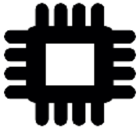
Storage Management

Database System Technology - Lecture 2

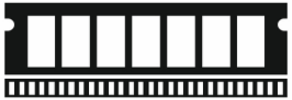
Niv Dayan

Different Physical Technologies



CPU caches

SRAM



memory

DRAM

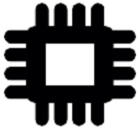
SSD

NAND flash

disk

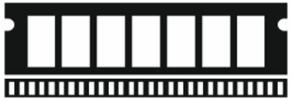
Magnetic Material

Different Physical Technologies



CPU caches

SRAM



memory

DRAM

SSD

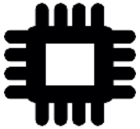
NAND flash

disk

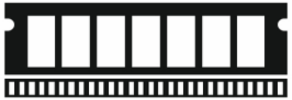
Magnetic Material

The properties of these devices motivate much of how modern database systems are built

The memory Hierarchy



CPU caches



memory

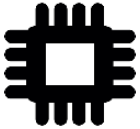
SSD

disk

Expensive & fast

Slow & cheap

Volatility - Does data stay when power is off?



CPU caches

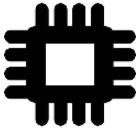
memory

SSD

disk

Data disappears

Data Remains



CPU caches

memory

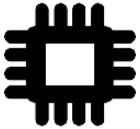
Data is brought here for processing

SSD

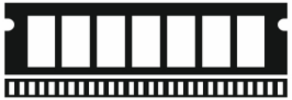
disk

Data resides here

Access granularity



CPU caches



memory

SSD

disk

Byte-addressable
64-128 B

Block-addressable
4-16 KB

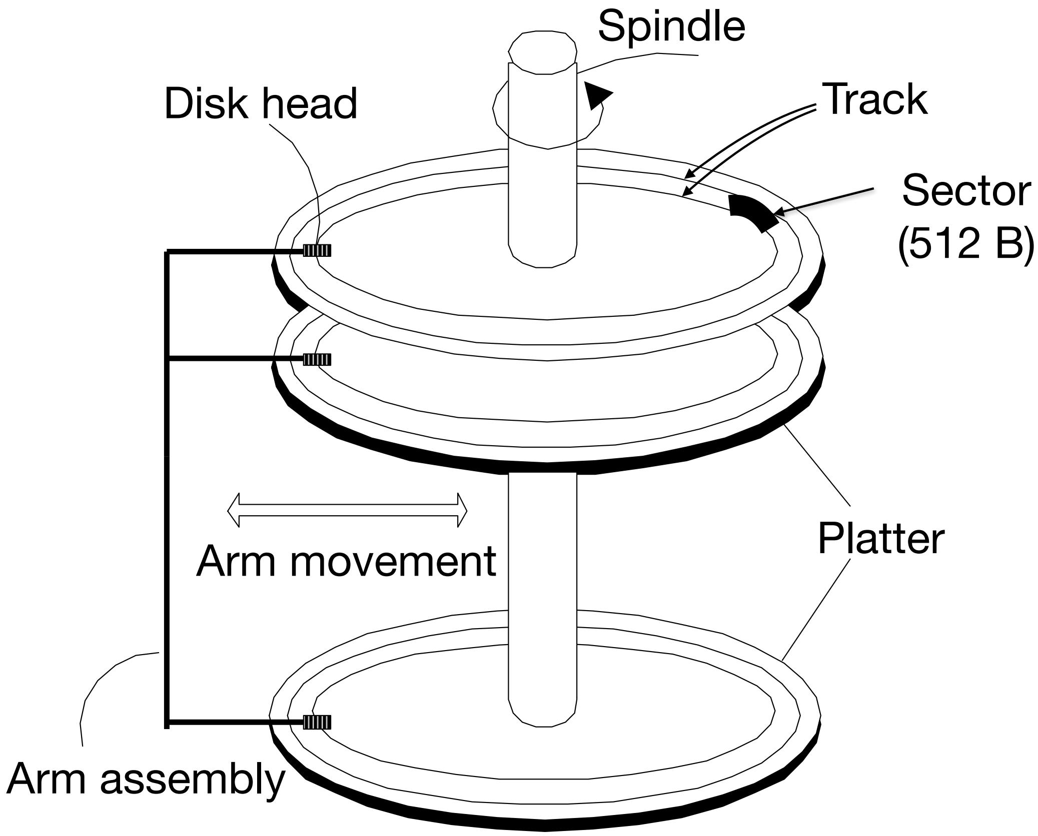
disk

The platters spin continuously (e.g., 7200 rps).

The arm assembly is moved in or out to position a head on a desired track.

Only one head reads/writes at any one time.

Data is read/written in multiples of sectors



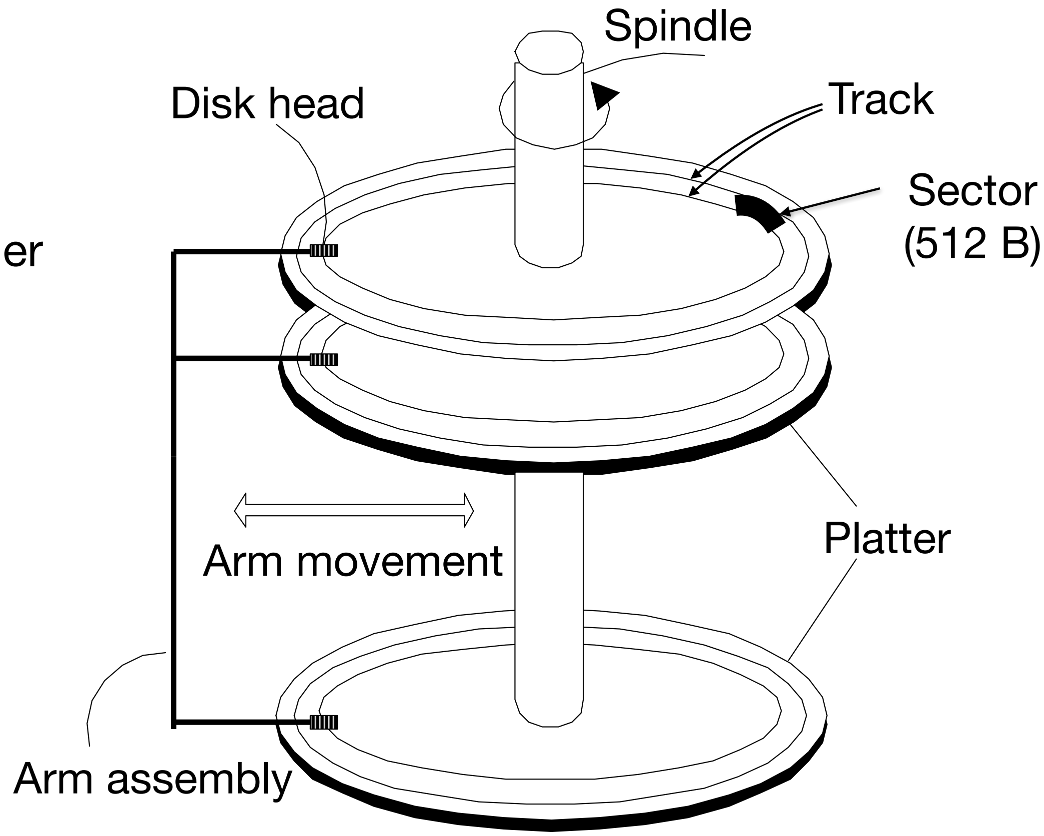
disk: access times

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)

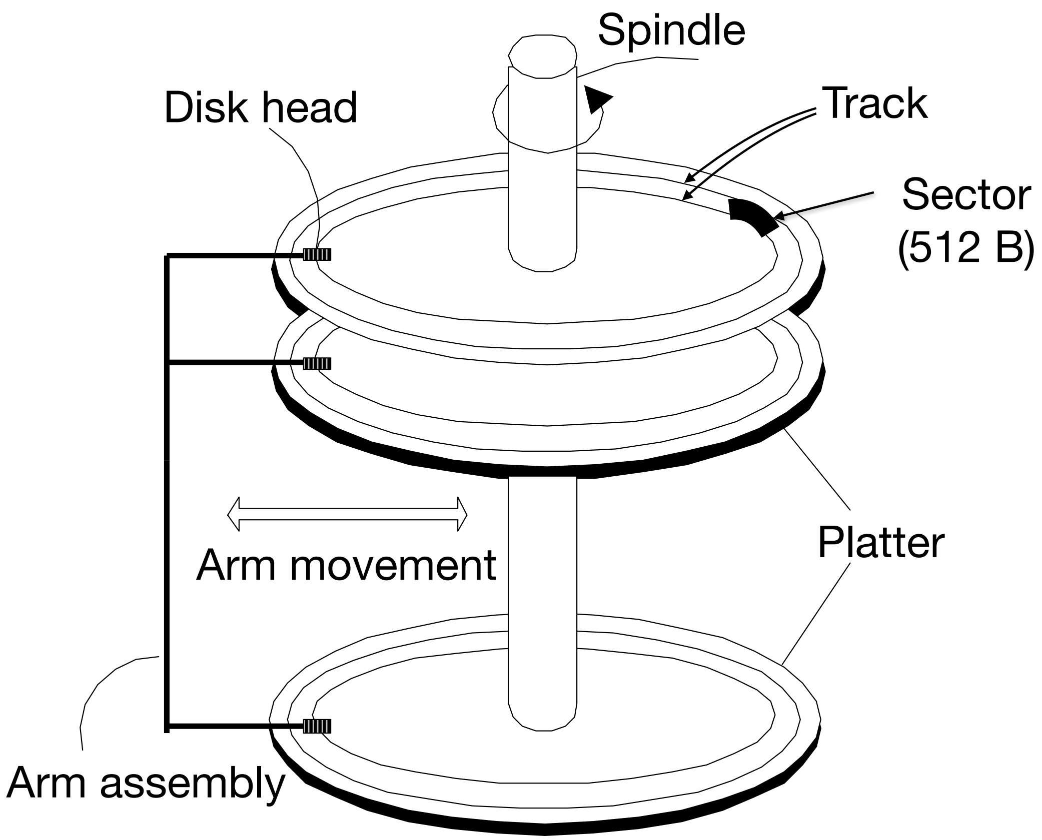
seek/rotational delays dominate



disk layout principles

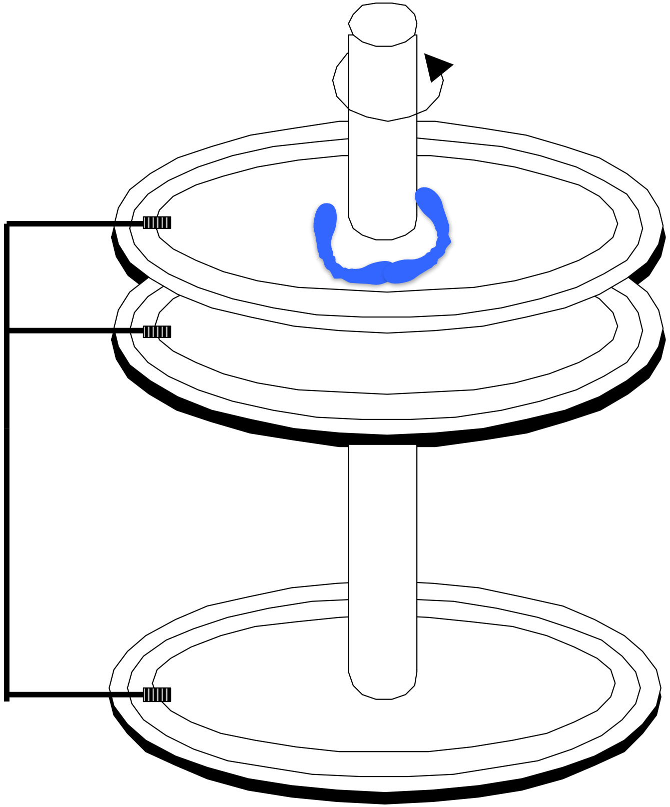
(1) Small random reads/writes are slow

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



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.

