

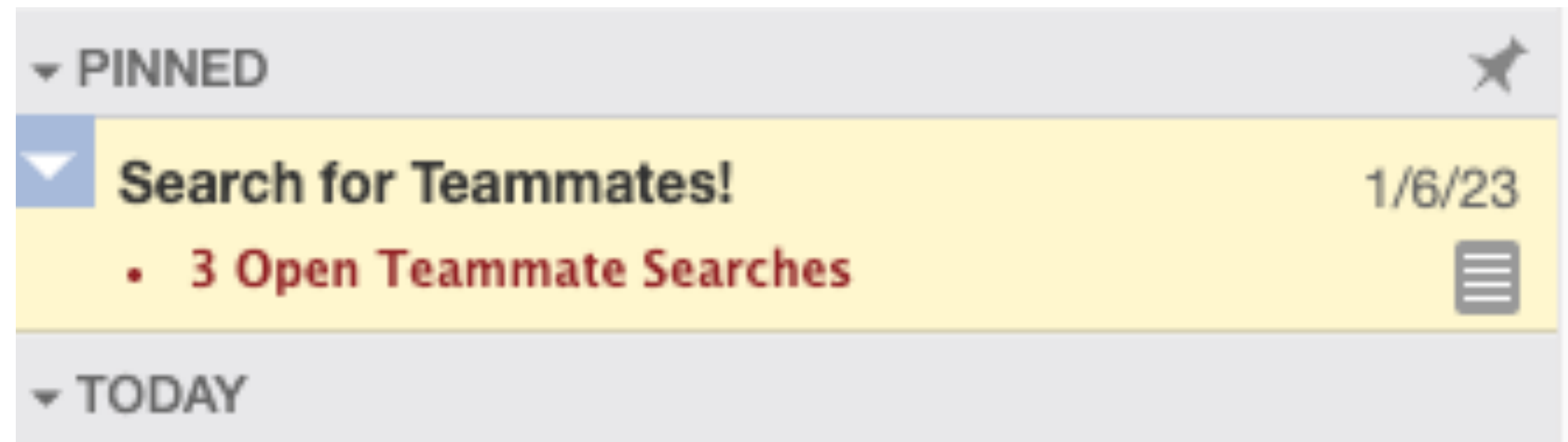
# **Tutorial**

## **Storage, Table & Buffer management**

**Niv Dayan - Jan 19, 2023**

**We will start at 3:10 pm**

If you don't have a group



If you are a group of 3 that would like an extra member, email me.

# Question 1

Suppose we are performing uniformly randomly distributed unaligned 2KB random writes to an SSD, for which the page size is 4KB the logical capacity is 85% of the physical capacity. What's the worst-case write-amplification we might expect?

Suppose we are doing these writes on RAID 1. How does this affect your answer?

# Question 1

Suppose we are performing uniformly randomly distributed unaligned 2KB random writes to an SSD, for which the page size is 4KB the logical capacity is 85% of the physical capacity. What's the worst-case write-amplification we might expect?

**Two 4KB pages are rewritten for each 2KB update, so we have immediate WA factor of 4.**



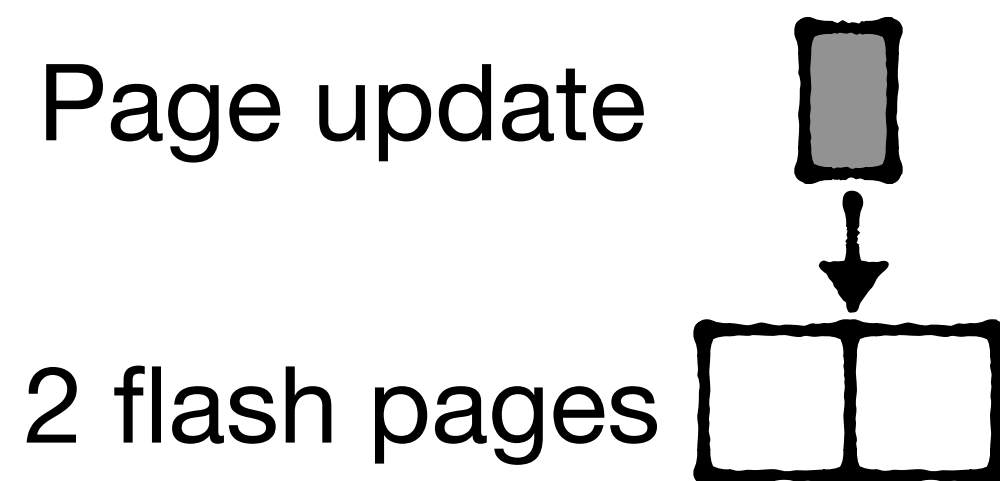
Suppose we are doing these writes on RAID 1. How does this affect your answer?

# Question 1

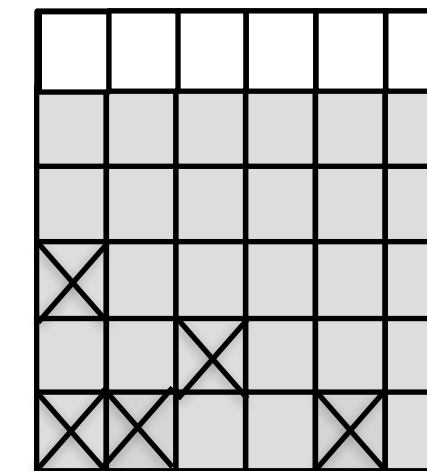
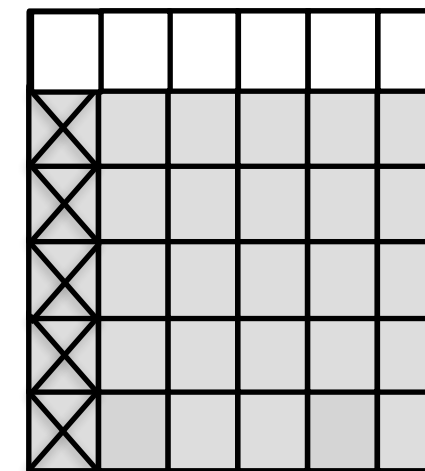
Suppose we are performing uniformly randomly distributed unaligned 2KB random writes to an SSD, for which the page size is 4KB the logical capacity is 85% of the physical capacity. What's the worst-case write-amplification we might expect?

