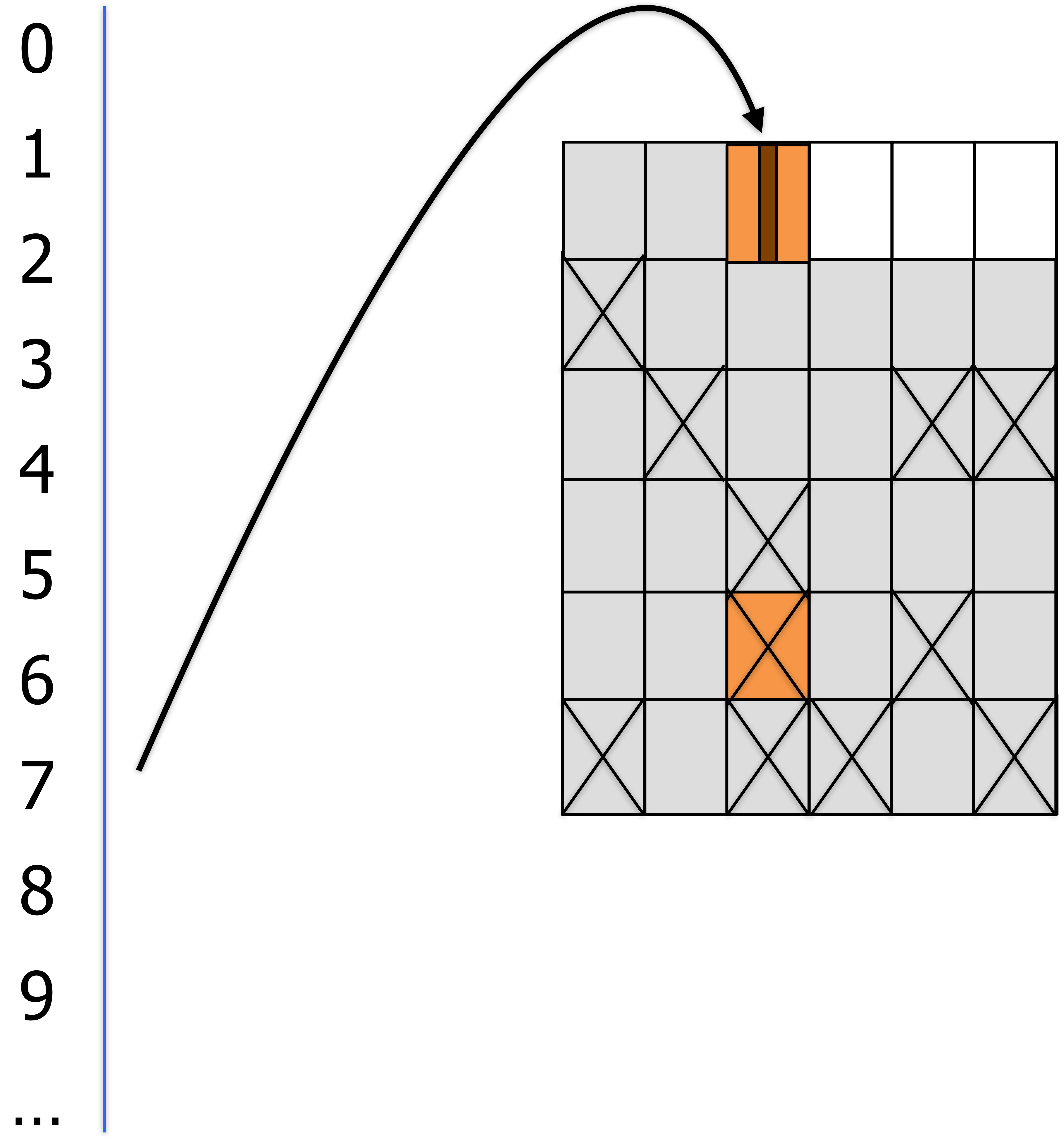
Application
Update smaller than page size



Principle 1: Avoid Small Updates

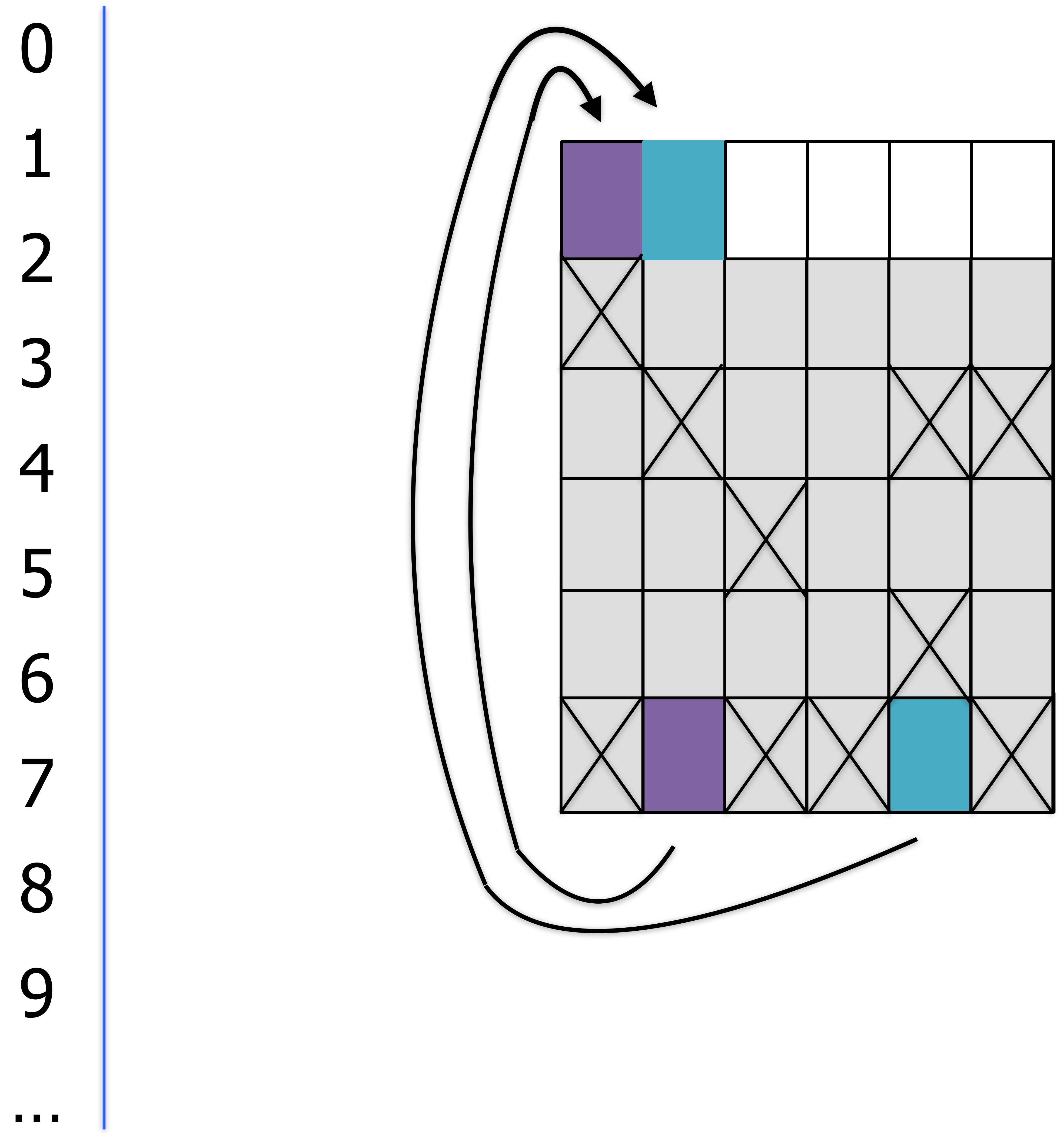
Small Updates are terrible - they force reading a whole page and rewriting it.