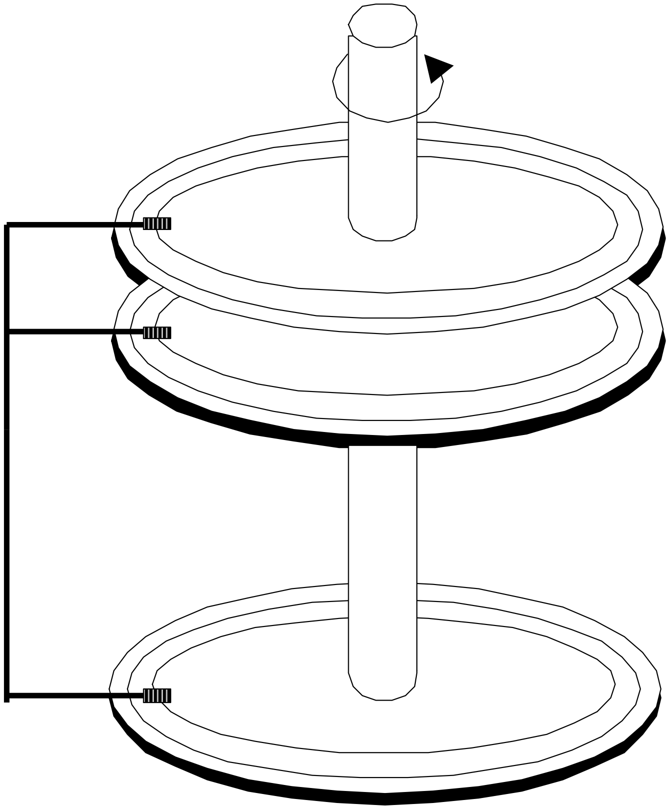
Databases do this too, as we'll see.

disk: Block device interface

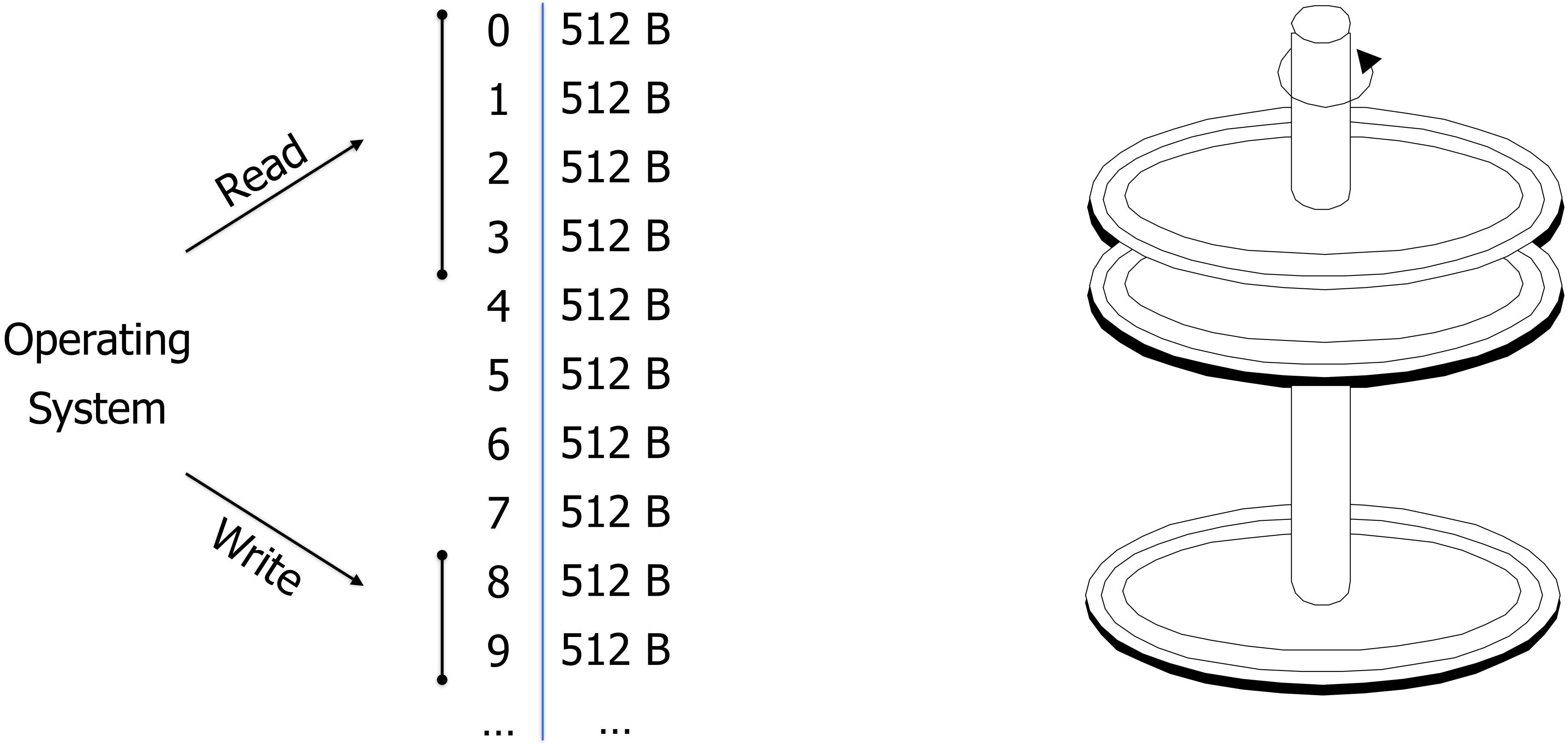
Sector addresses

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
...	...