Two 4KB pages are rewritten for each 2KB update, so we have immediate WA factor of 4.

**Garbage-collection contributes a multiplicative maximum factor of  $1/(1-0.85) \approx 6$  to give 24.**



**Worst-case**



**Typical**

Older  
↓

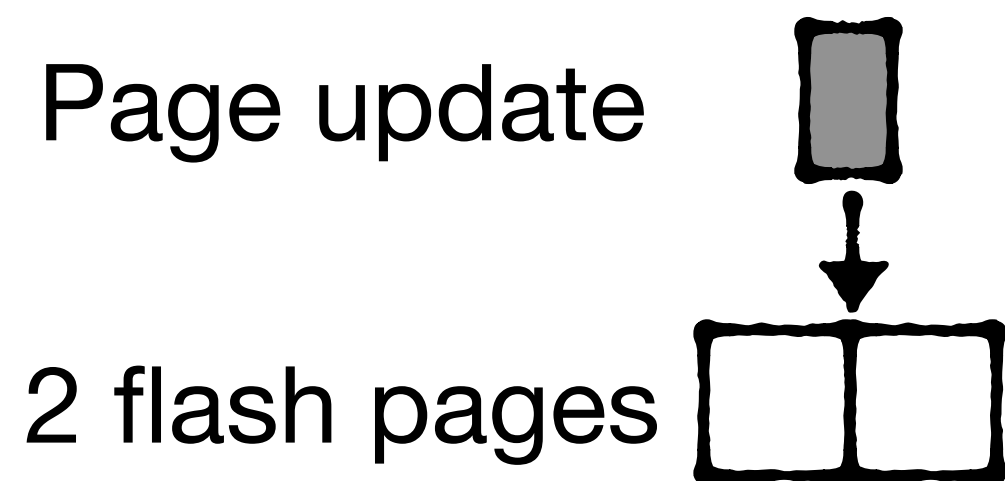
Suppose we are doing these writes on RAID 1. How does this affect your answer?

# Question 1

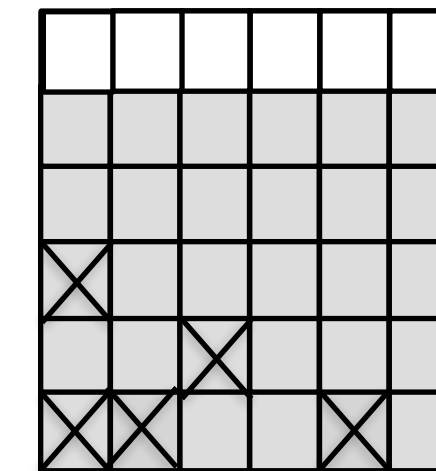
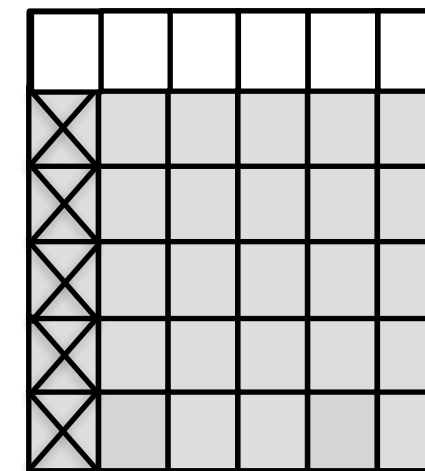
Suppose we are performing uniformly randomly distributed unaligned 2KB random writes to an SSD, for which the page size is 4KB the logical capacity is 85% of the physical capacity. What's the worst-case write-amplification we might expect?

Two 4KB pages are rewritten for each 2KB update, so we have immediate WA factor of 4.

Garbage-collection contributes a multiplicative maximum factor of  $1/(1-0.85) \approx 6$  to give 24.



Worst-case



Typical

Older

Suppose we are doing these writes on RAID 1. How does this affect your answer?