$$\text{Write-amplification} = \frac{\textbf{Page size}}{\textbf{Update size}}$$



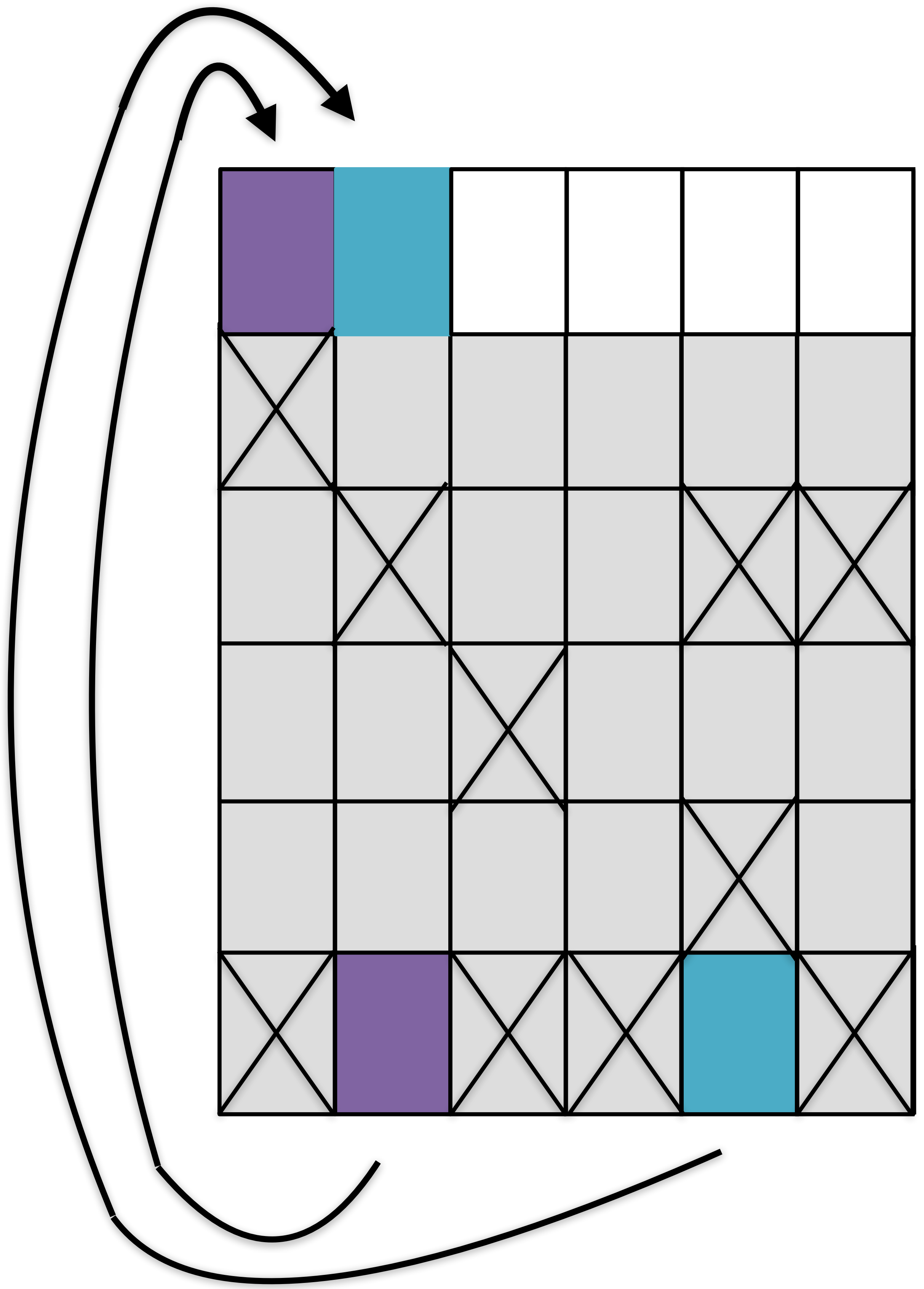
Principle 2: Avoid Random Updates

Avoid random updates, as they lead to garbage-collection overheads



Principle 2: Avoid Random Updates

Avoid random updates, as they lead to garbage-collection overheads

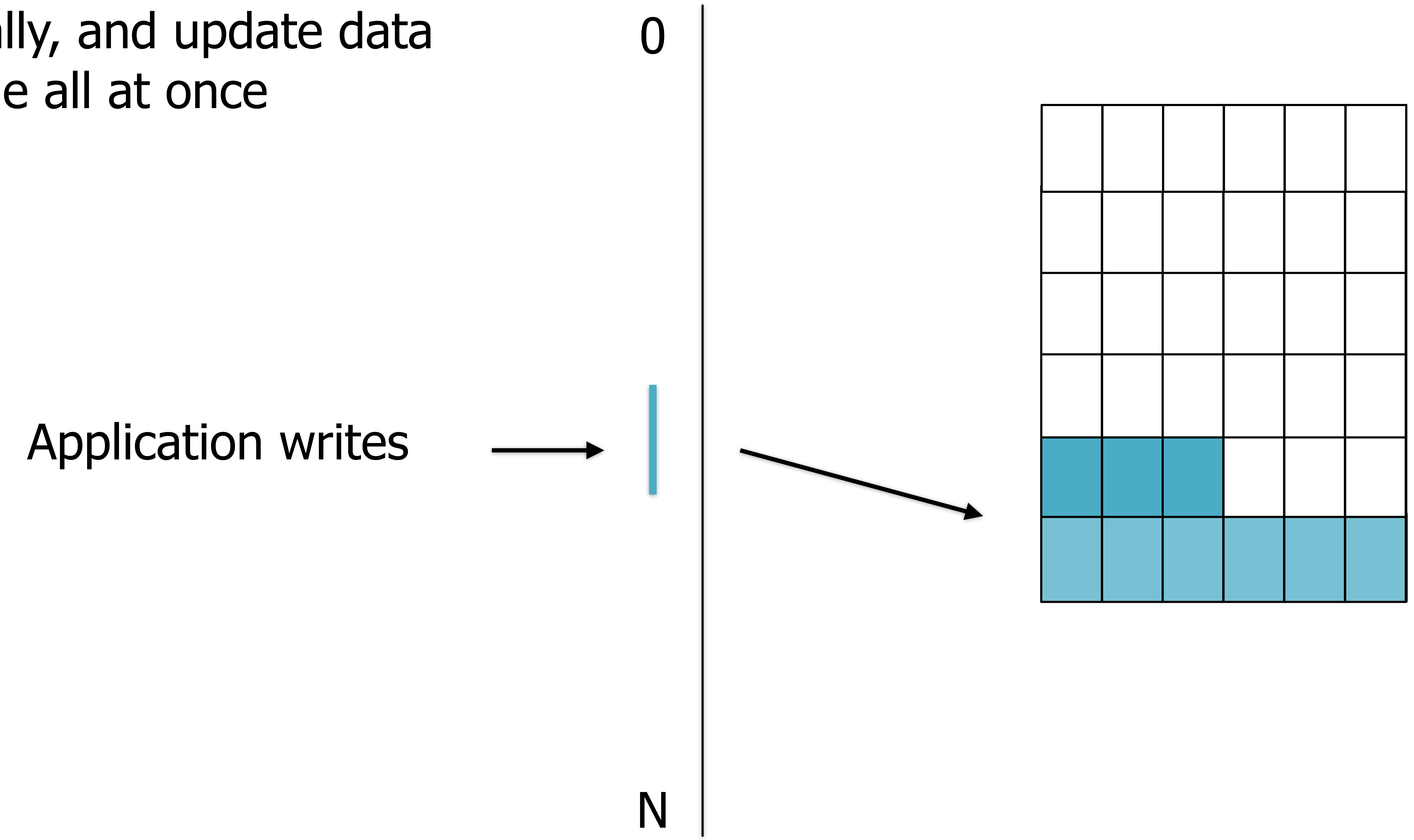


$$\text{GC WA formula} = 1 + \frac{\#\ \text{migrated}}{\#\ \text{freed}} = 1 + \frac{X}{1-X}$$

X = fraction of pages in erase unit migrated each garbage-collection operation operation