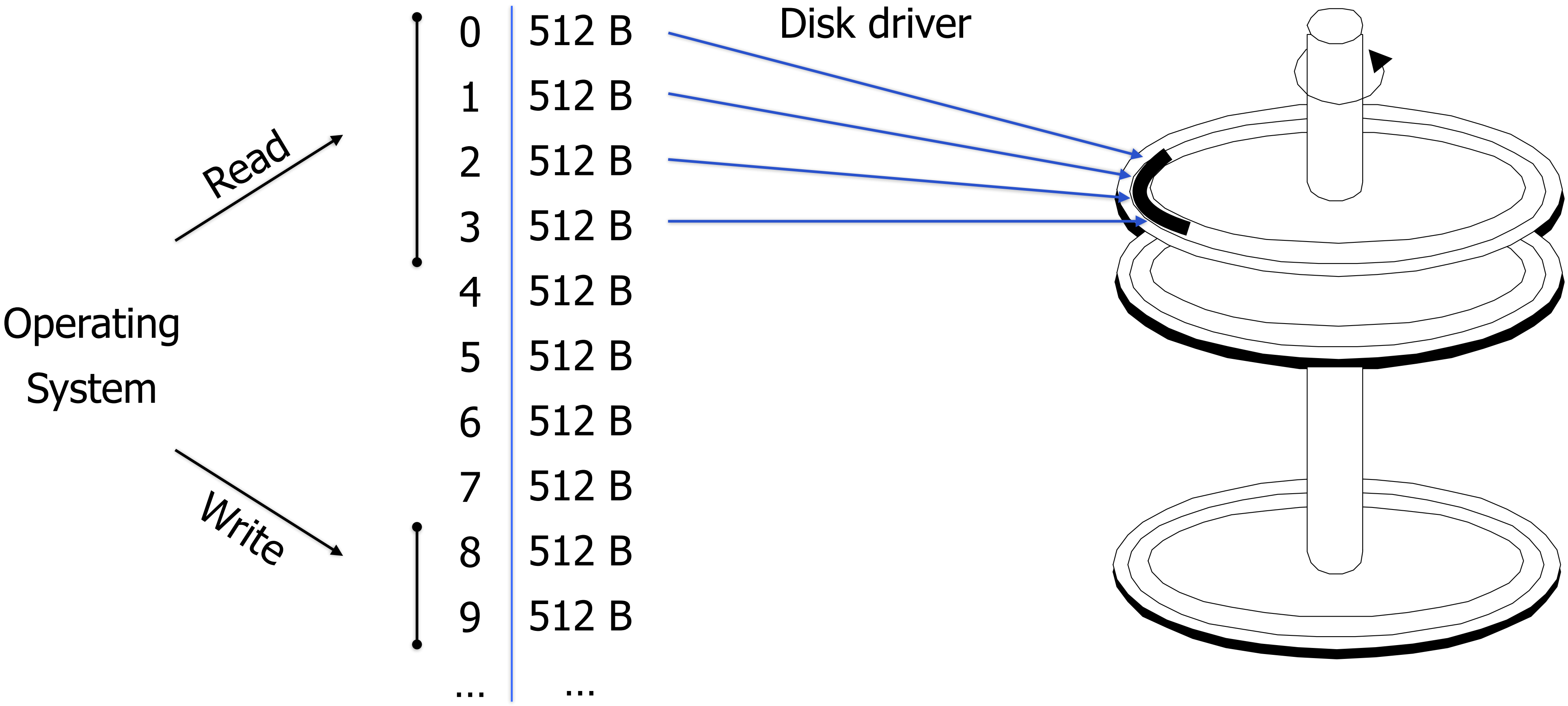
Payloads



disk: Block device interface



disk: Block device interface



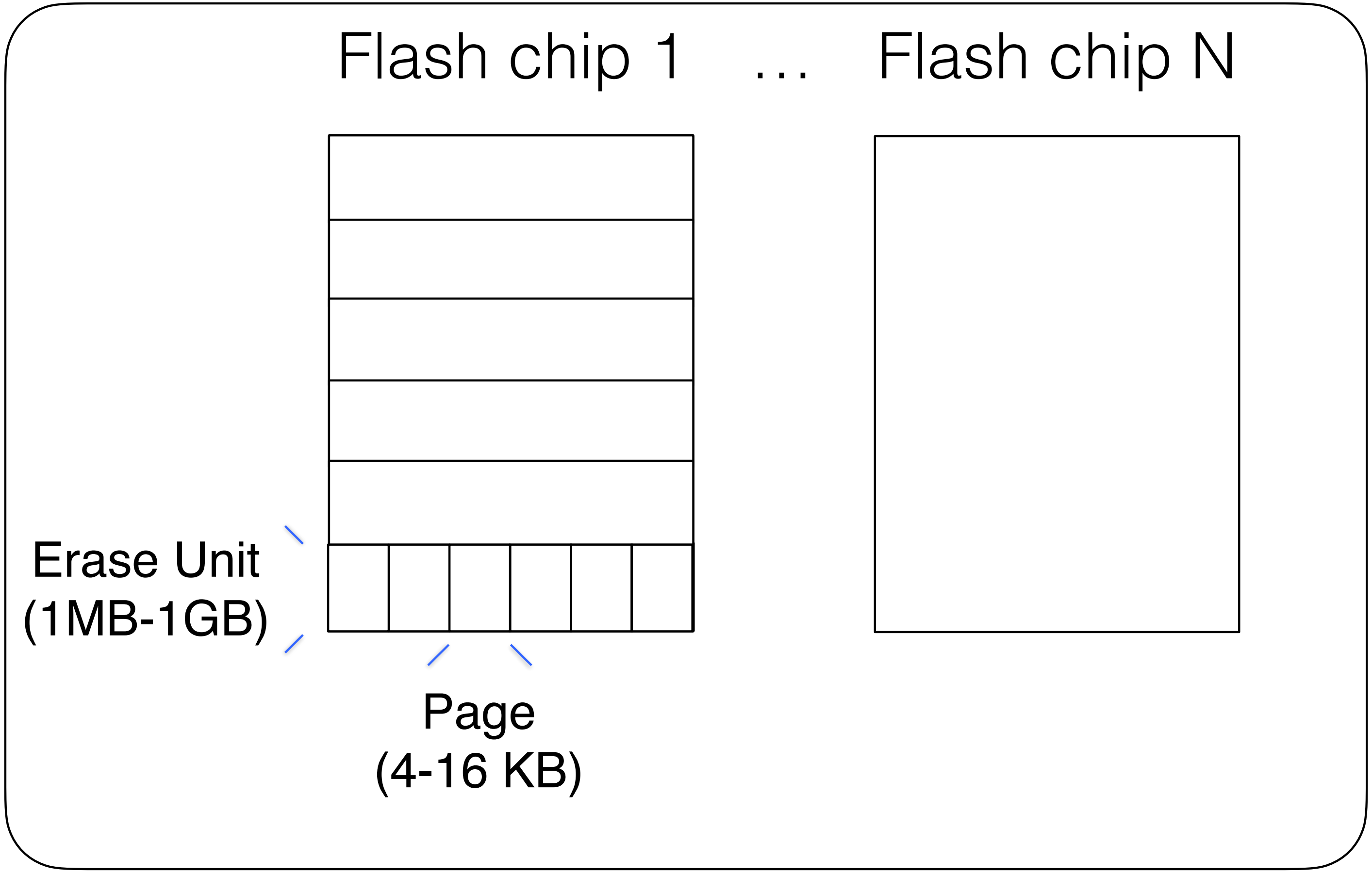
SSD

Became mainstream around 2005

disk

Traditional storage medium for decades

SSD



A page is the minimum read/write unit

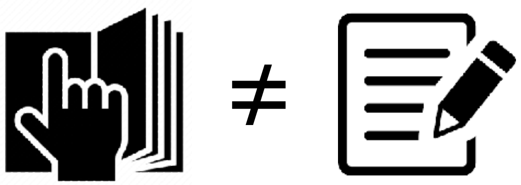
Pages must be written sequentially in an erase unit

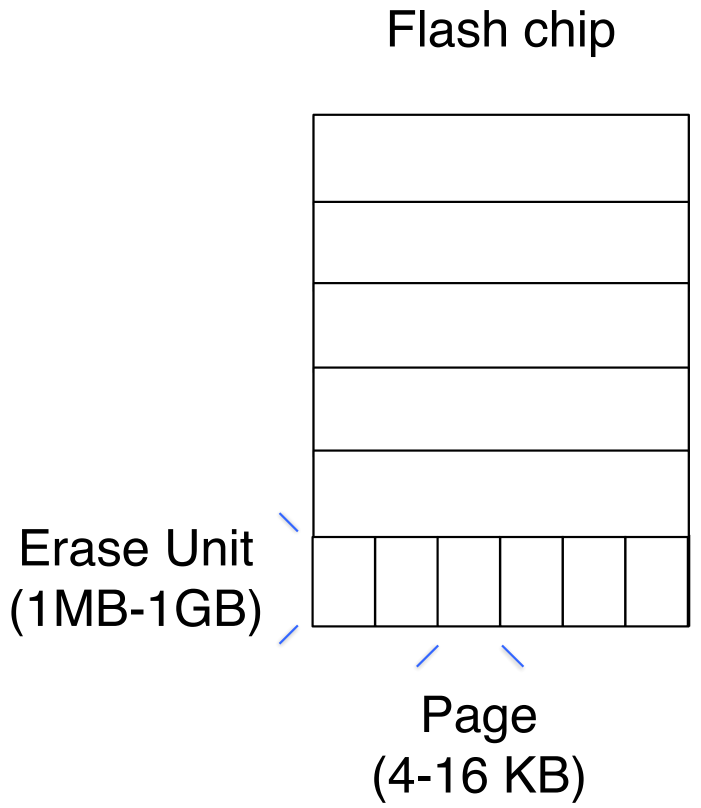
All data in an erase unit is erased at the same time

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?

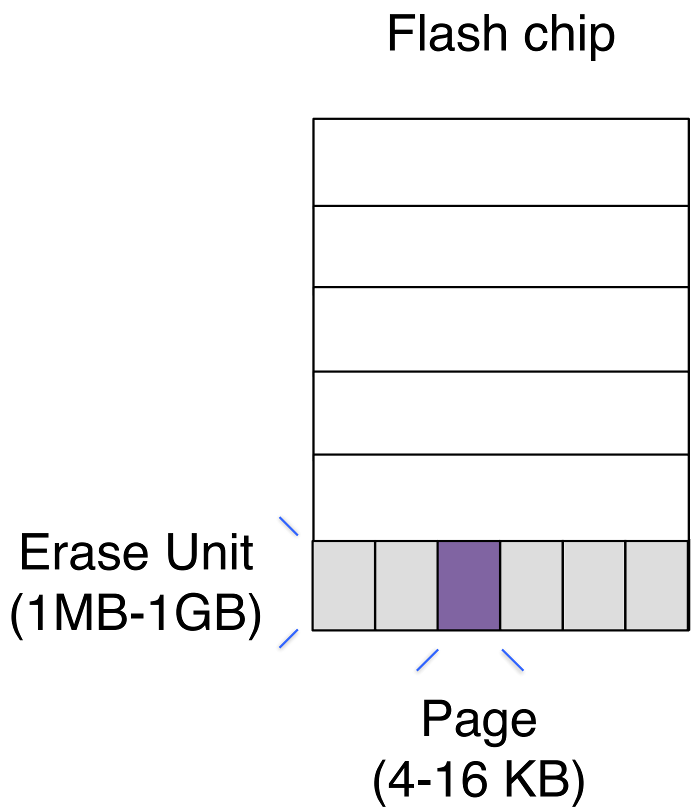
Read & rewrite the entire erase block

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

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

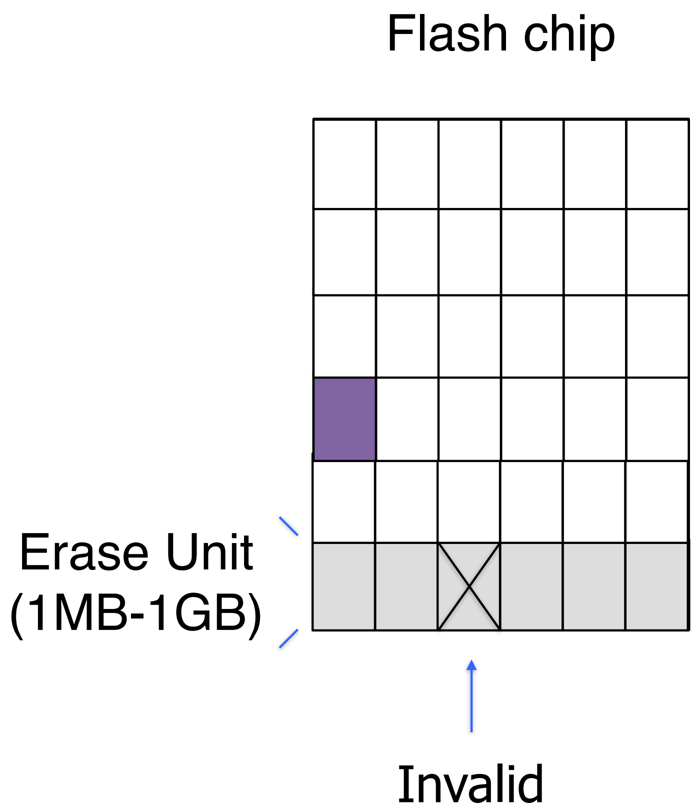


Updating a Page Out-of-Place

Better solution?

Copy the page to an erase unit with free space

Mark the original page as invalid (using a bitmap)



Garbage-Collection

Eventually many invalid pages accumulate.

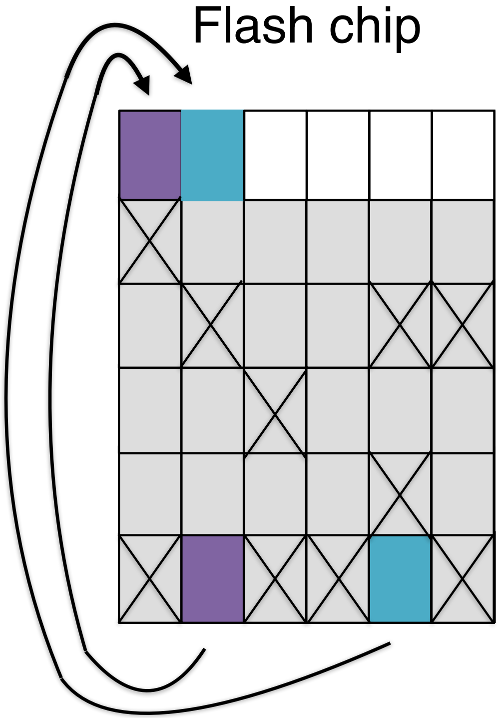
How do we reclaim space to support more writes?

Find the erase unit with the least live data left.

Copy it to an erase unit with free space.

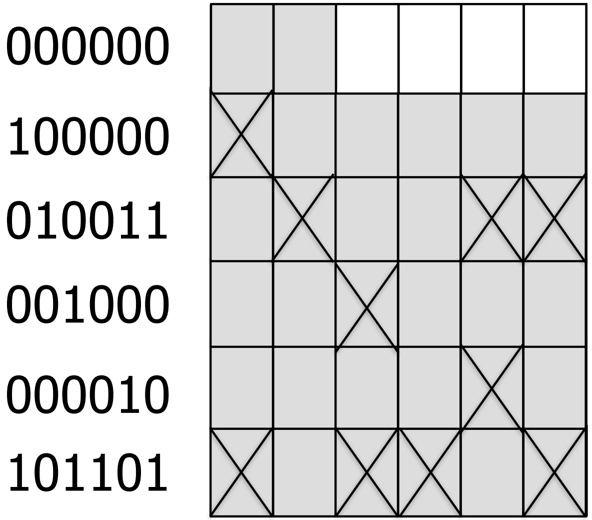
Erase the the target unit.

Which data structures do we need to enable this?