**RAID 1 does mirroring, so this would double our maximum WA to 48.**

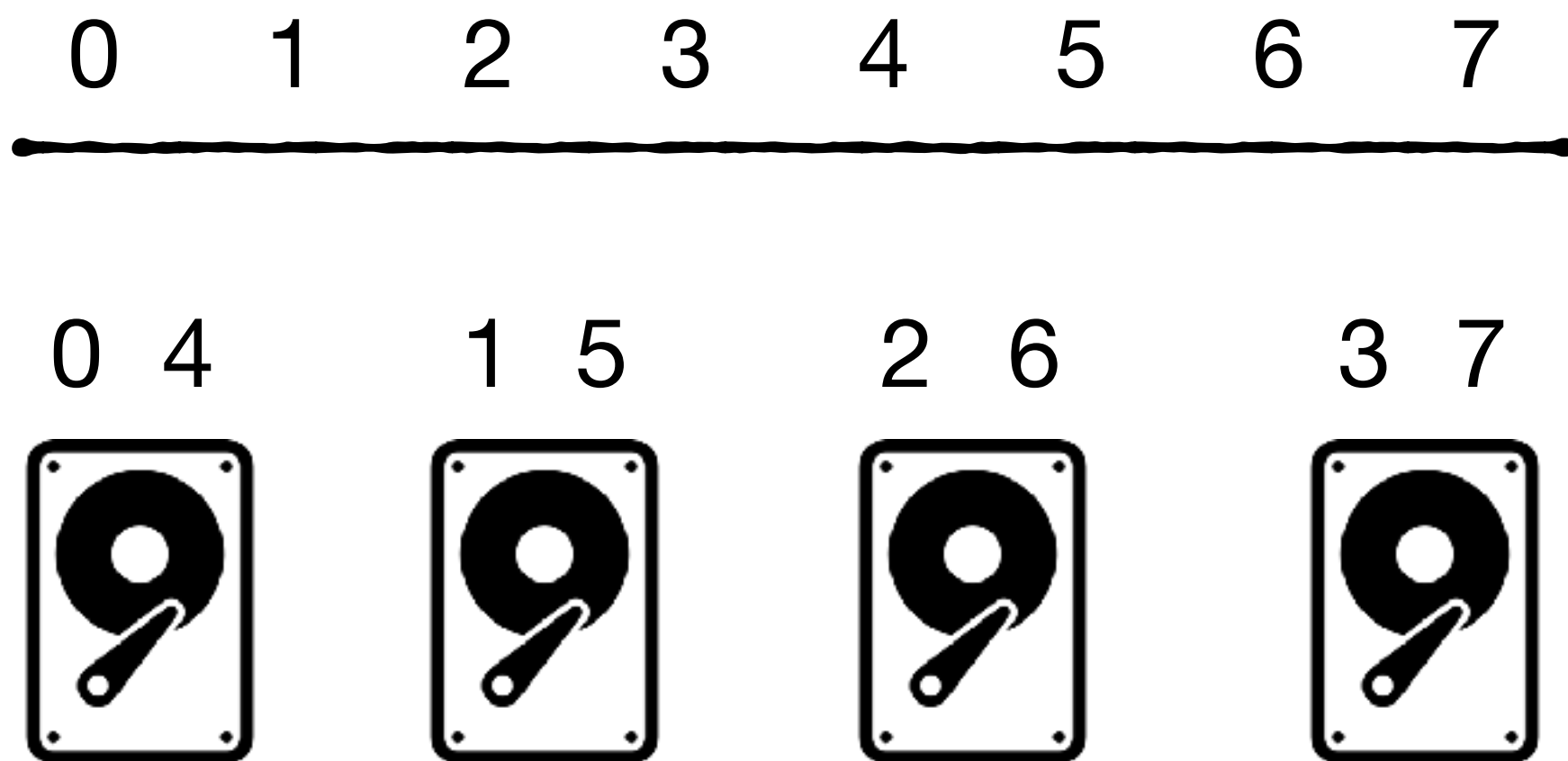
## Question 2

Suppose a disk that's completely resistant to failure is invented. Which RAID option would you choose for it, assuming we mostly care about sequential read/write performance? How about if we only care about random read performance?

## Question 2

Suppose a disk that's completely resistant to failure is invented. Which RAID option would you choose for it, assuming we mostly care about sequential read/write performance? How about if we only care about random read performance?