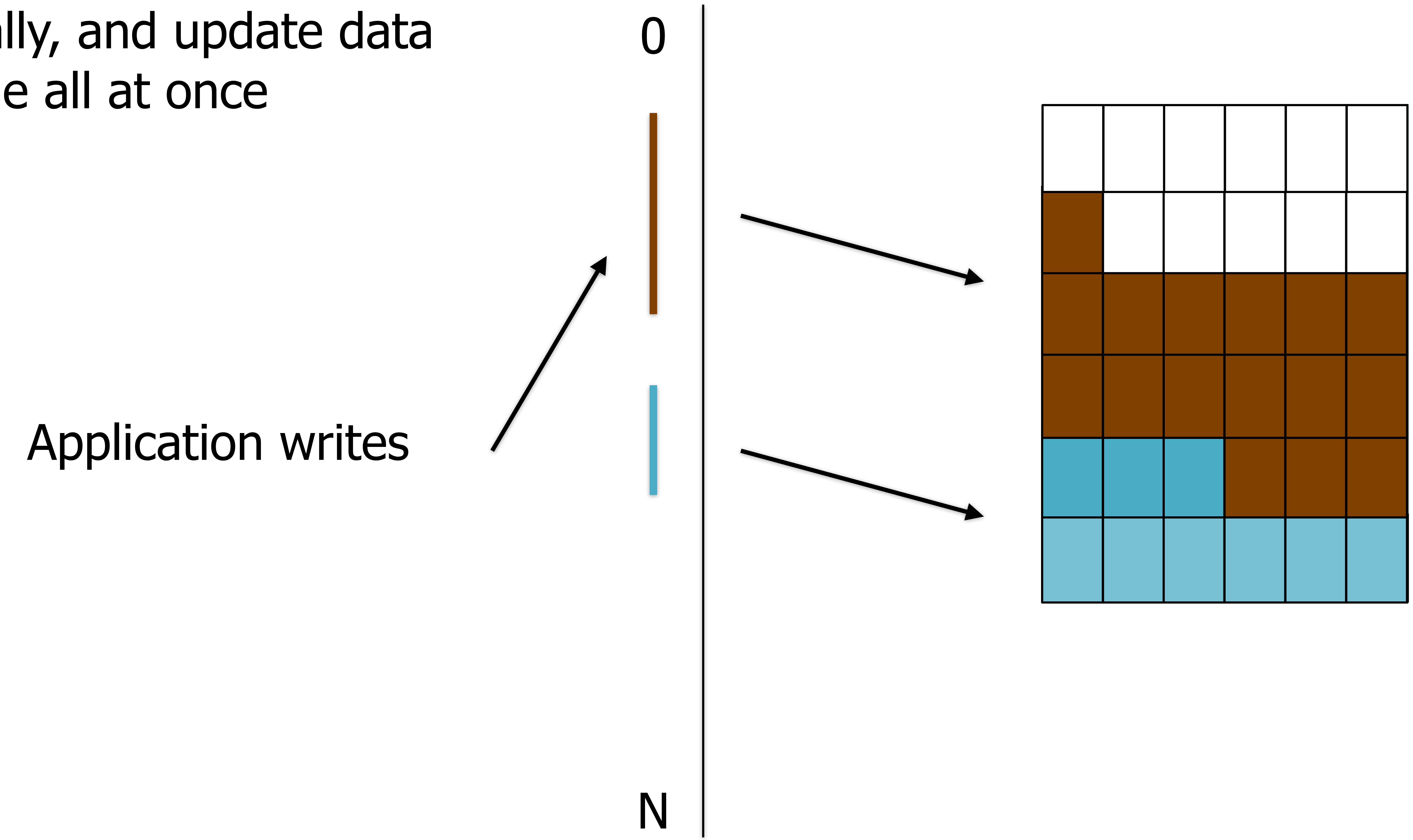
Principle 2: Avoid Random Updates

Instead write sequentially, and update data written at the same time all at once



Principle 2: Avoid Random Updates

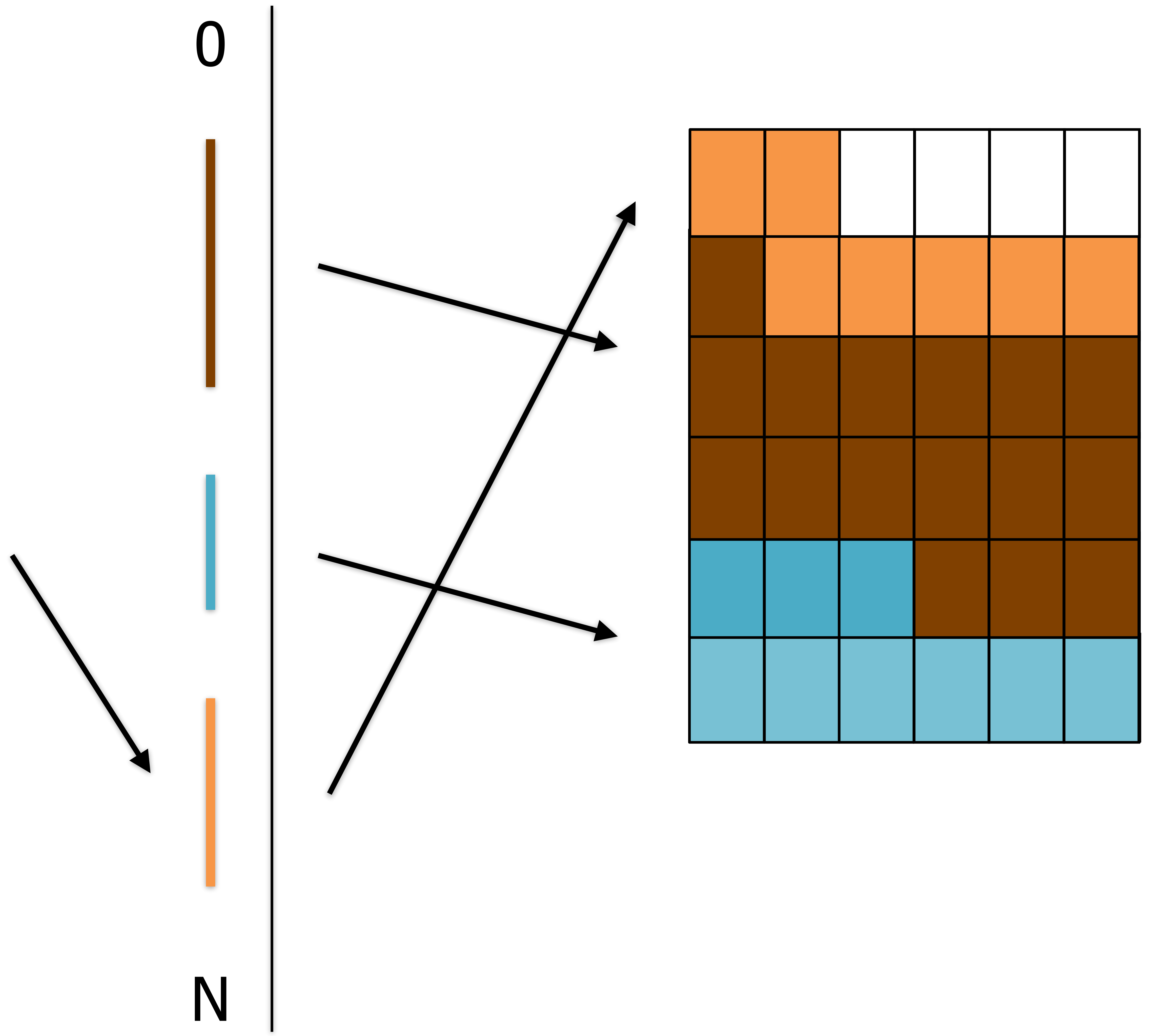
Instead write sequentially, and update data written at the same time all at once



Principle 2: Avoid Random Updates

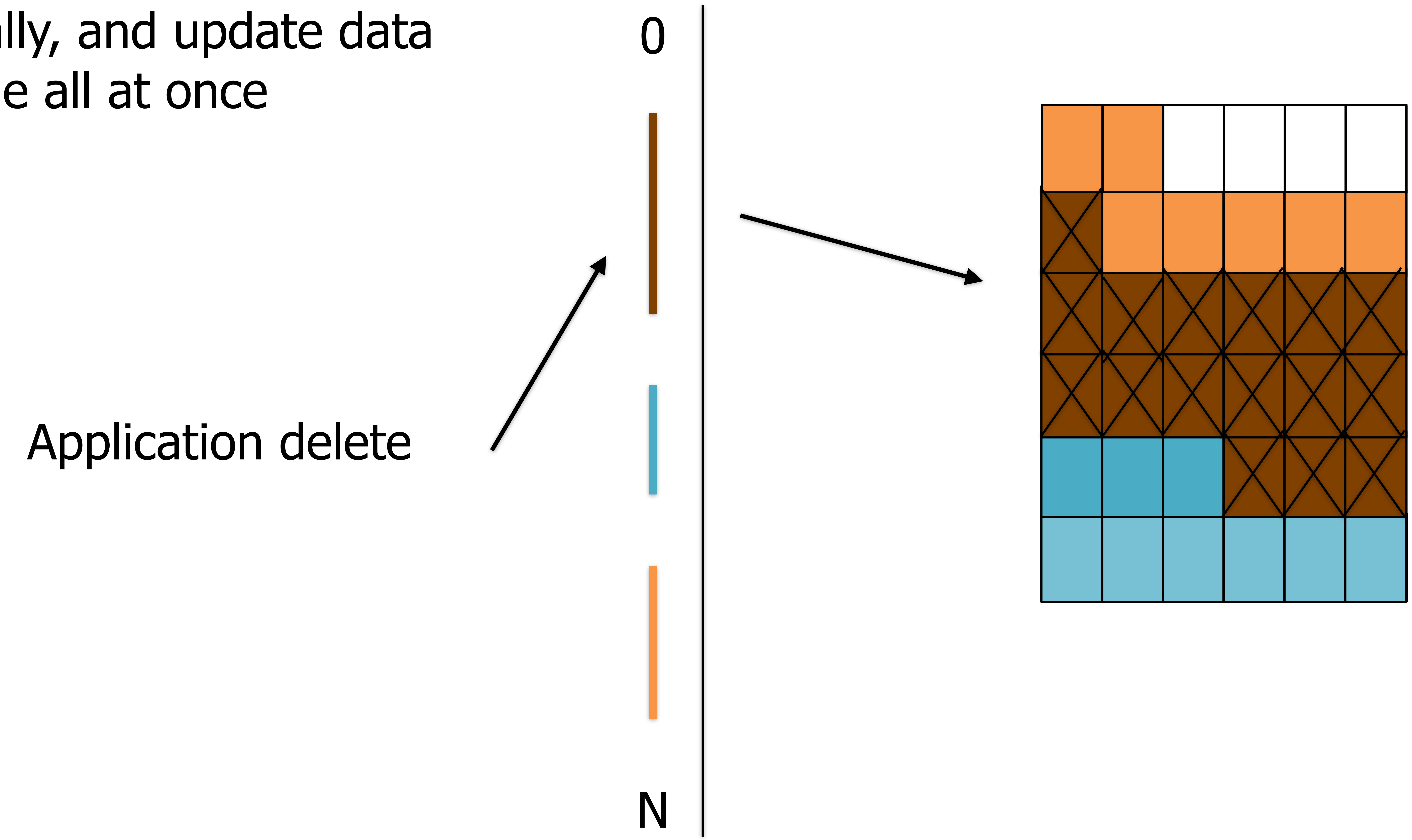
Instead write sequentially, and update data written at the same time all at once