SSD Internal Data Structures

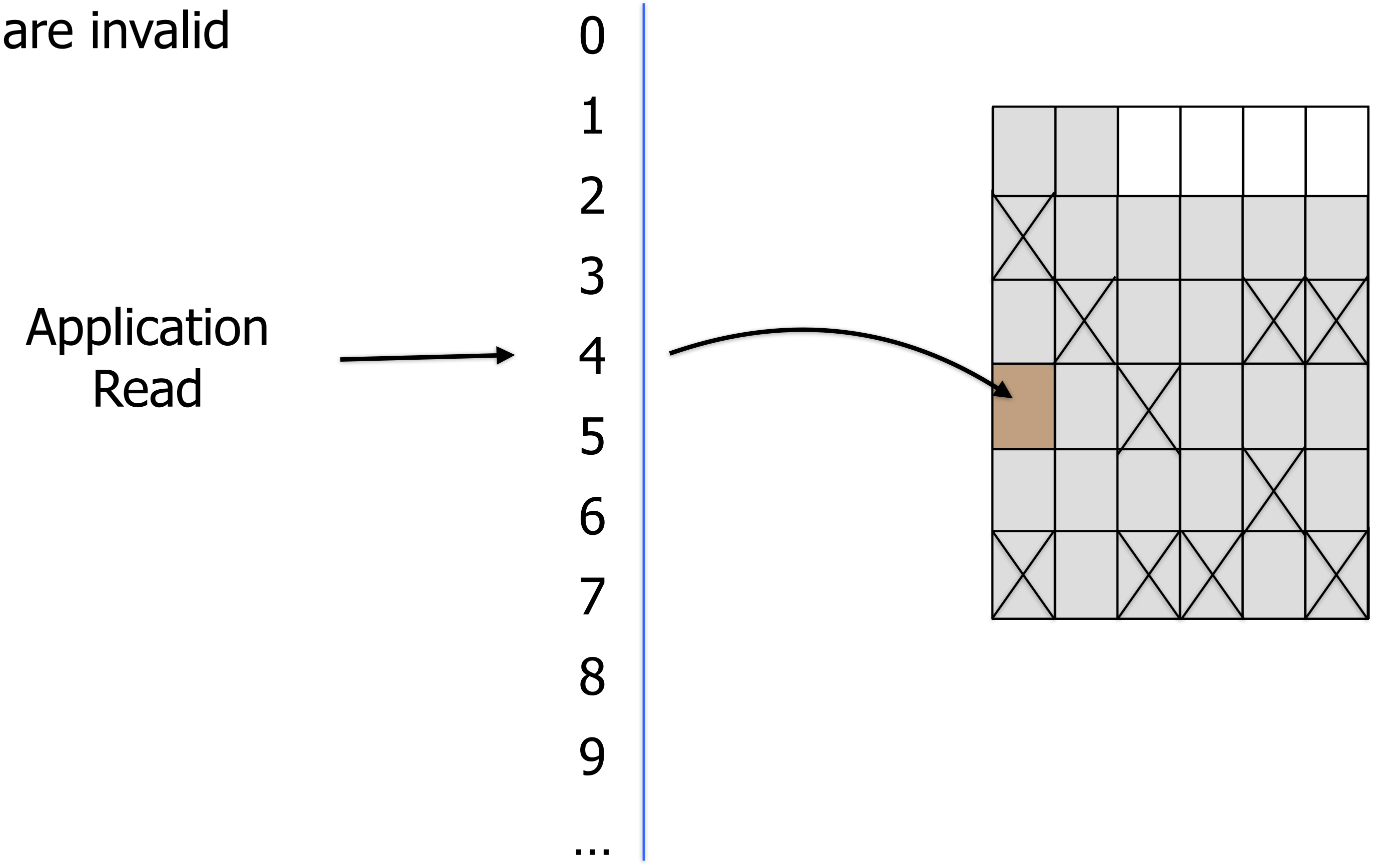
(1) bitmap of which pages are invalid



SSD Internal Data Structures

(1) bitmap of which pages are invalid

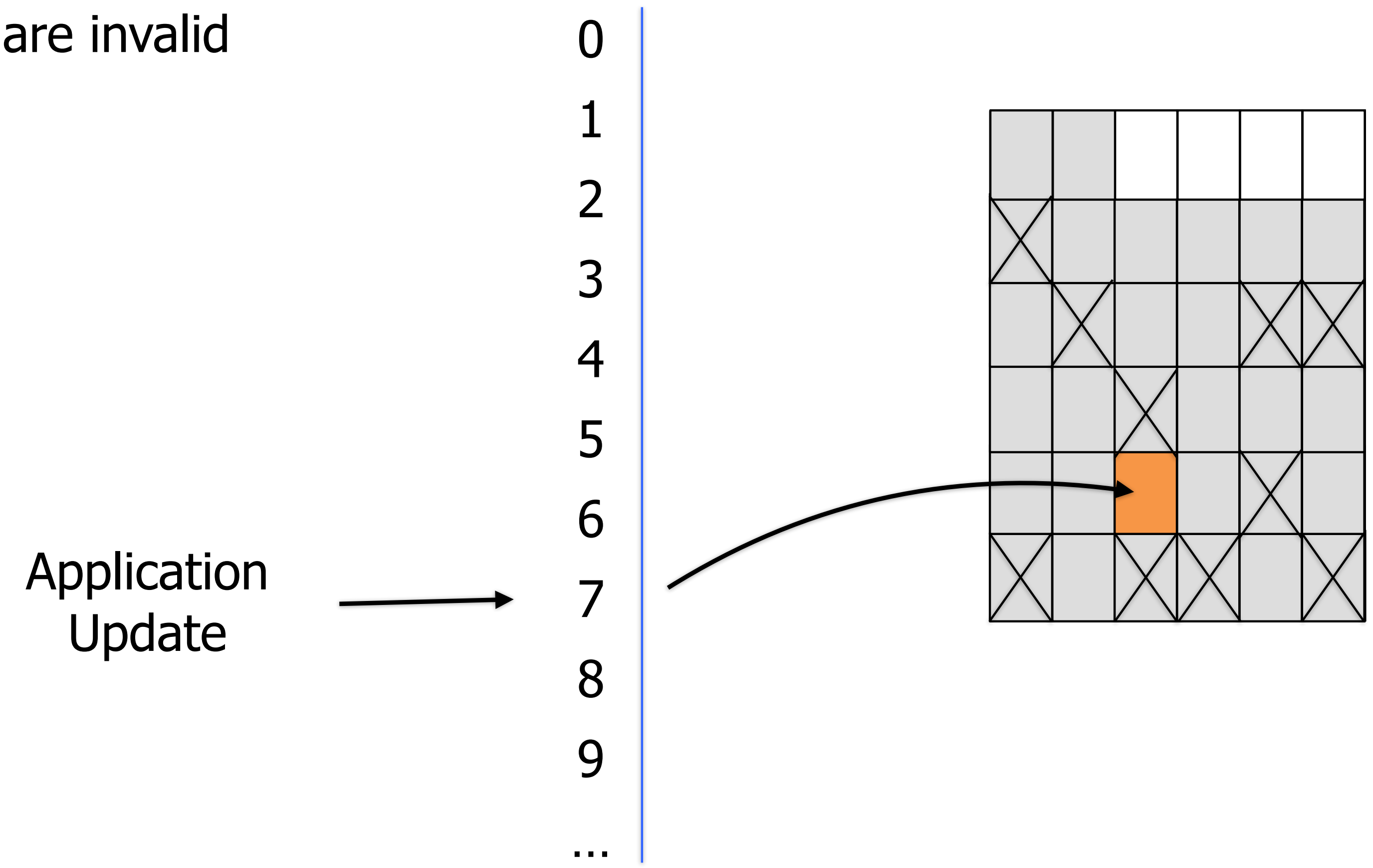
(2) page mapping table



SSD Internal Data Structures

(1) bitmap of which pages are invalid

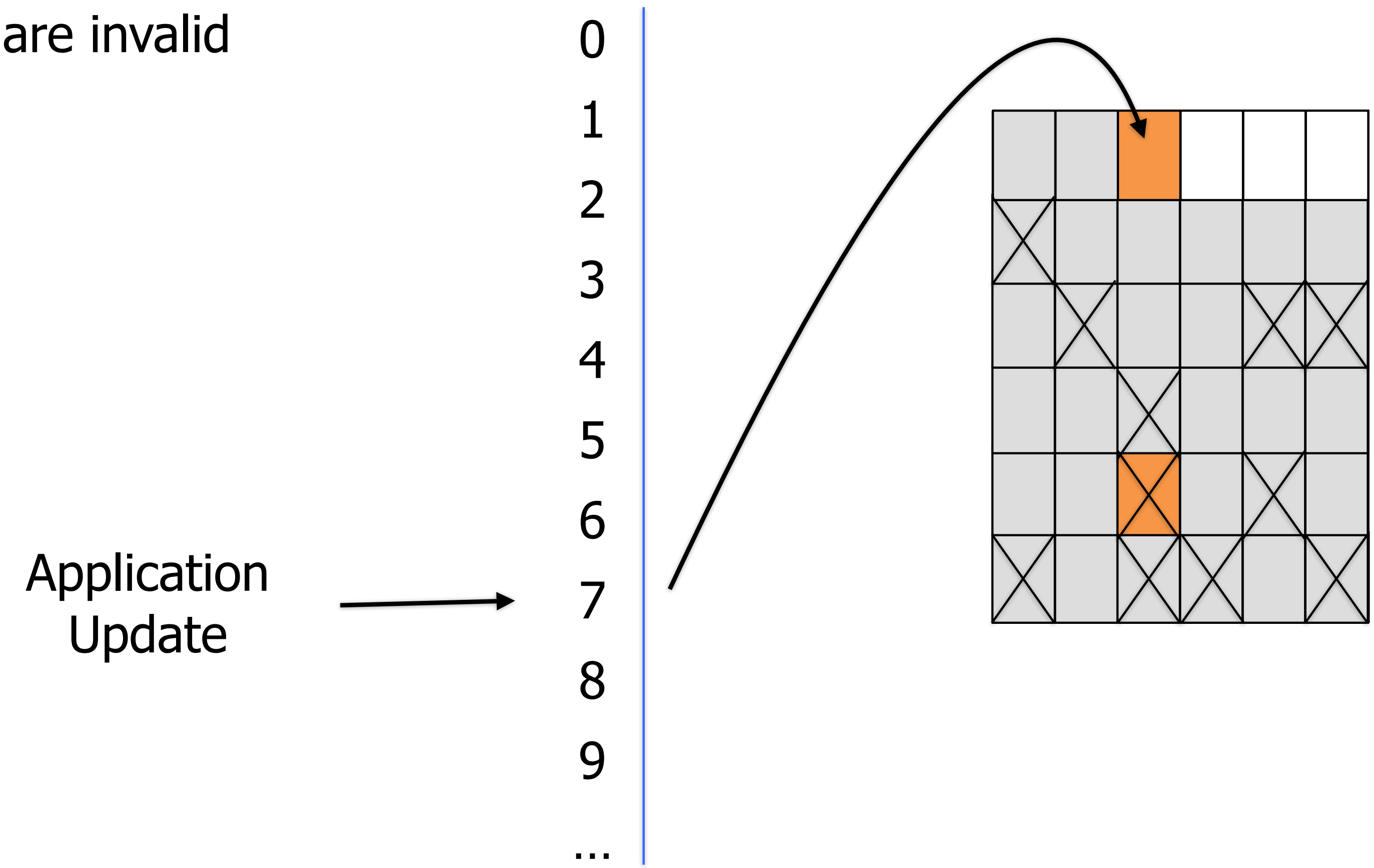
(2) page mapping table



SSD Internal Data Structures

(1) bitmap of which pages are invalid

(2) page mapping table



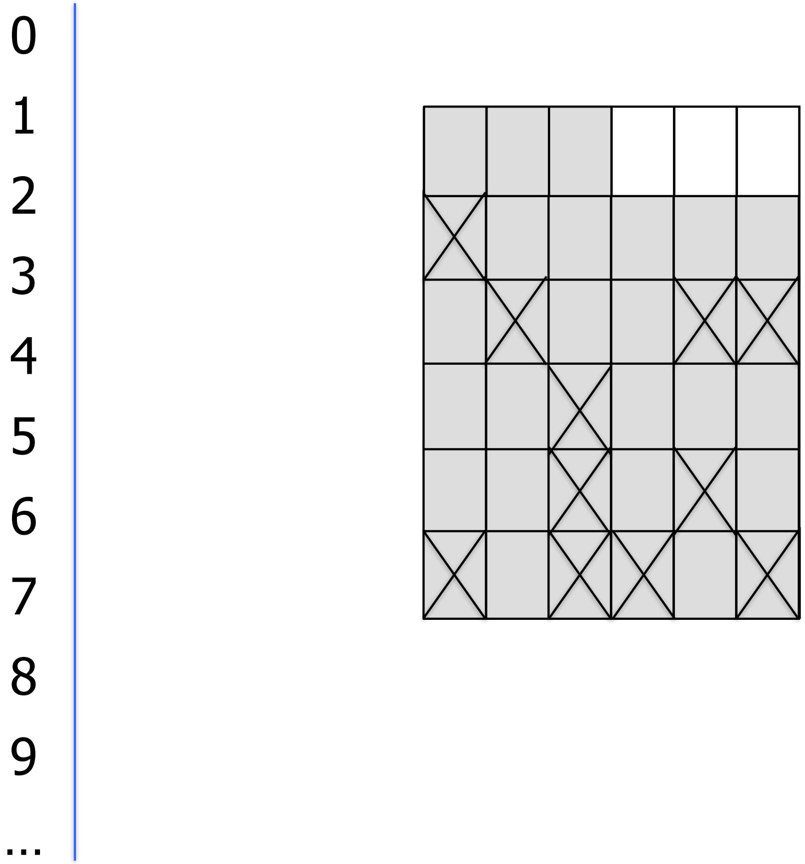
SSD Internal Data Structures

(1) bitmap of which pages are invalid

(2) page mapping table

a.k.a **Flash Translation Layer**

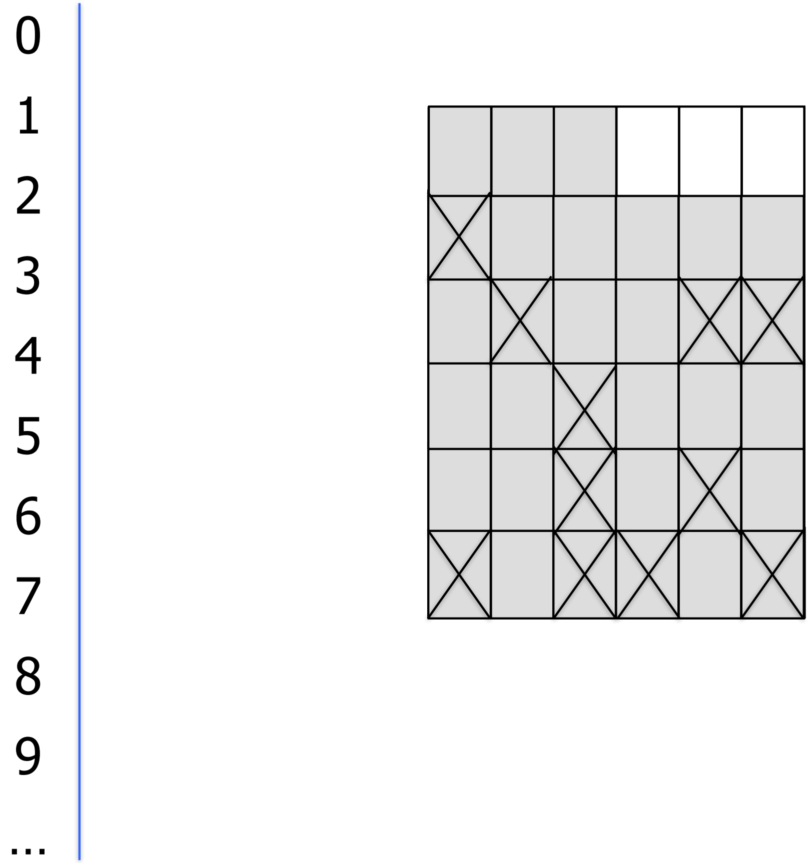
Subject of much research