RAID 0 (pure striping) if we care about sequential I/O

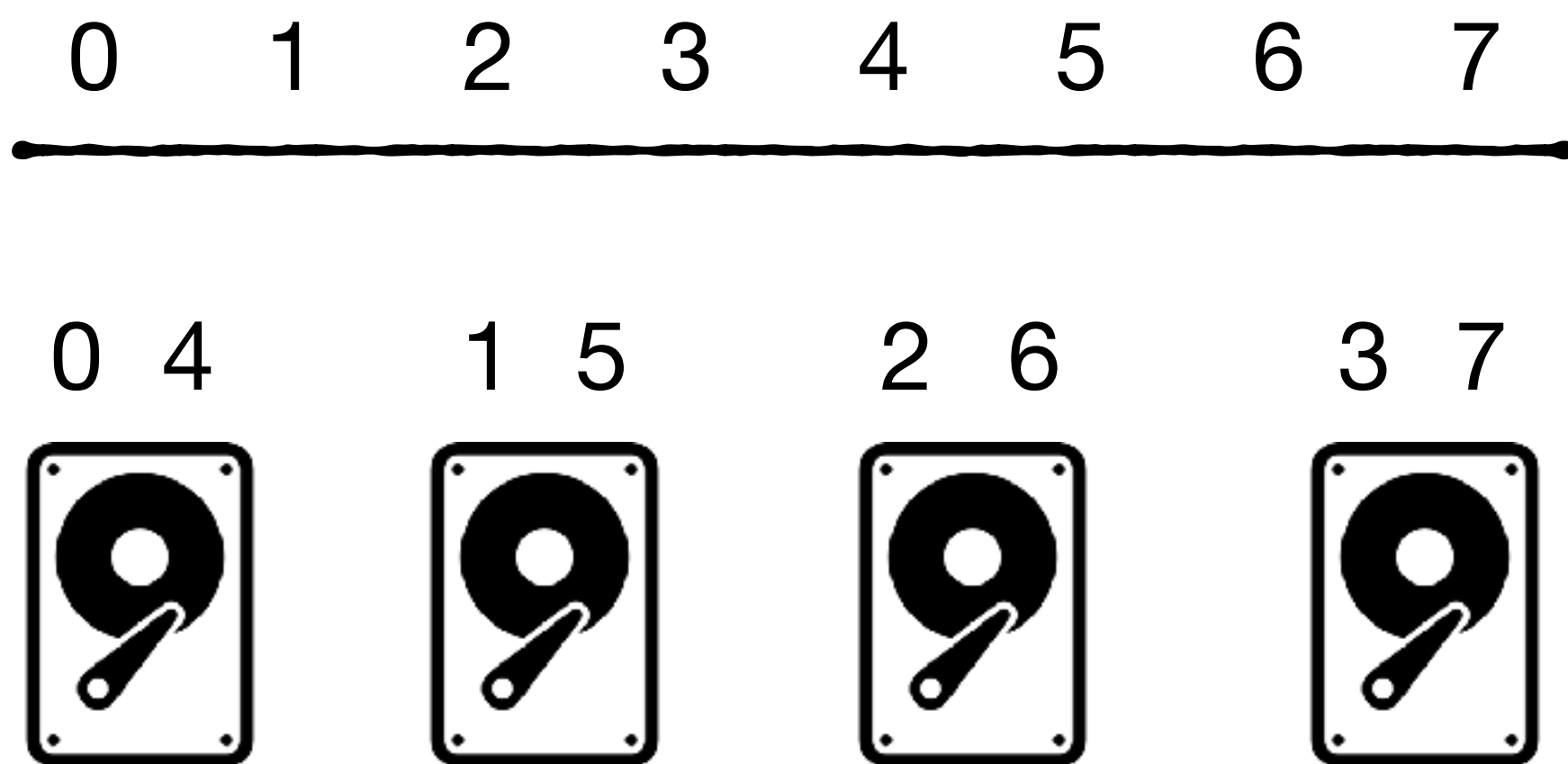




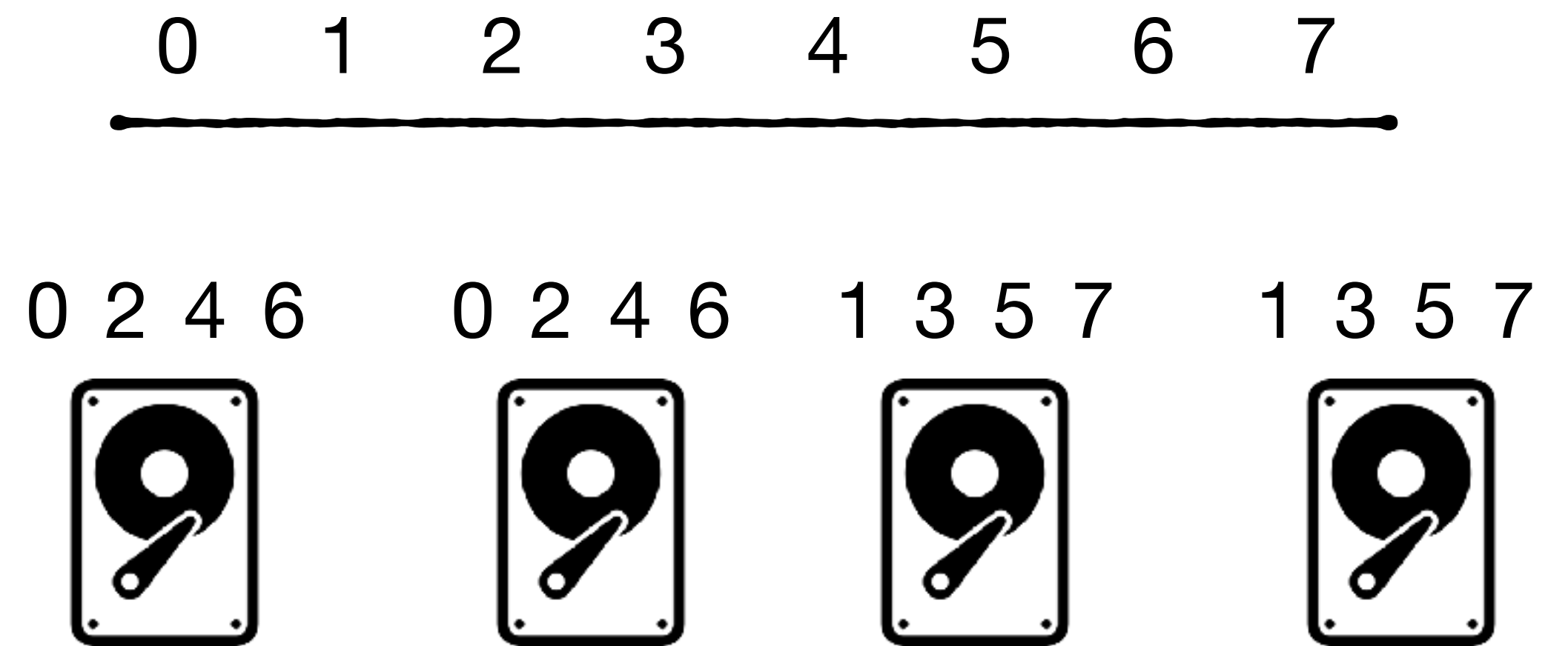
## Question 2

Suppose a disk that's completely resistant to failure is invented. Which RAID option would you choose for it, assuming we mostly care about sequential read/write performance? How about if we only care about random read performance?

RAID 0 (pure striping) if we care about sequential I/O



RAID 1 or 1+0 (with mirroring) if we care about random read I/O, as we have more flexibility



## Question 3

Consider a 4 disks, each with 100 MB/s sequential throughput. What sequential write throughput would you expect on:

**RAID 0?**

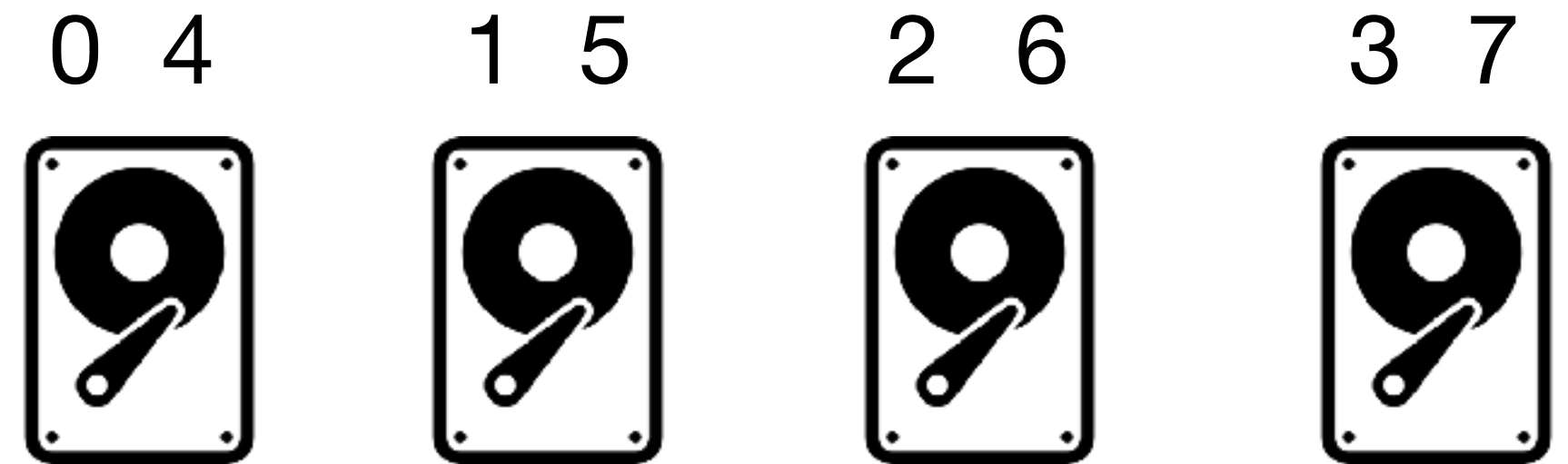
**RAID 1?**

**RAID 0+1?**

## Question 3

Consider a 4 disks, each with 100 MB/s sequential throughput. What sequential write throughput would you expect on:

**RAID 0?** 400 MB/s as it's doing pure striping.



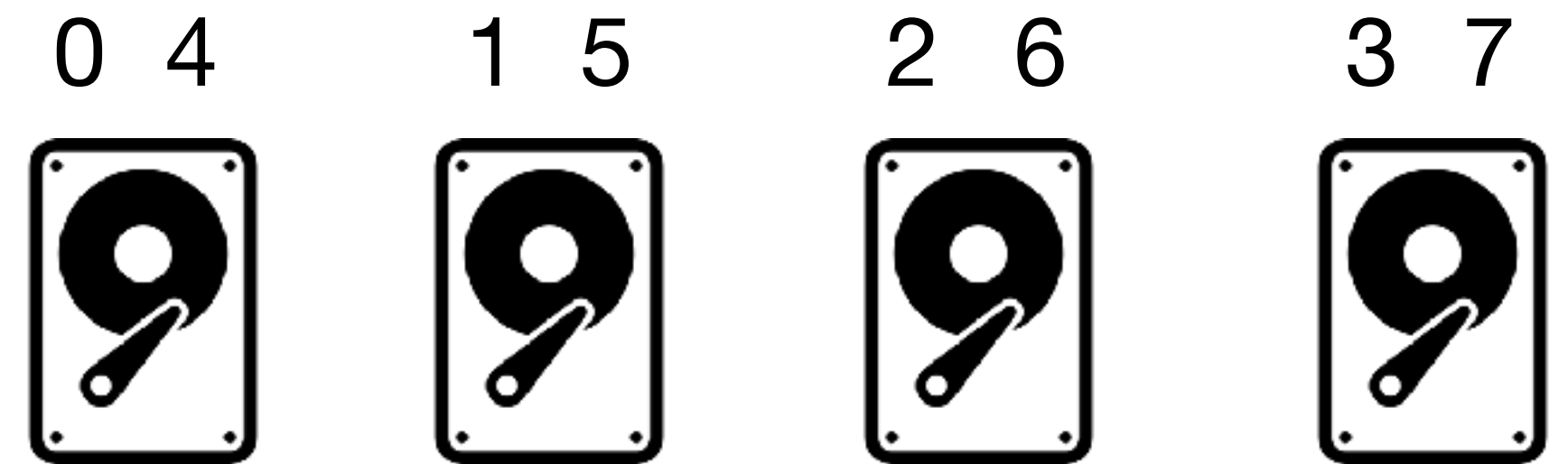
**RAID 1?**

**RAID 0+1?**

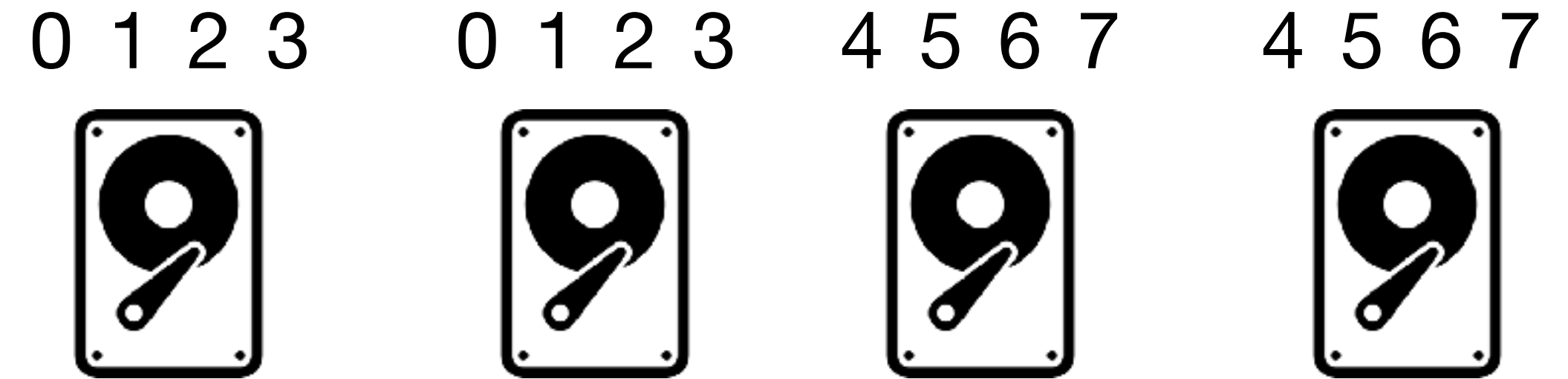
## Question 3

Consider a 4 disks, each with 100 MB/s sequential throughput. What sequential write throughput would you expect on:

**RAID 0?** 400 MB/s as it's doing pure striping.



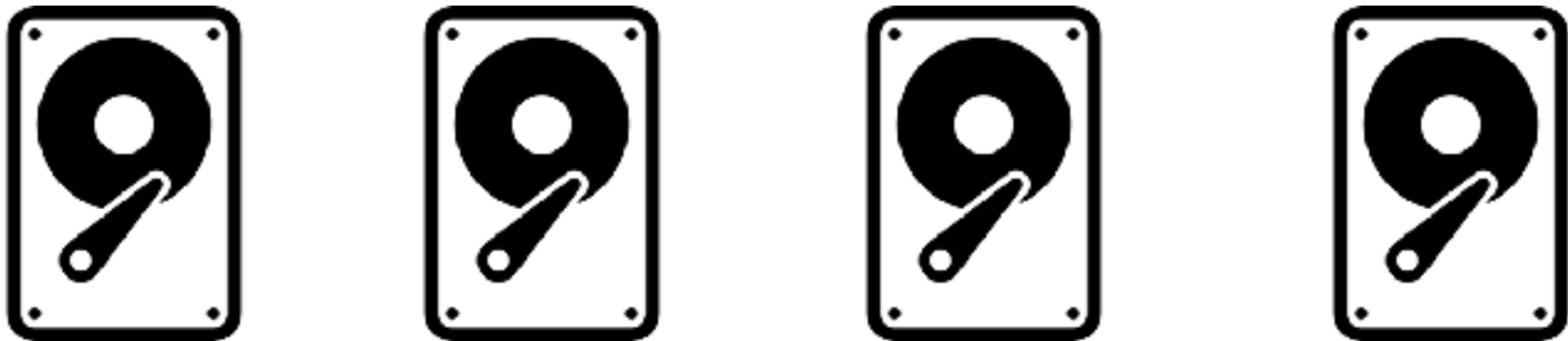
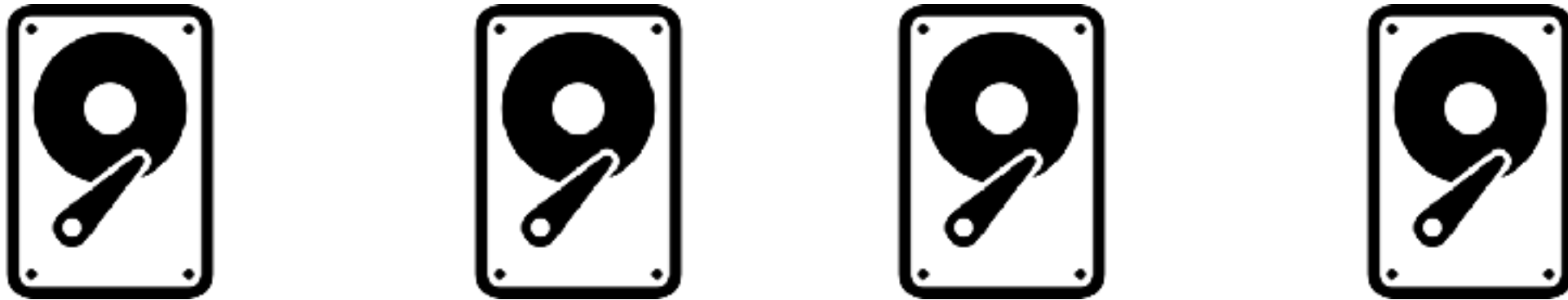
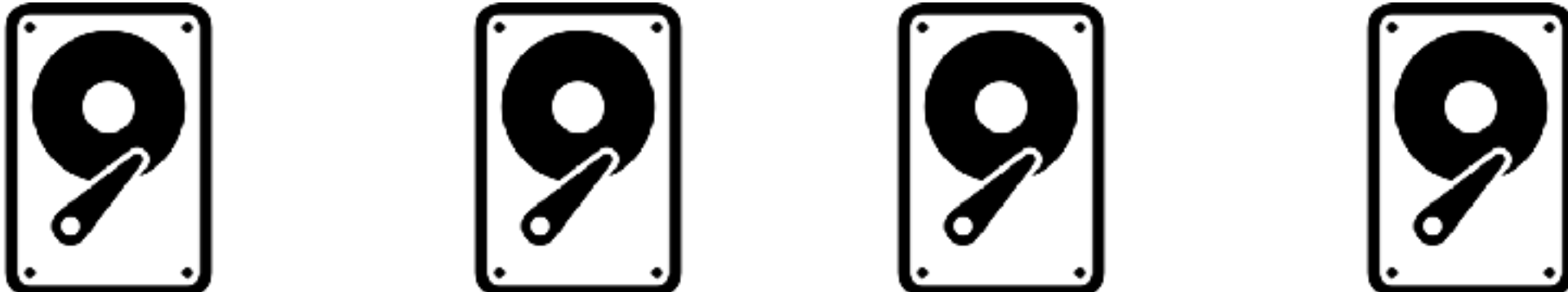
**RAID 1?** 100 MB/s as it's just doing mirroring



**RAID 0+1?**

## Question 3

Consider a 4 disks, each with 100 MB/s sequential throughput. What sequential write throughput would you expect on:

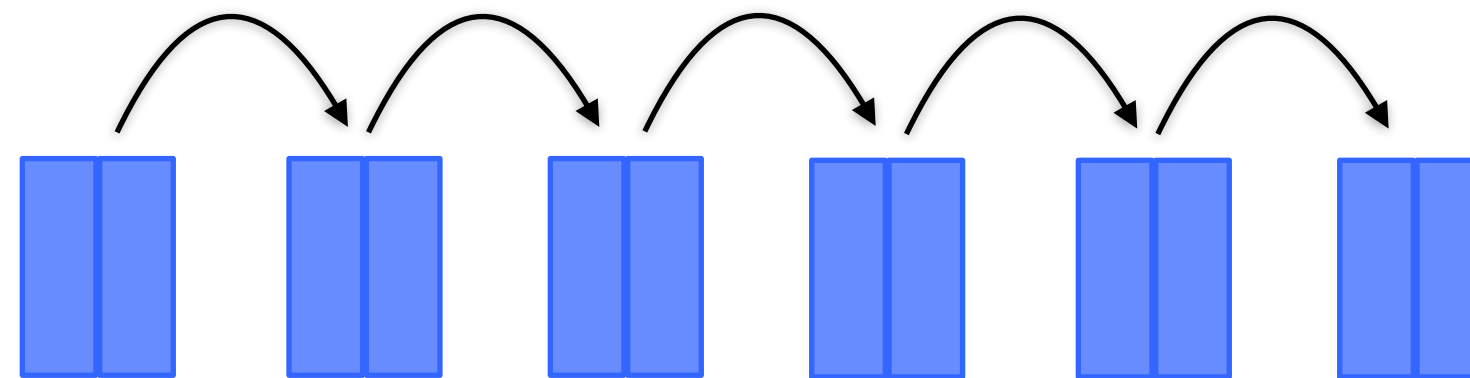
- RAID 0?** 400 MB/s as it's doing pure striping.
- 0 4      1 5      2 6      3 7
- 
- RAID 1?** 100 MB/s as it's just doing mirroring
- 0 1 2 3      0 1 2 3      4 5 6 7      4 5 6 7
- 
- RAID 0+1?** 200 MB/s as it's doing stripping across two pairs of drives
- 0 2 4 6      0 2 4 6      1 3 5 7      1 3 5 7
- 

## Question 4

Consider a table allocated as a linked list of database pages.

What scan throughput (in MB/s) would you expect on disk? How about an SSD?

How can we better structure this table to improve throughput?



## Question 4

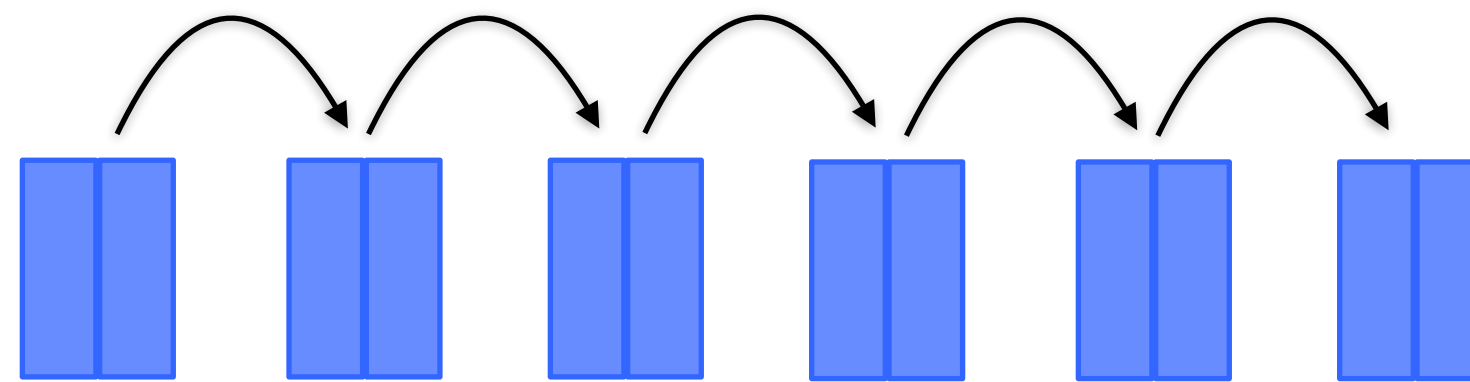
Consider a table allocated as a linked list of database pages.