Application writes



Principle 2: Avoid Random Updates

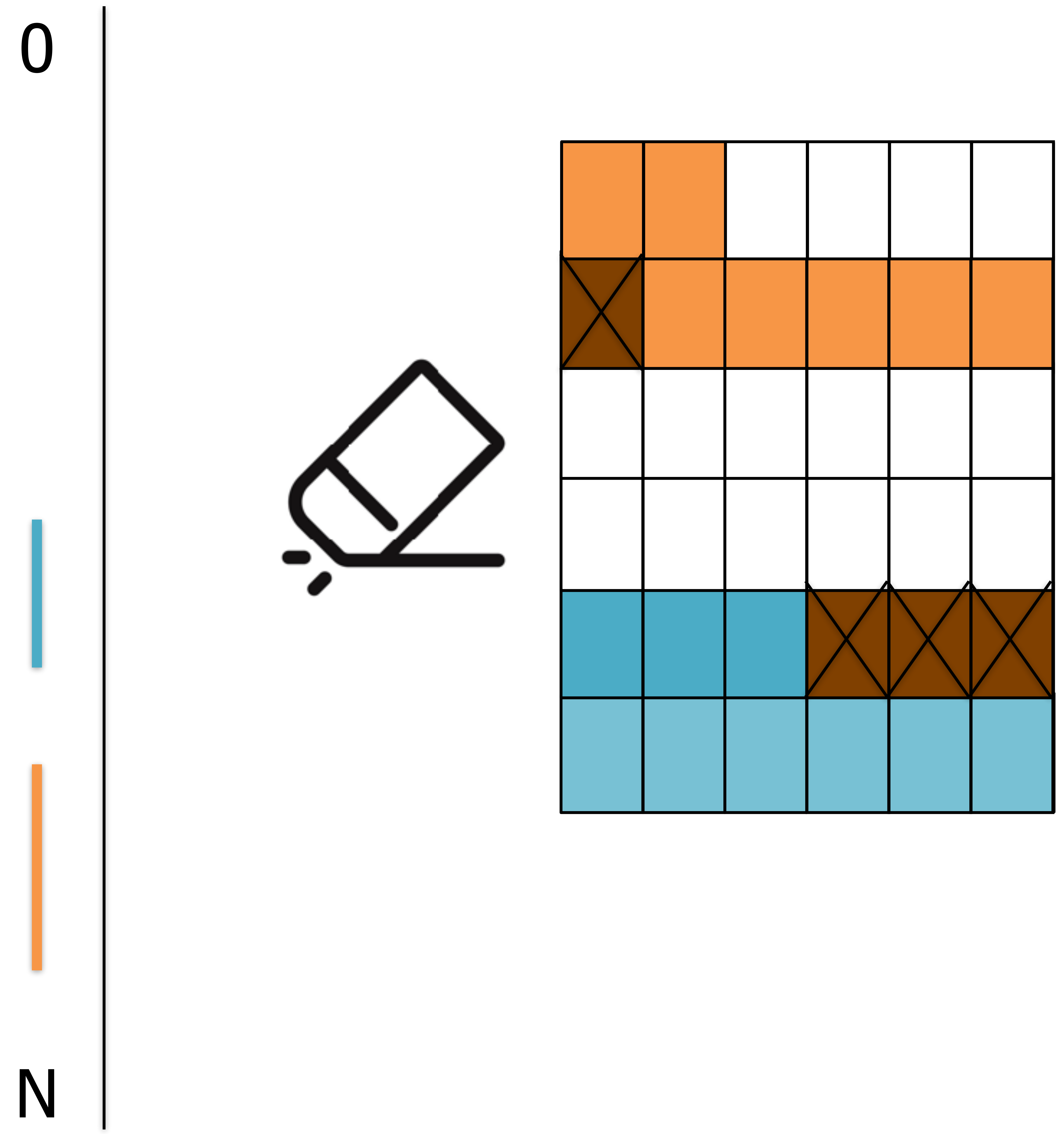
Instead write sequentially, and update data written at the same time all at once