SSD Over-Provisioning

Any SSD contains some space reserved for accommodating invalid data before garbage-collection is triggered.

The more over-provisioning, the more robust performance is guaranteed for random writes.

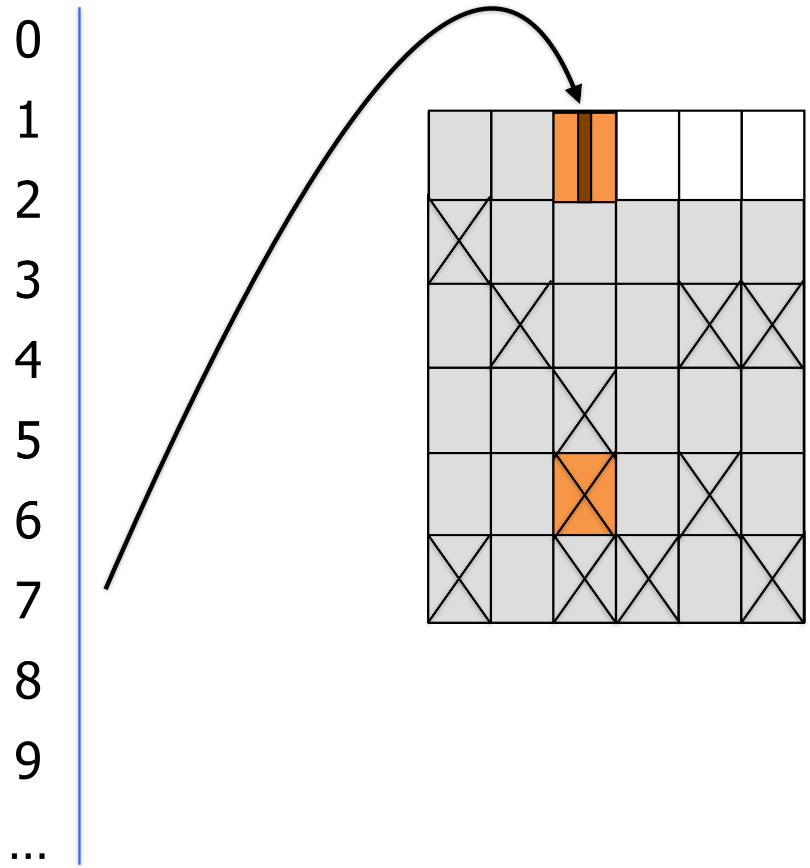


Principle 1: Avoid Small Updates

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

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

Application
Update smaller than page size

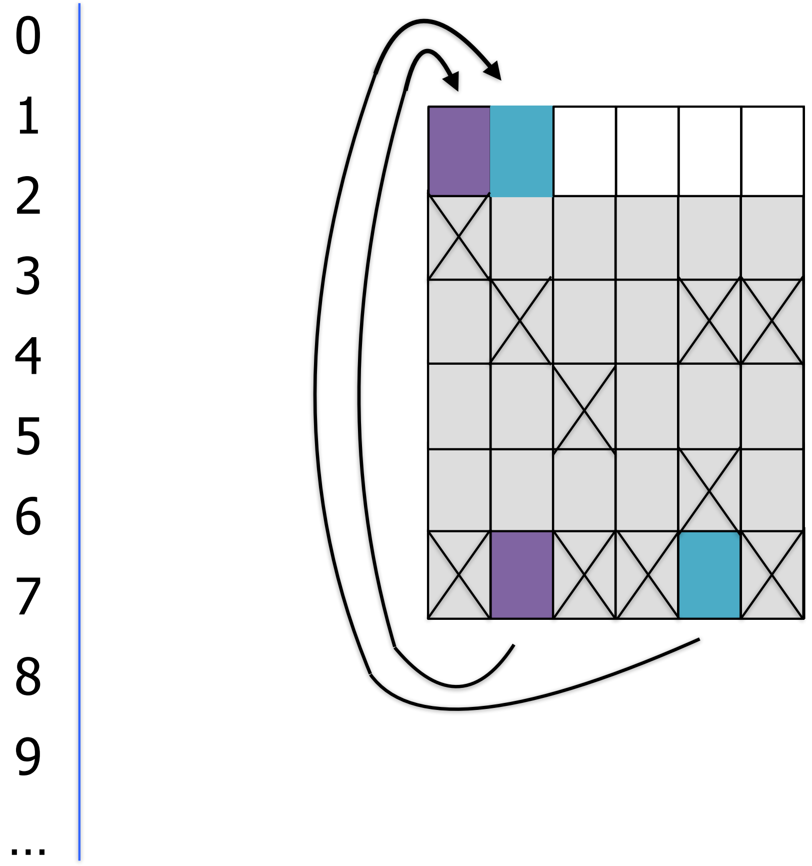


Principle 2: Avoid Random Updates

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

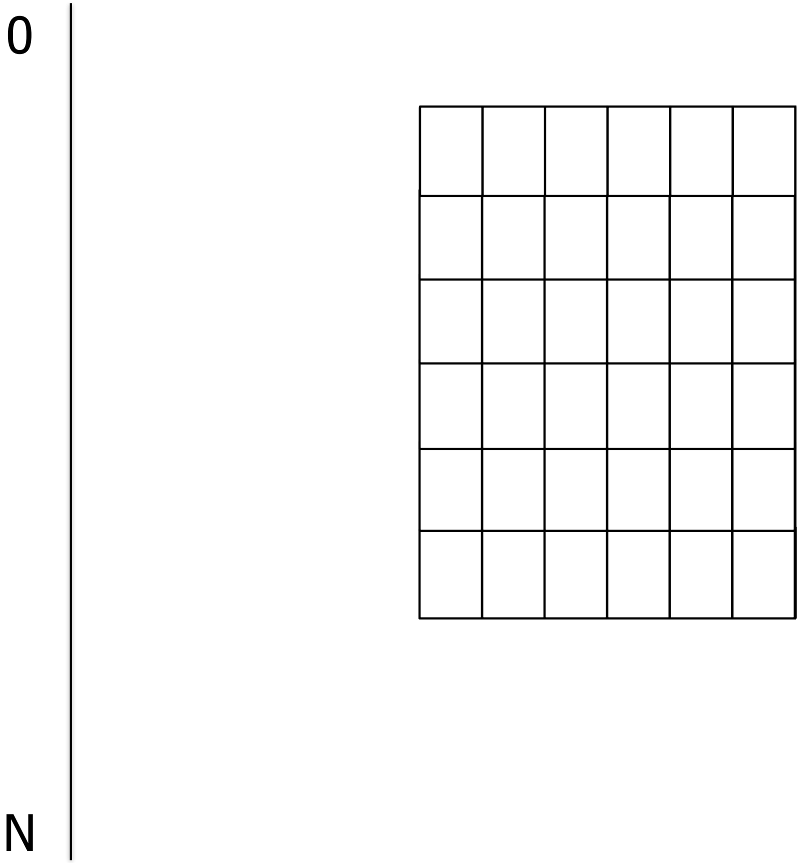
GC = fraction of pages in erase unit migrated each garbage-collection operation operation