What scan throughput (in MB/s) would you expect on disk? How about an SSD?

**A disk I/O takes 10ms to read 4KB pages. Throughput: 0.4 MB/s.**

**An SSD I/O takes 100us to read 4KB pages. Throughput: 40 MB/s.**

How can we better structure this table to improve throughput?



## Question 4

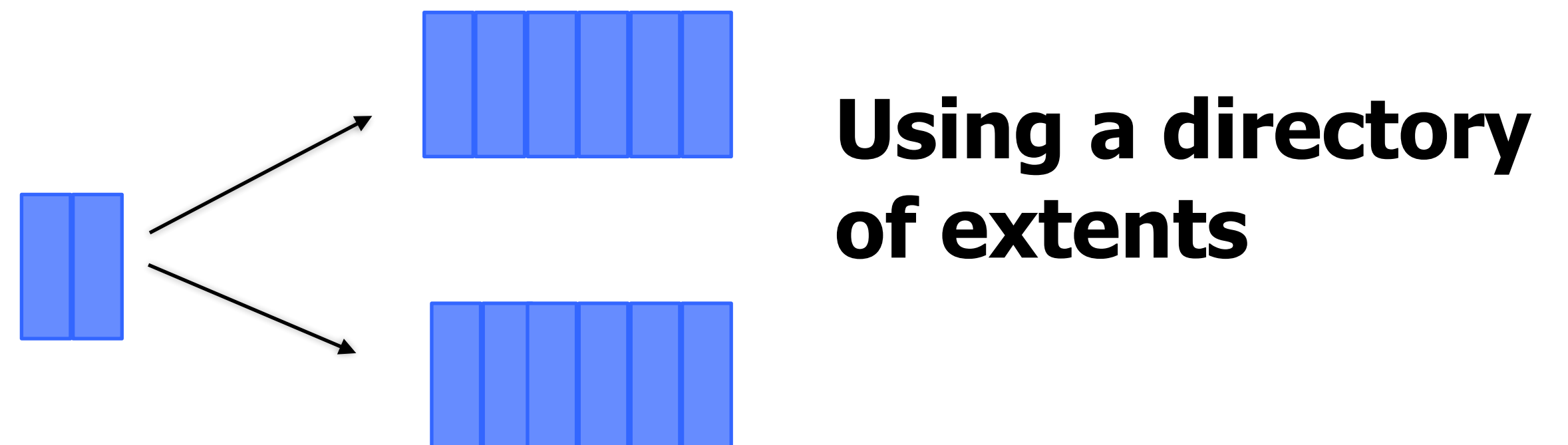
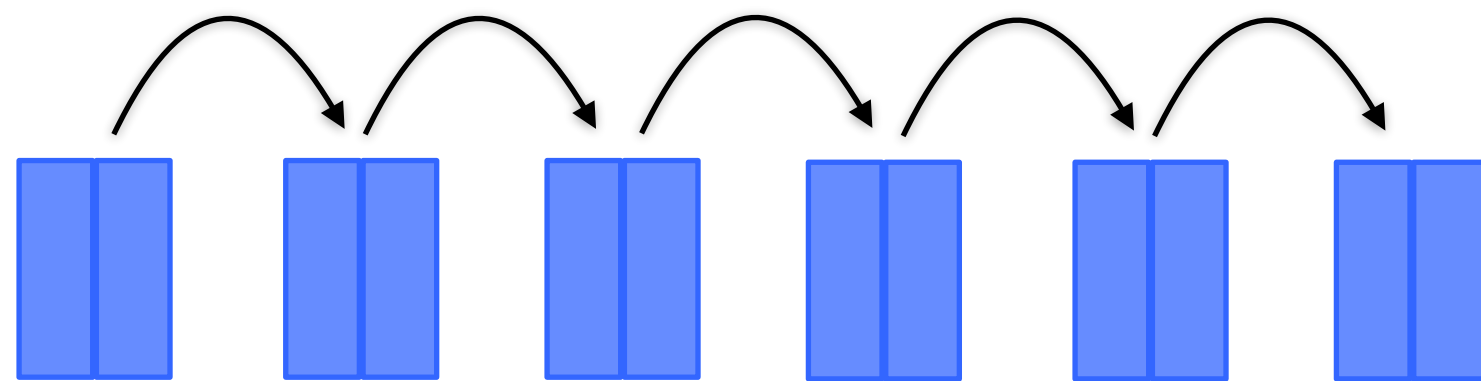
Consider a table allocated as a linked list of database pages.

What scan throughput (in MB/s) would you expect on disk? How about an SSD?

A disk I/O takes 10ms to read 4KB pages. Throughput: 0.4 MB/s.

An SSD I/O takes 100us to read 4KB pages. Throughput: 40 MB/s.