Principle 2: Avoid Random Updates

Instead write sequentially, and update data written at the same time all at once

Free erase blocks without garbage-collection

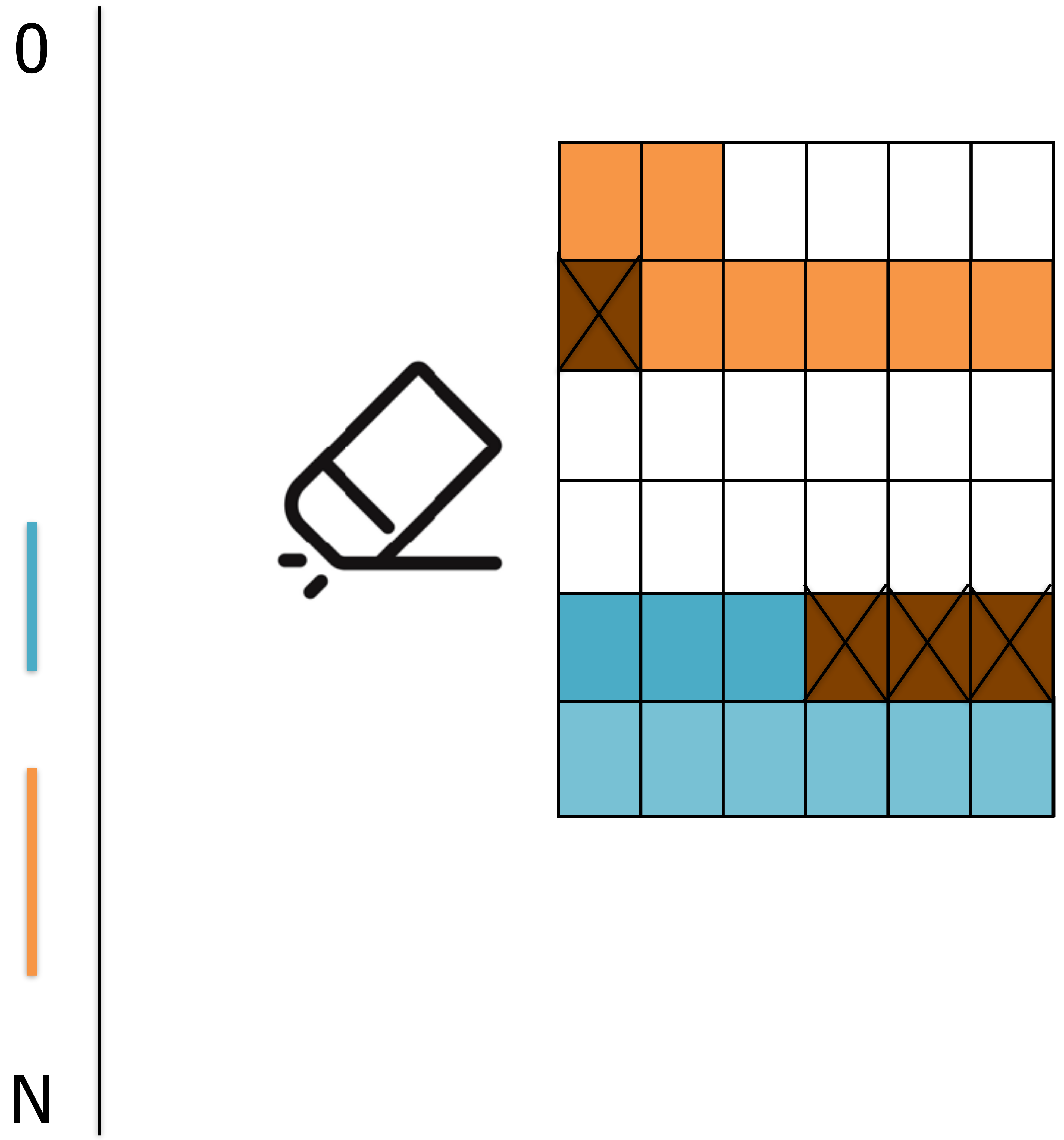


Principle 2: Avoid Random Updates

Instead write sequentially, and update data written at the same time all at once

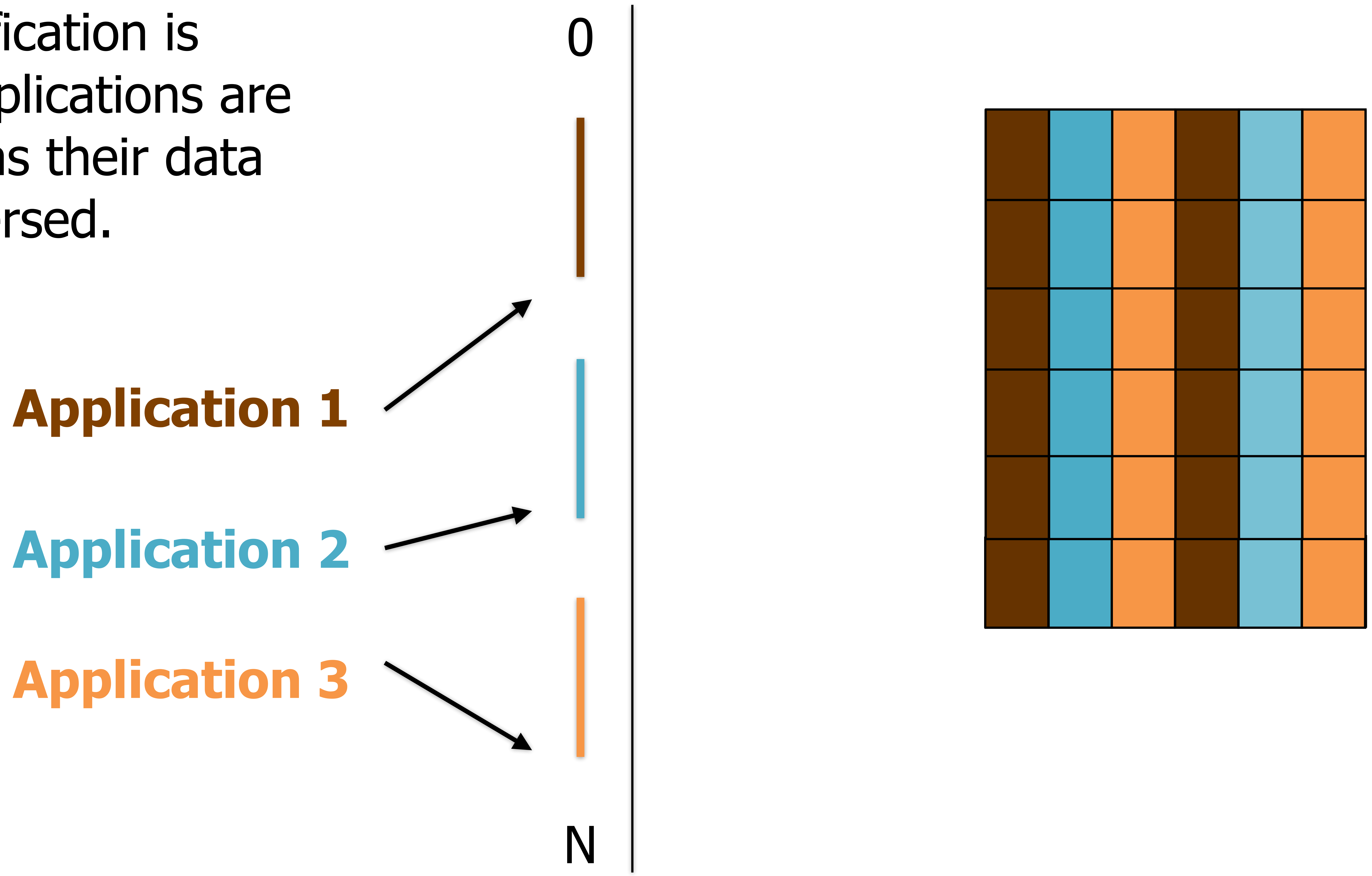
Free erase blocks without garbage-collection

Write-amplification = 1



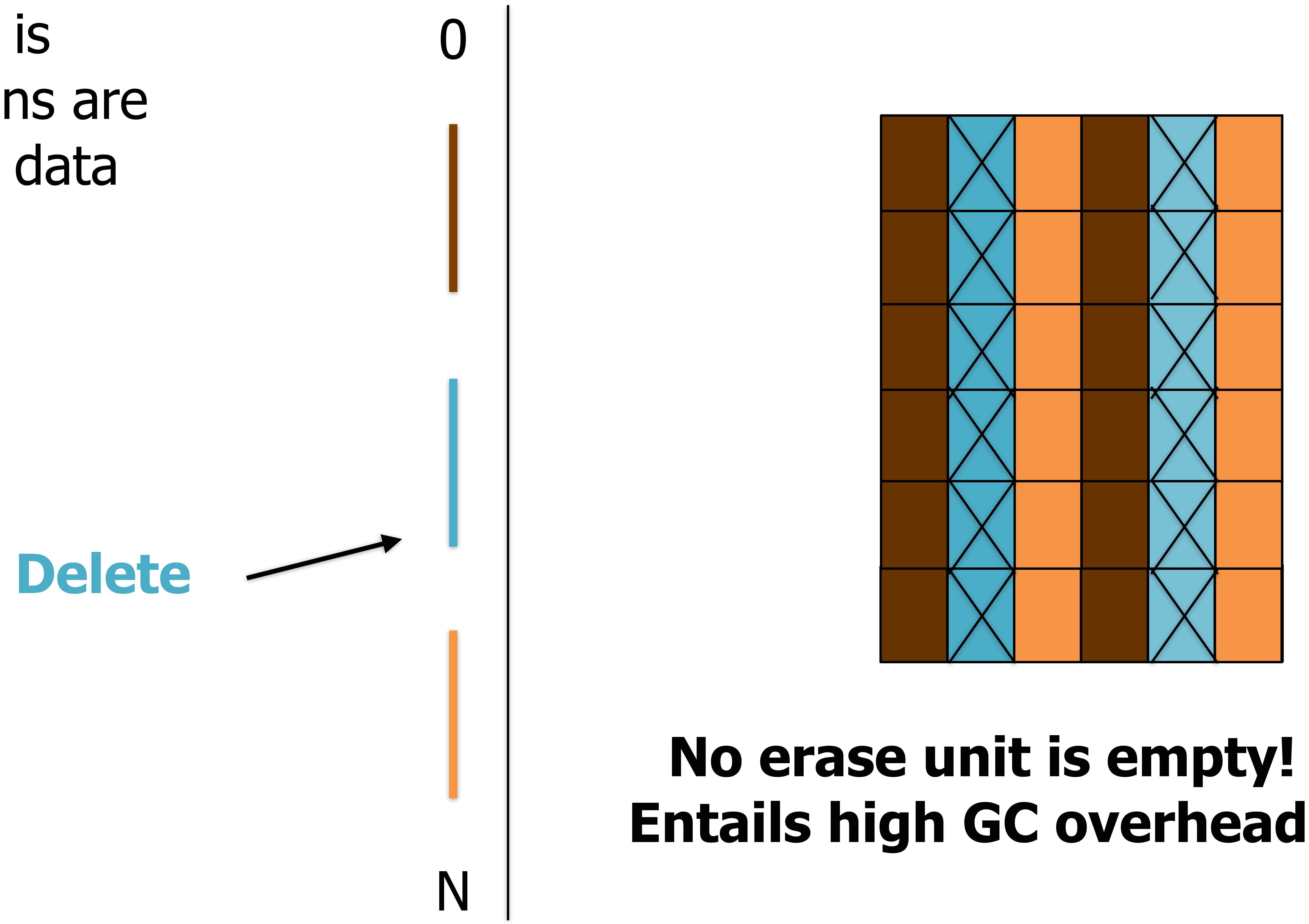
Principle 2: Avoid Random Updates

However, some write-amplification is inevitable when multiple applications are running on the same SSD, as their data becomes physically interspersed.