$$\text{Write-amplification} = \frac{\mathbf{1}}{\mathbf{1\text{-}GC}}$$



Principle 2: Avoid Random Updates

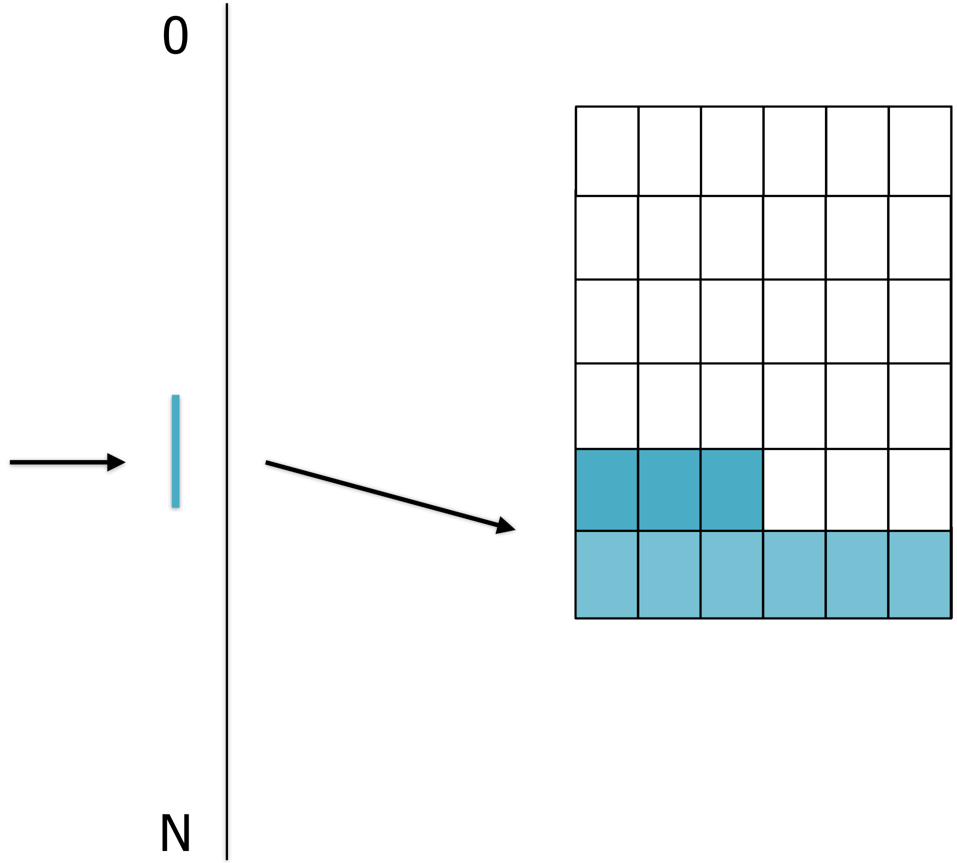
Instead write sequentially, and update data written at the same time all at once



Principle 2: Avoid Random Updates

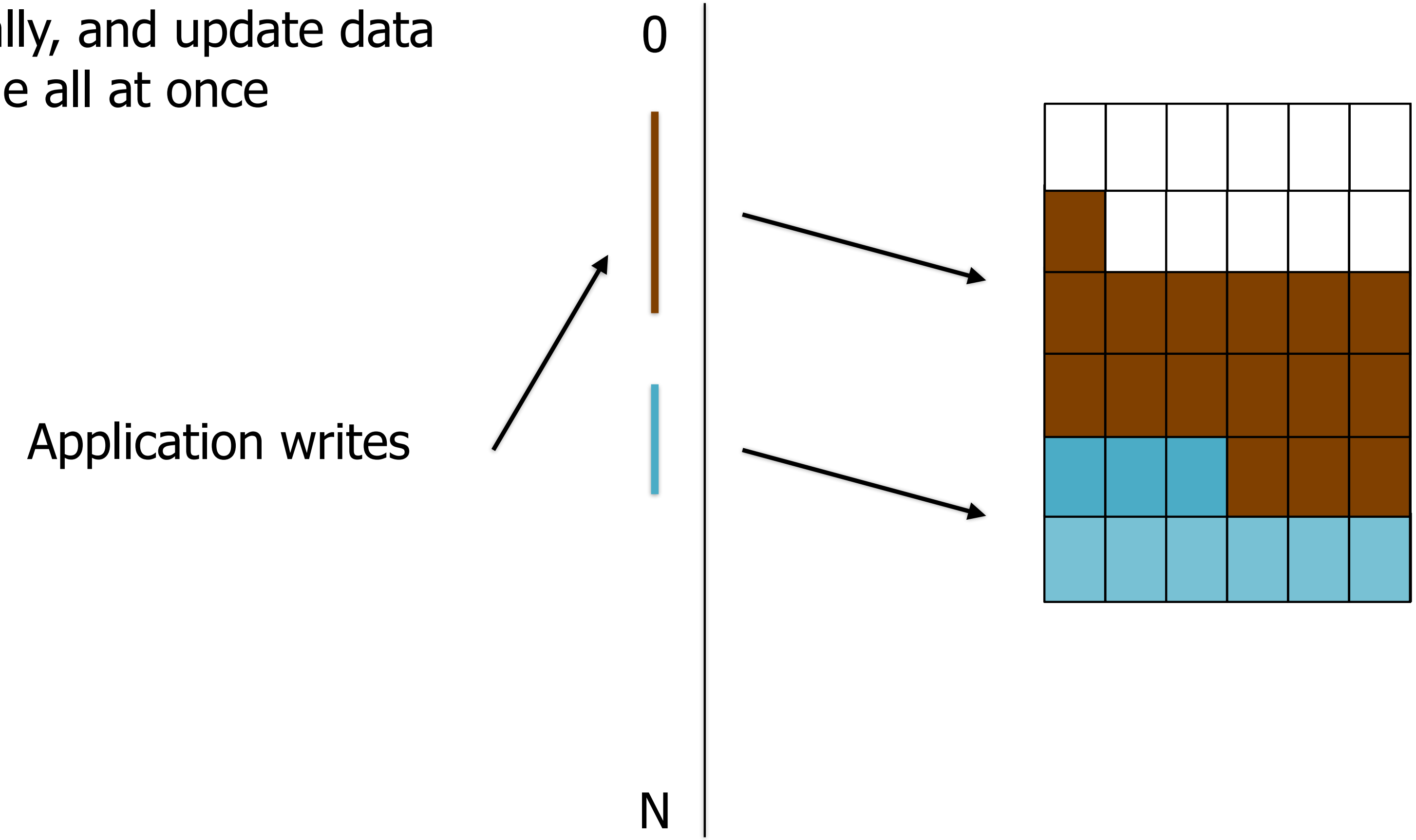
Instead write sequentially, and update data written at the same time all at once