Principle 2: Avoid Random Updates

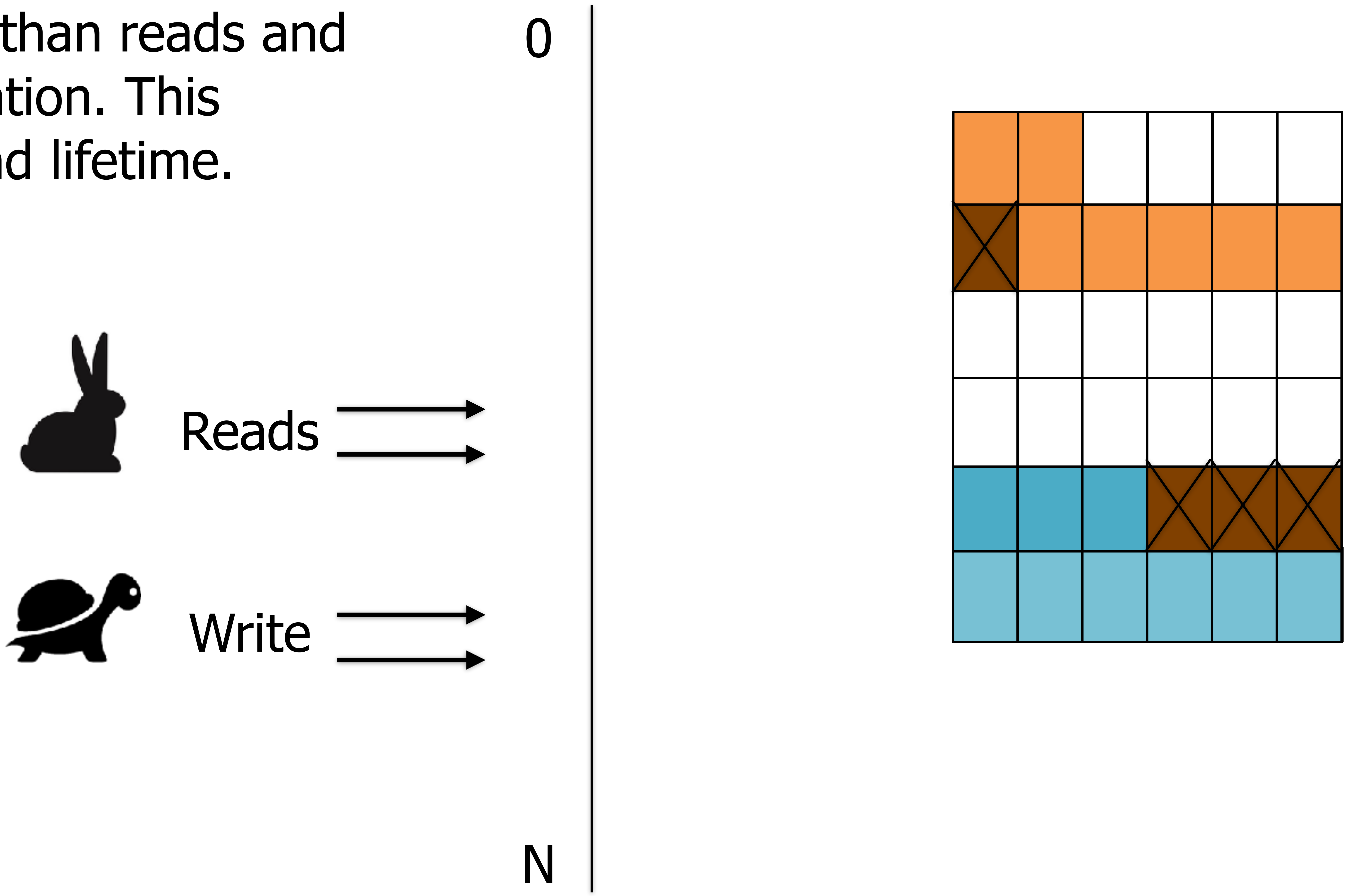
However, some write-amplification is inevitable when multiple applications are running on the same SSD, as their data becomes physically interspersed.



No erase unit is empty!
Entails high GC overheads

Principle 3: Prioritize Reads over Writes

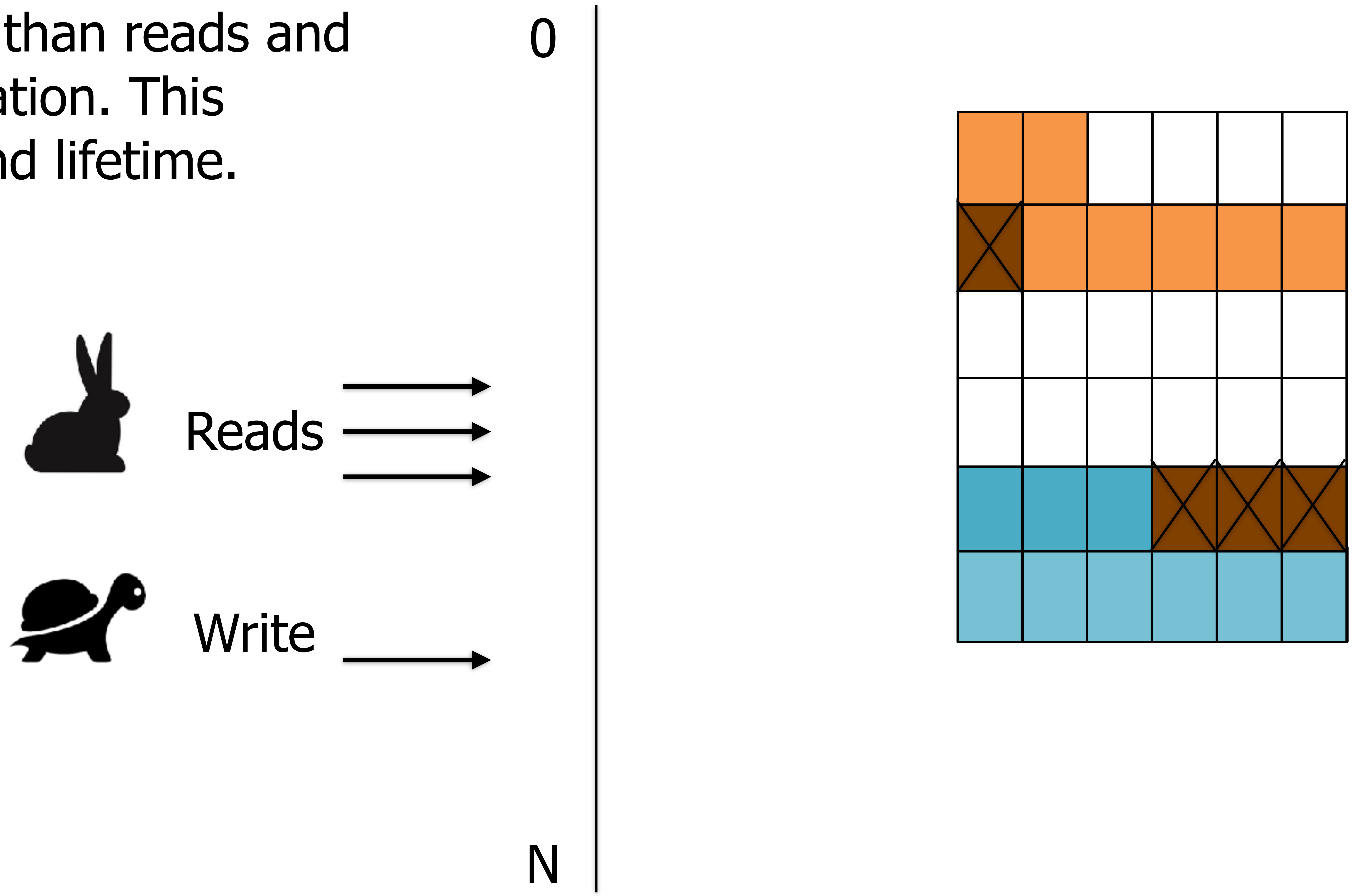
Writes take longer to execute than reads and may also entail write-amplification. This degrades SSD performance and lifetime.



Principle 3: Prioritize Reads over Writes

Writes take longer to execute than reads and may also entail write-amplification. This degrades SSD performance and lifetime.