Application writes



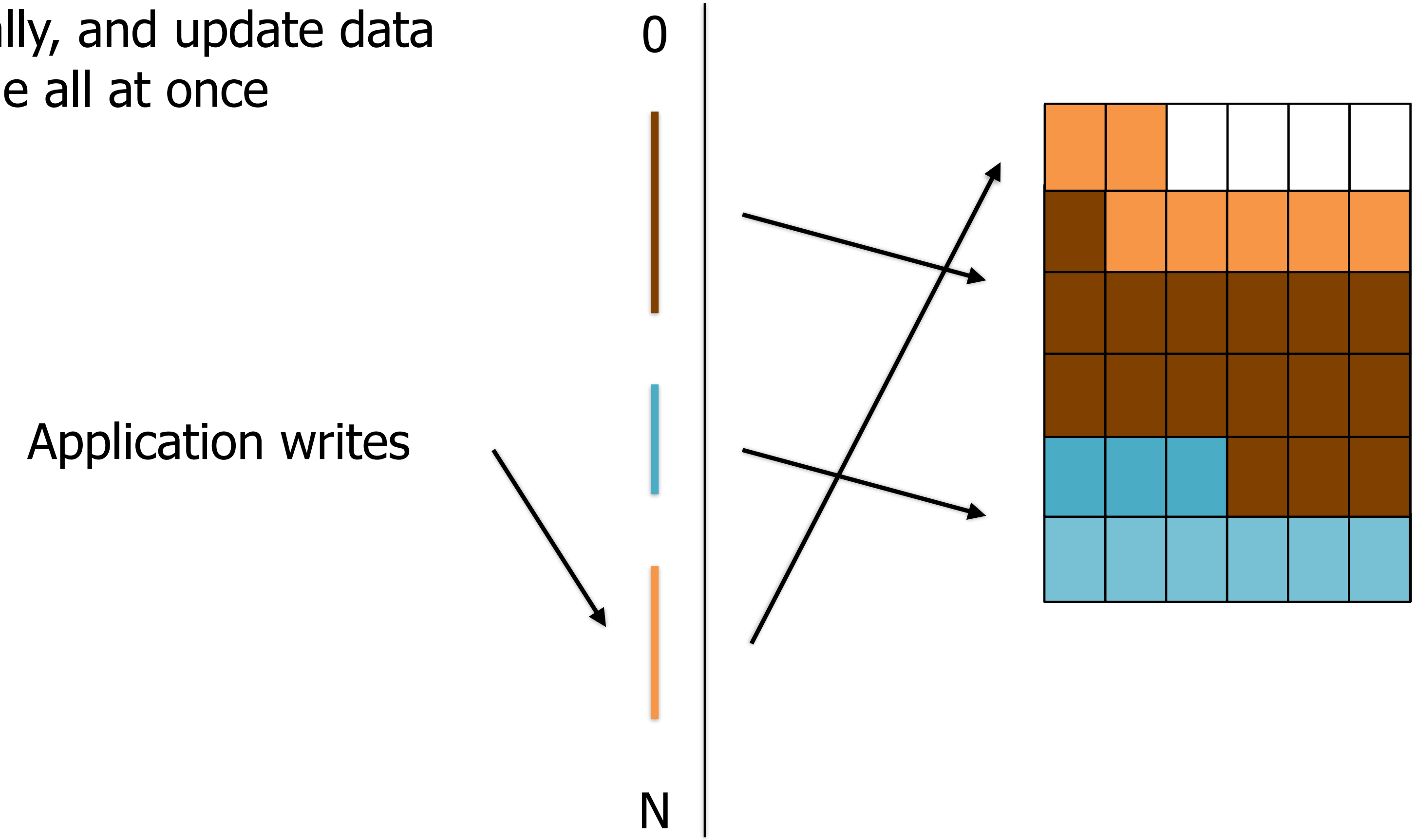
Principle 2: Avoid Random Updates

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



Principle 2: Avoid Random Updates

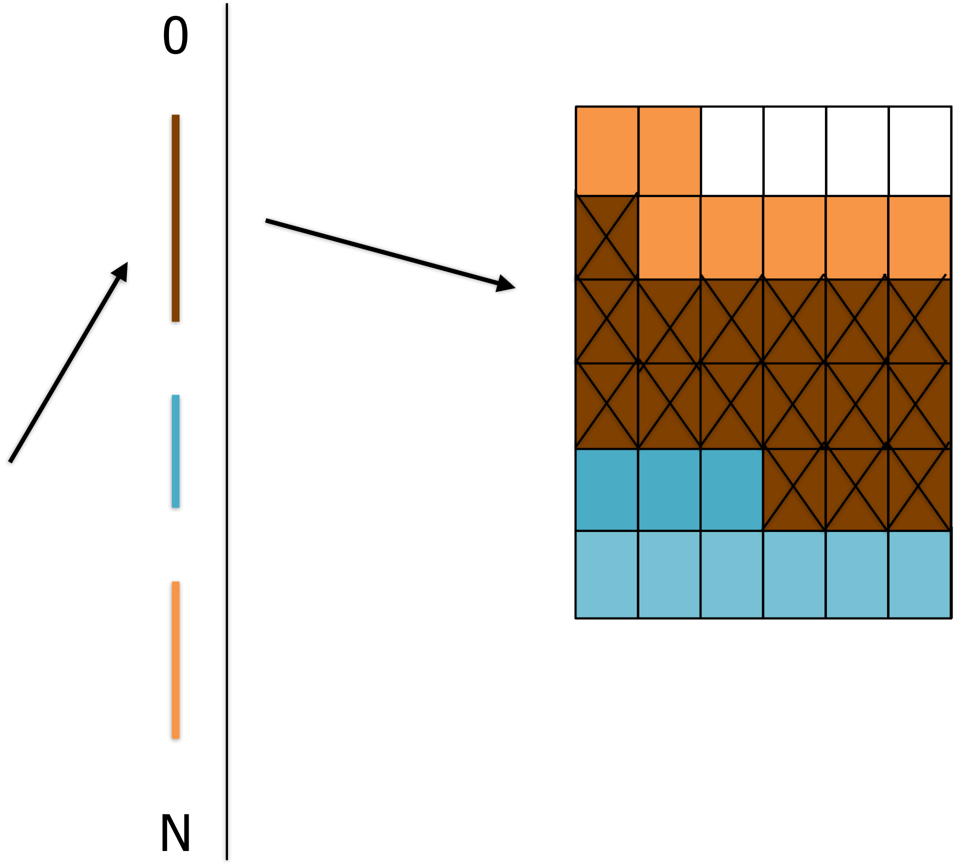
Instead write sequentially, and update data written at the same time all at once



Principle 2: Avoid Random Updates

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

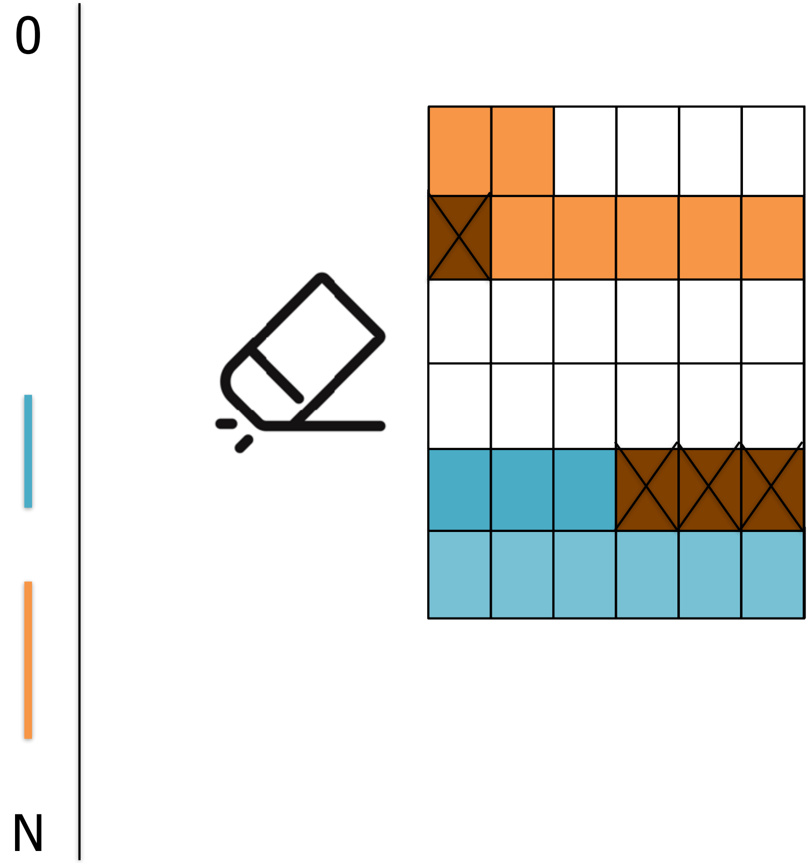
Application delete



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

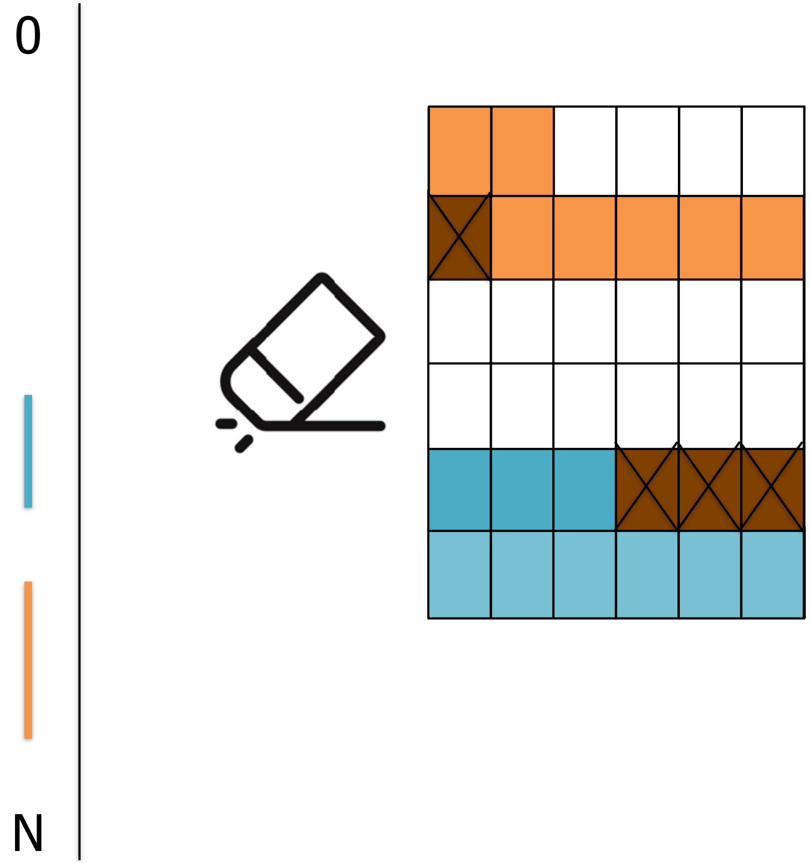


Principle 2: Avoid Random Updates

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

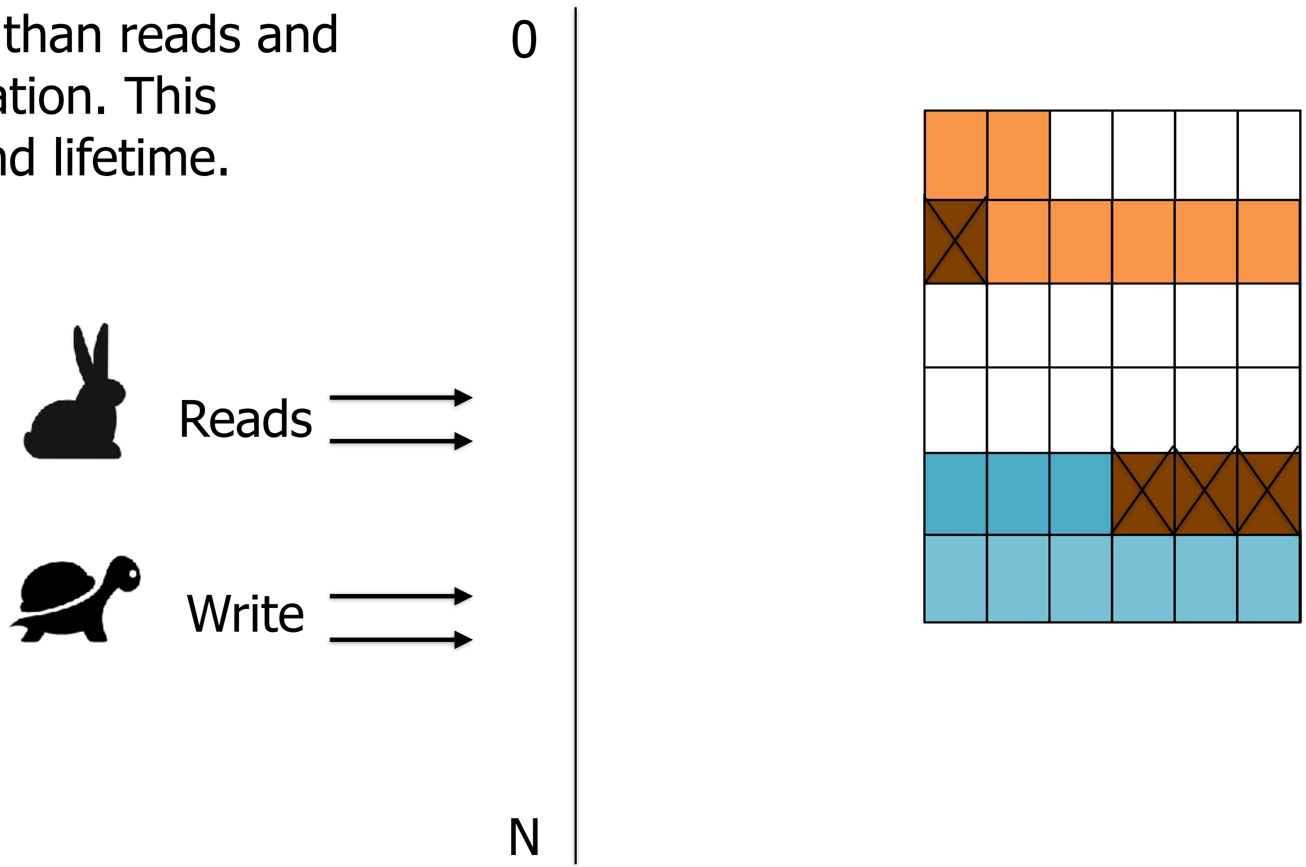
Free erase blocks without garbage-collection

Write-amplification = 1



Principle 3: Prioritize Reads over Writes

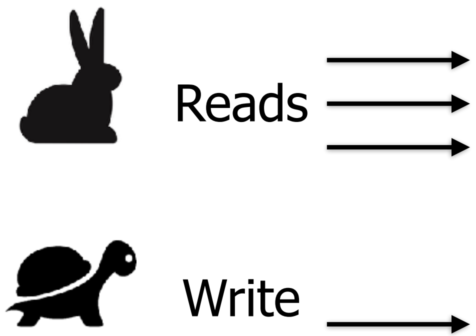
Writes take longer to execute than reads and may also entail write-amplification. This degrades SSD performance and lifetime.