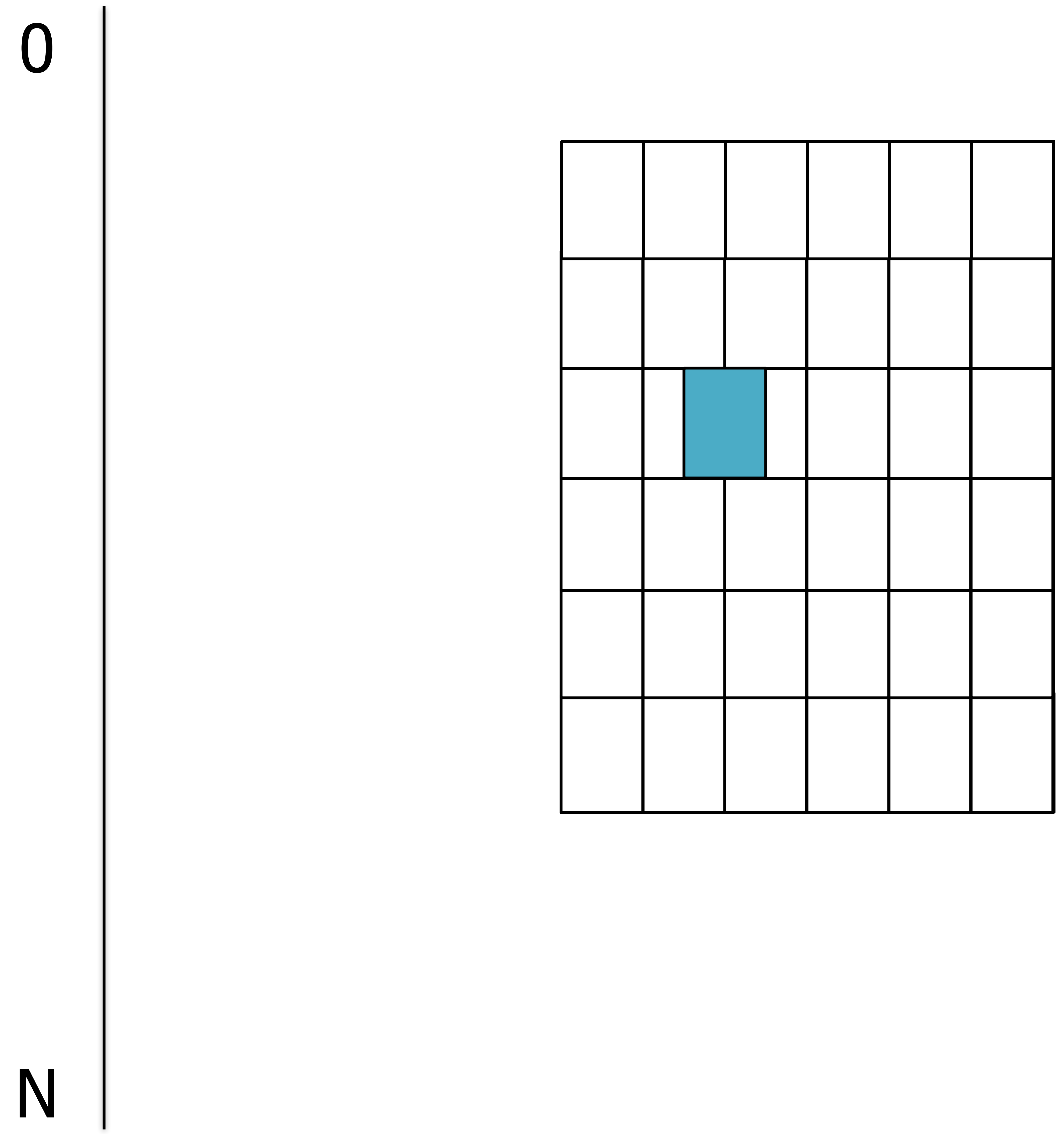
Study data structures that entail fewer writes at the expense of more reads.



Principle 4: Reads/Writes should be page aligned

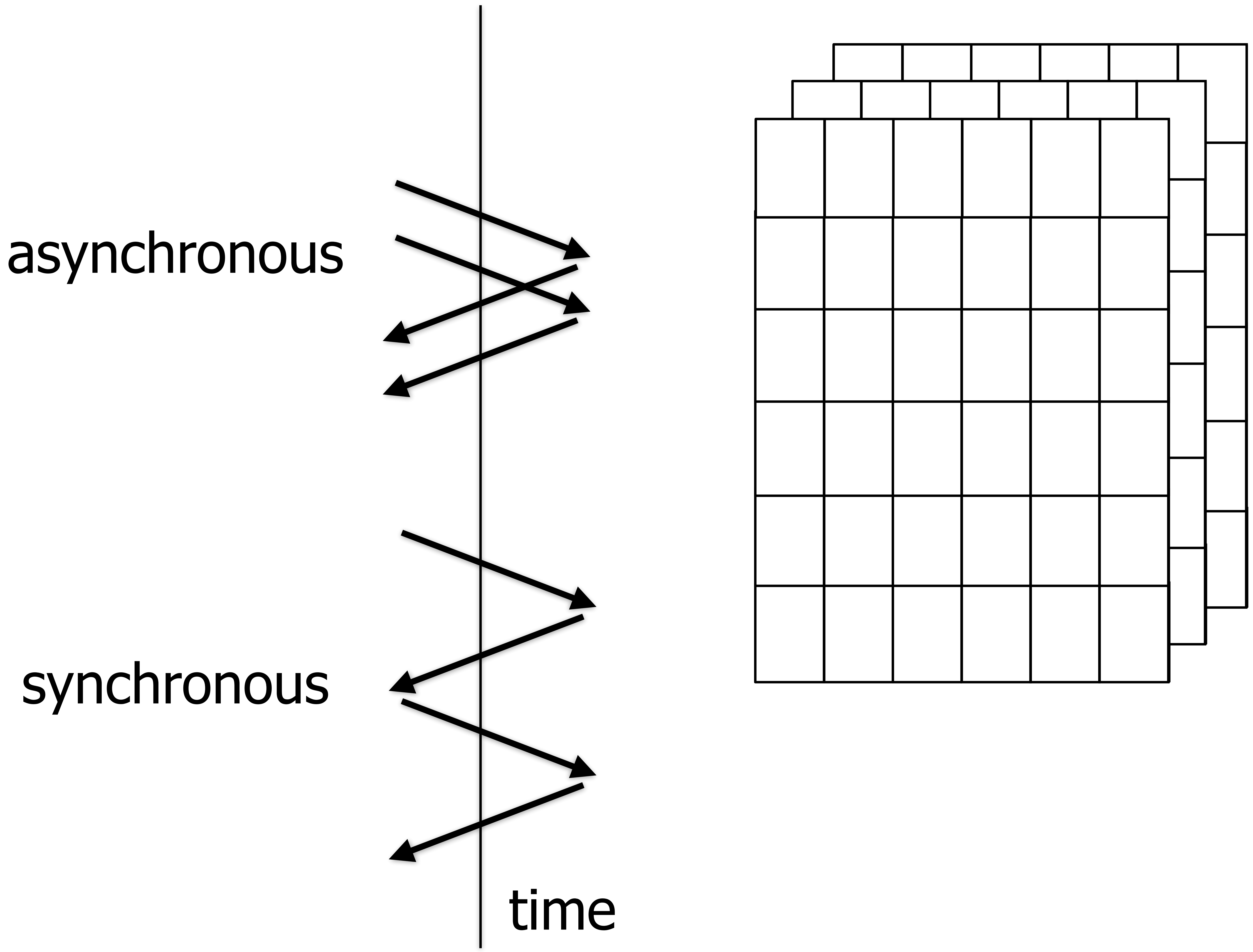
Reading a misaligned page triggers 2 flash reads