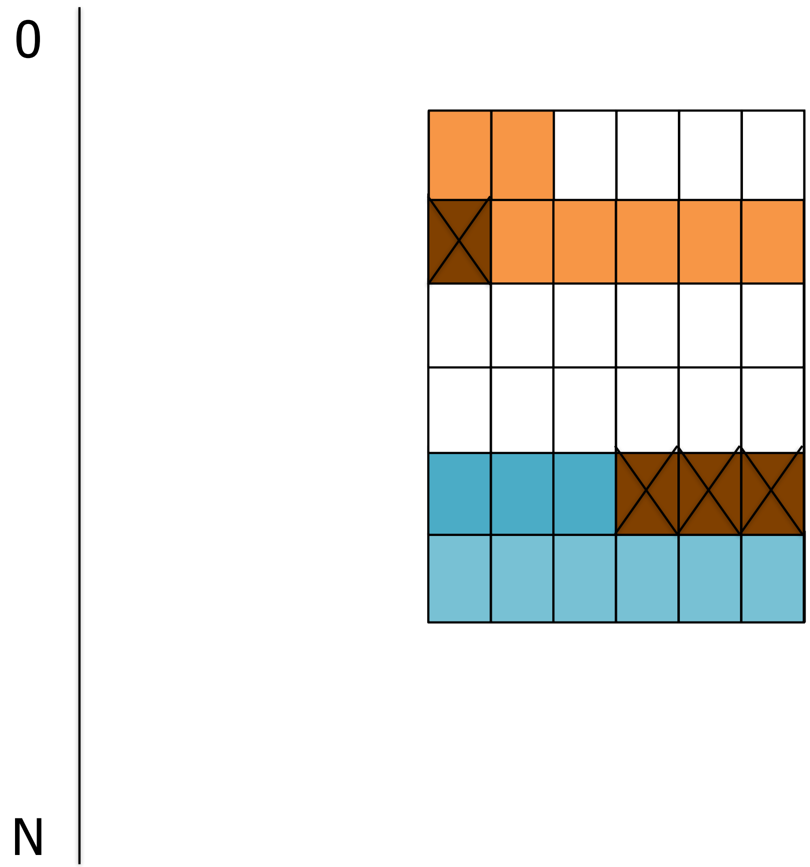


Principle 3: Prioritize Reads over Writes

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.

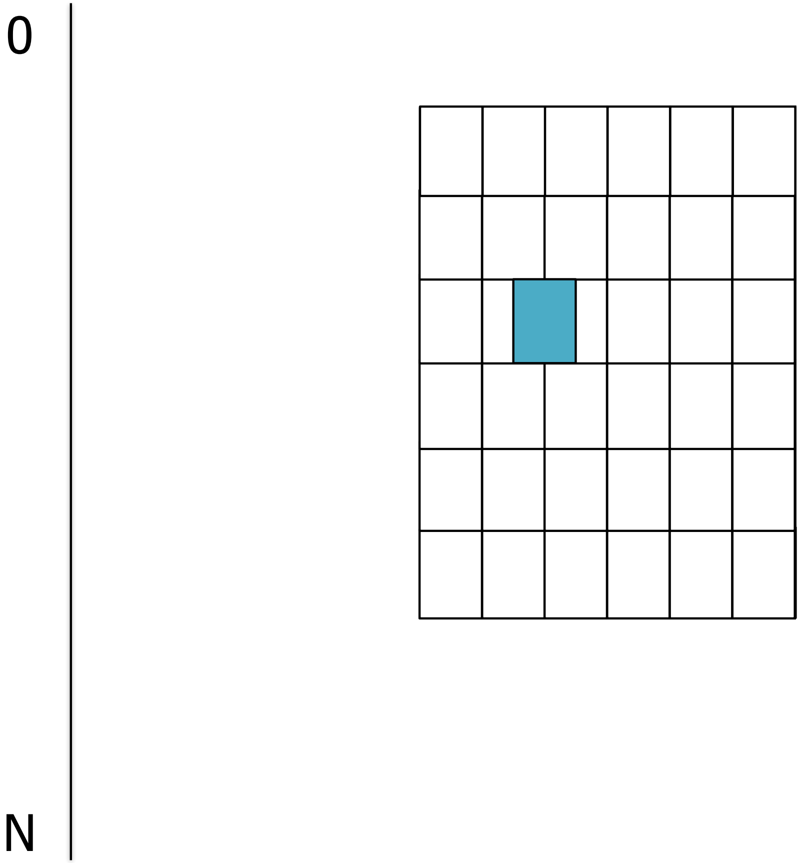




Principle 4: Reads/Writes should be page aligned

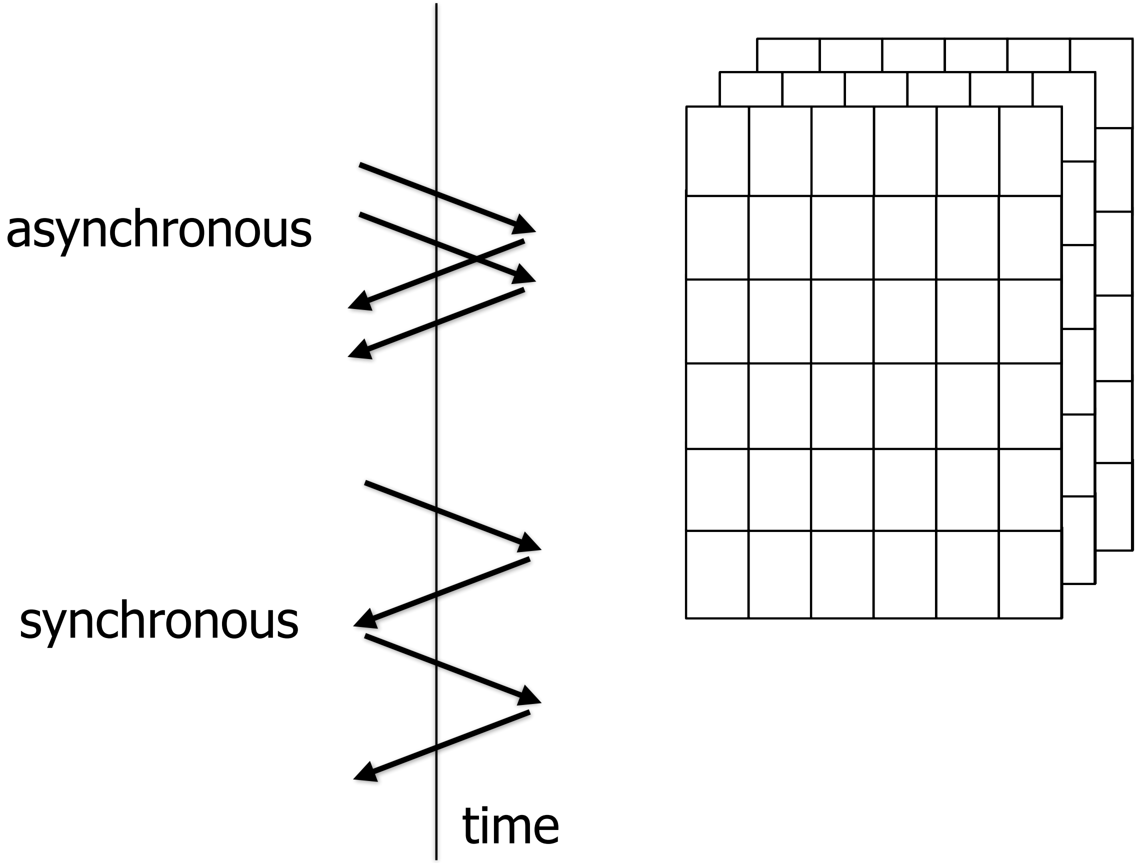
Reading a misaligned page triggers 2 flash reads

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



Principle 5: Asynchronous I/Os for the win

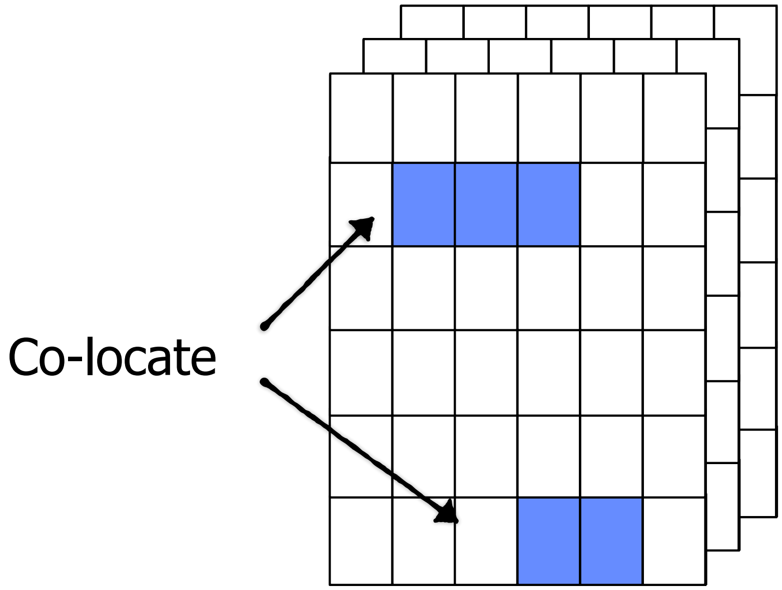
Internal parallelism of SSD makes asynchronous I/O faster than synchronous I/O



Principle 6: Defragmentation on SSDs is a Bad Idea

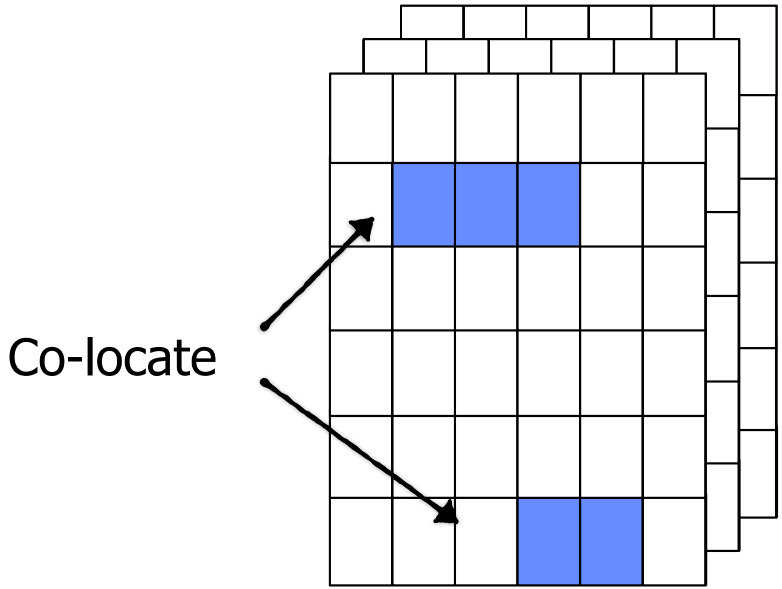
Random read I/Os are fast, so relocating parts of a file to be close does not significantly improve read speed.

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



Principle 7: Artificial Over-Provisioning

The less data you store, the more space the SSD will have for performing efficient garbage-collection



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 →

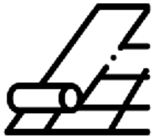
Conclusion

Phase-Change memory

Discontinued

SSD

disk



Shingled disks

New

Tape

Archival

DNA

Research

← Faster

Cheaper →

Conclusion

Introduced to memory hierarchy
Inner workings of disks and SSDs

SSD

disk

Take-away: sequential writes   sandom writes