Updating a misaligned page triggers 2 flash reads and 2 flash writes

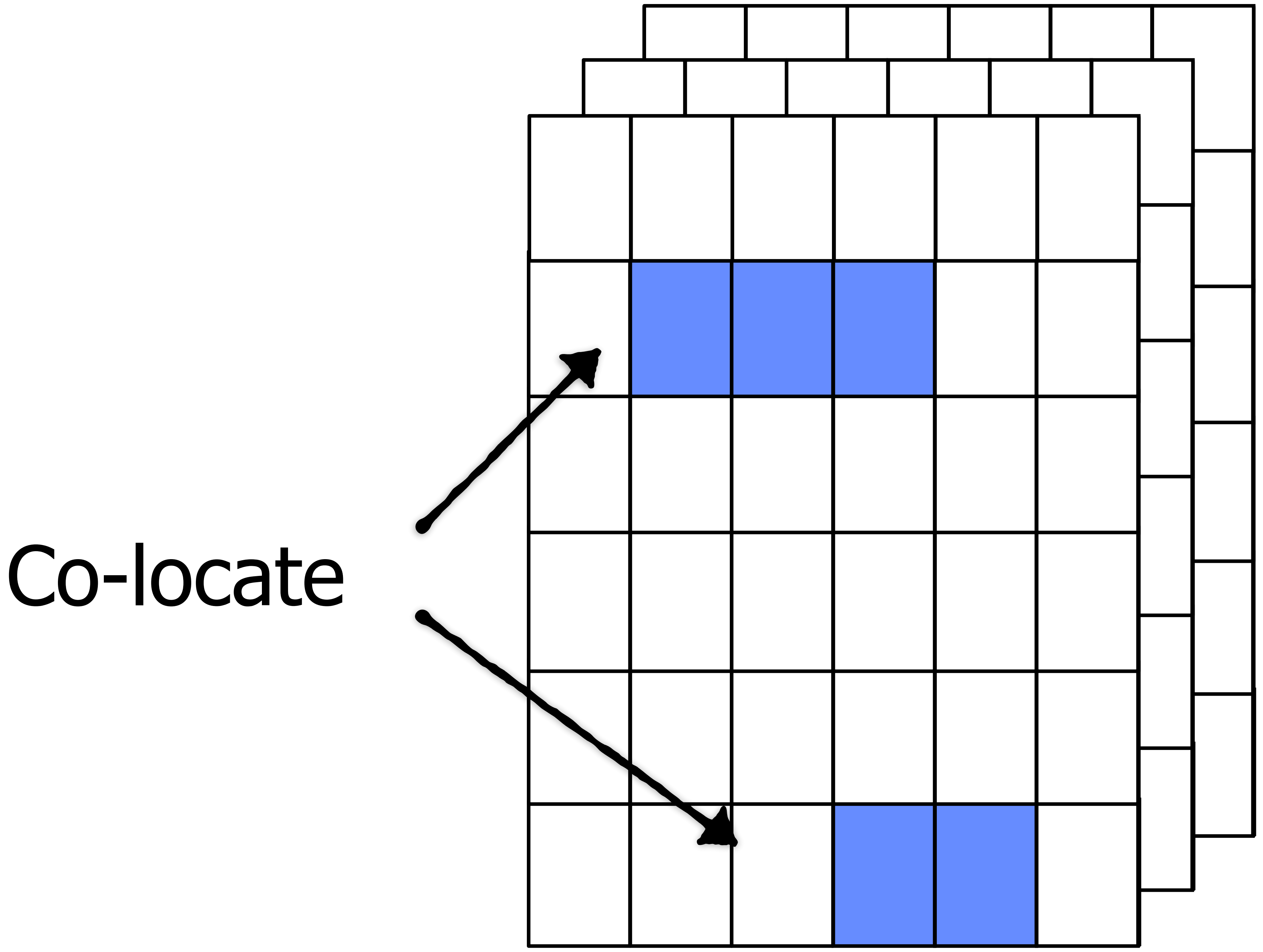


Principle 5: Asynchronous I/Os for the win

Internal parallelism of SSD makes asynchronous I/O faster than synchronous I/O



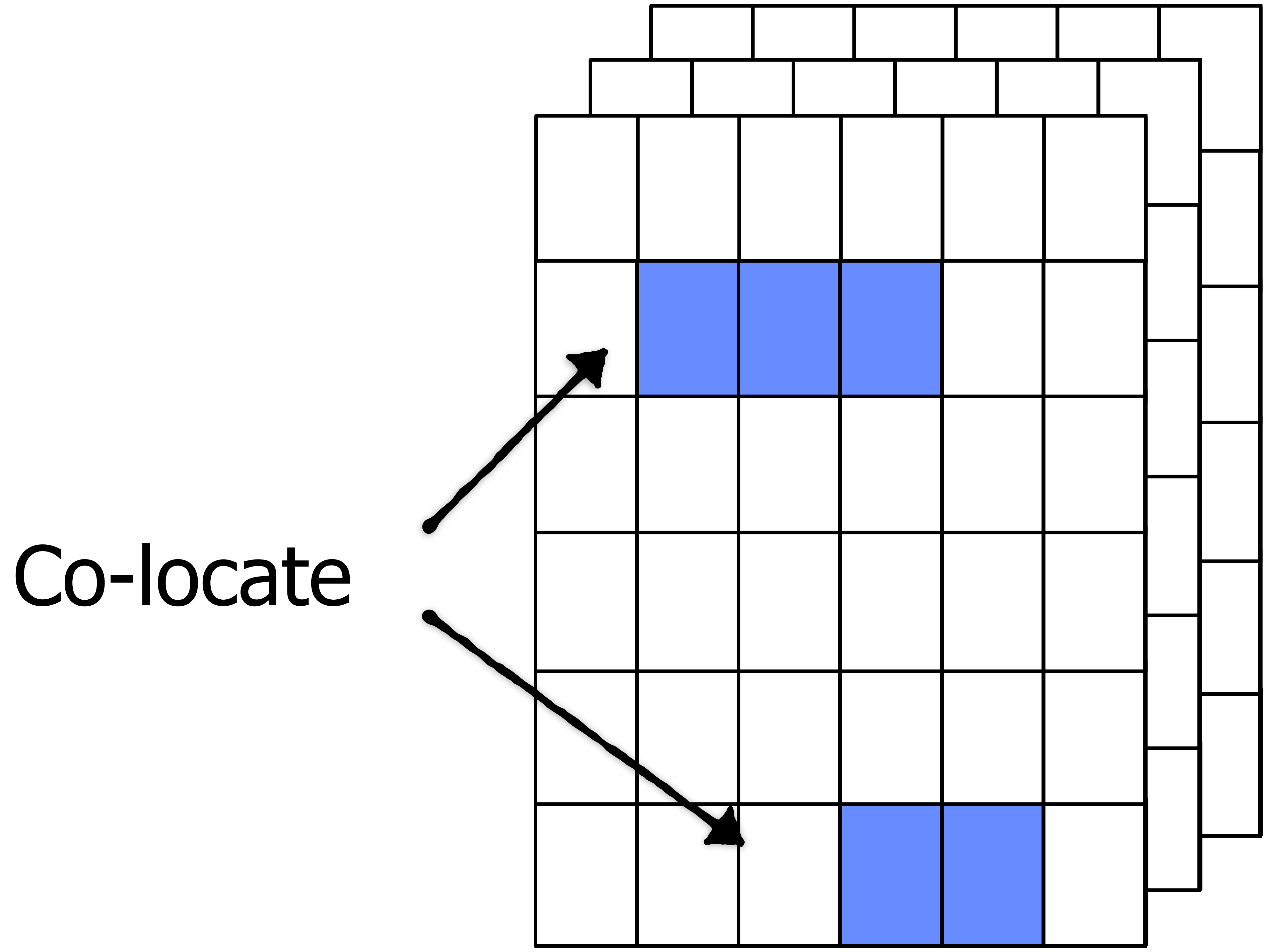
Principle 6: Defragmentation on SSDs?



Principle 6: Defragmentation on SSDs is a Bad Idea

Random read I/Os are fast, so relocating parts of a file to be close does not significantly improve read speed.

On the other hand, it contributes to write-amplification and consumes the device's lifetime.



02

Week 2: Storage Devices

This week will take a deep look at the memory hierarchy. As we will see throughout the course, the properties of the memory hierarchy give rise to how modern databases are architected.

For information on storage, read Chapter 8 Part 8.1, and Chapter 9 Parts 9.1 and 9.2.

As you will notice, the textbook was written before SSDs became popular. I have written some background on SSDs here. Please study this carefully.

You may also skim the following article.

02

Week 2: Storage Devices

This week will take a deep look at the memory hierarchy. As we will see throughout the course, the properties of the memory hierarchy give rise to how modern databases are architected.

For information on storage, read Chapter 8 Part 8.1, and Chapter 9 Parts 9.1 and 9.2.

As you will notice, the textbook was written before SSDs became popular. I have written some background on SSDs here. Please study this carefully.

You may also skim the following article.

Important reading

Reasoning About SSD Write-Amplification - CSC443H1 Database System Technology – Niv Dayan

Pages and Erase Units. An SSD based on NAND flash memory consists of multiple pages, each approximately 4KB. Reads and writes take place at the granularity of pages. Pages are organized into erase units, and they must be written sequentially within an erase unit (consisting of hundreds to thousands of pages). To update a physical page, we must erase the entire erase unit that the page belongs to. Erase units have a finite lifetime: they can only be erased a certain number of times before they become too error-prone to store data reliably.

Out-of-Place Updates. SSDs are therefore designed to prevent having to erase and rewrite an entire erase unit every time we want to update the contents of an individual page. It does this by updating pages out-of-place. Specifically, it uses some metadata to mark the original version of the page as invalid, and it writes the new version of the page on some erase unit with free space. The SSD maintains a mapping table from the logical address of each phase to its physical address within the SSD. Overall, each page can have one of three states: free, valid, or invalid.

Garbage Collection. As updates take place, more physical pages within the SSD get marked as invalid. Eventually, as the SSD runs out of free space, we must reclaim space taken up by invalid pages to accommodate more updates from the application. The SSD does this by performing

…

Different types of flash devices

All use flash memory and (roughly) work as described above

NVME SSD

SATA SSD

USB stick

SD card

← **Faster**

Cheaper →

Phase-Change memory	SSD	disk	Shingled disks	Tape	DNA
Discontinued			**New**	**Archival**	**Research**

← Faster

Cheaper →

On Friday we’ll look at RAID