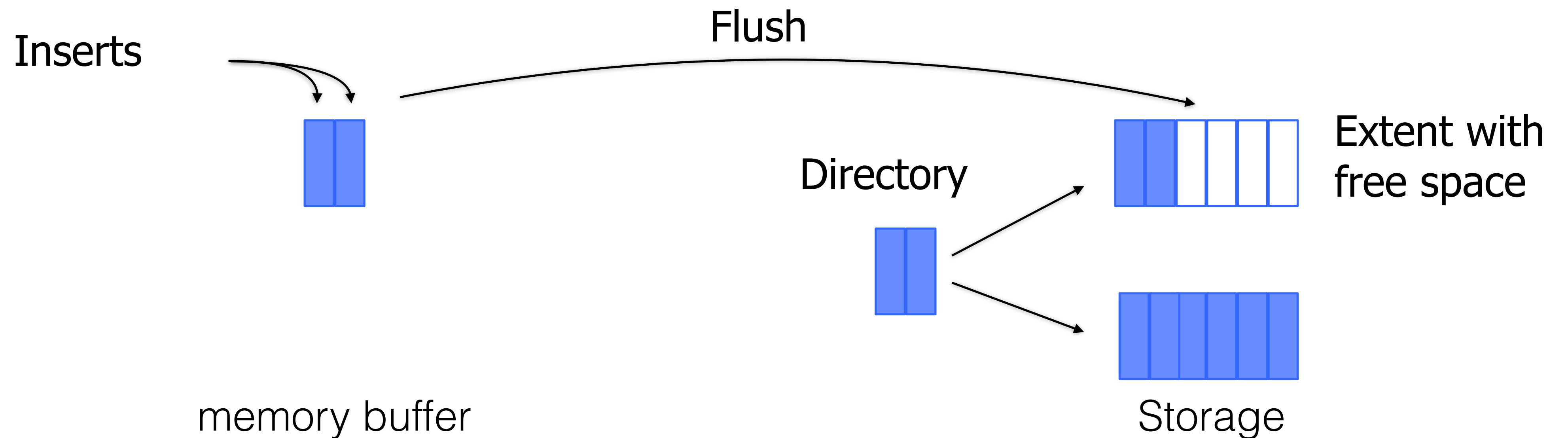
**How can we better structure this table to improve throughput?**





## Question 5

Recall how we can buffer insertions in memory until a page fills up and flush



## Question 5

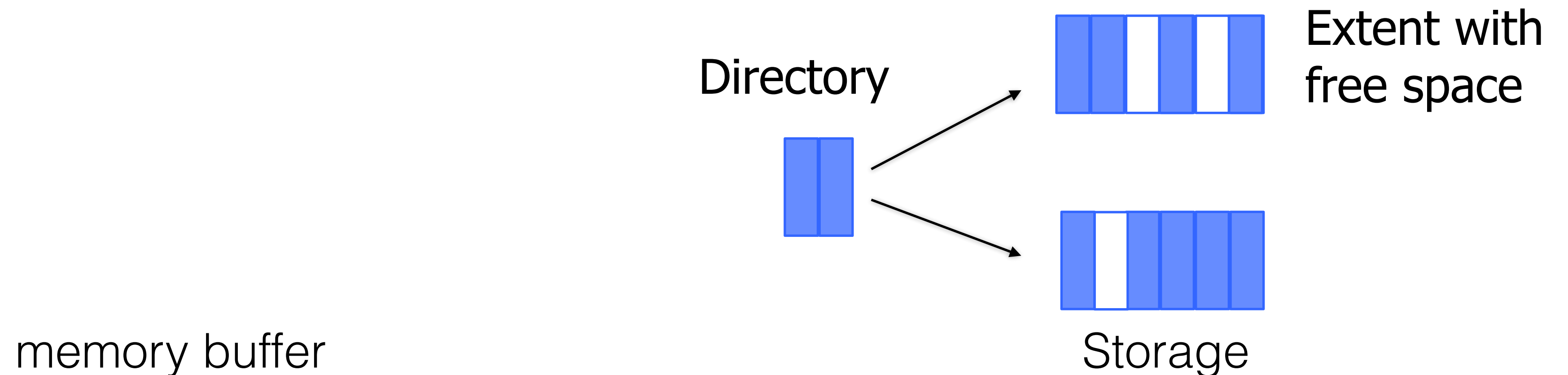
Consider a workload with only deletes and insertions. The deletes create “holes”.

Suppose that whenever the table consists of 50% holes, we rewrite it completely to save space.

Assume all data entries in the table are fixed-length, and that we have an oracle that tells us exactly where the entry we want to delete is, thus obviating the need to scan the table to find it.

Analyze the cost of both inserts and updates in the DAM model.

It is ok to analyze combined read & write I/Os.



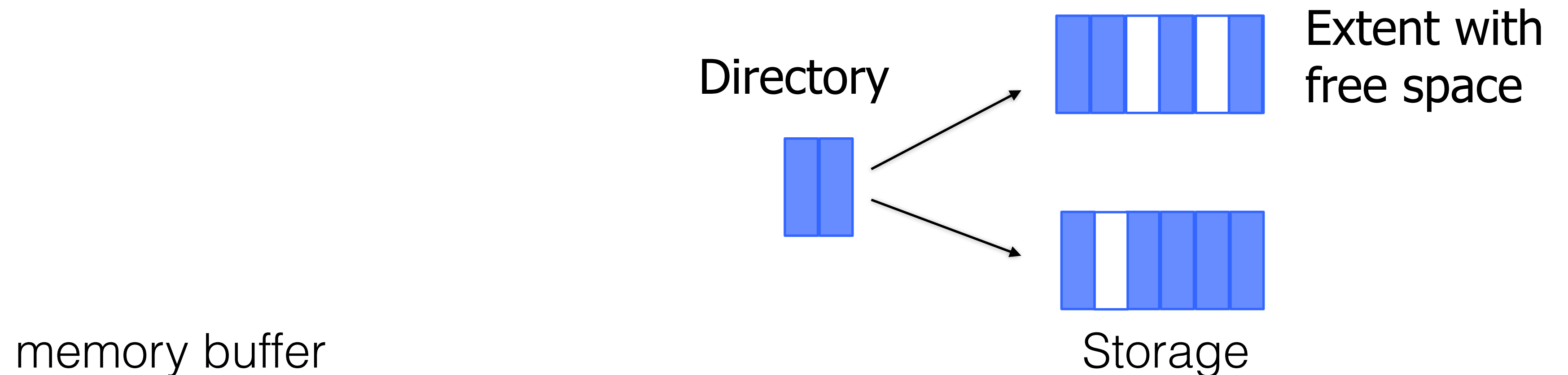
## Question 5

Insertion cost is still  $O(1/B)$  as we saw in class as deletes do not affect insertions.

Each in-place delete creates a hole at a cost of  $O(1)$  I/O.

For every  $\approx N/2$  deletes, we must read & write  $\approx N/B$  pages. This costs  $O(1/B)$  I/O per delete.

Total delete cost is  $O(1+1/B)$  or just  $O(1)$  I/O, dominated by having to delete in-place.



## Question 6

Consider a database that's subject to uniformly random reads.

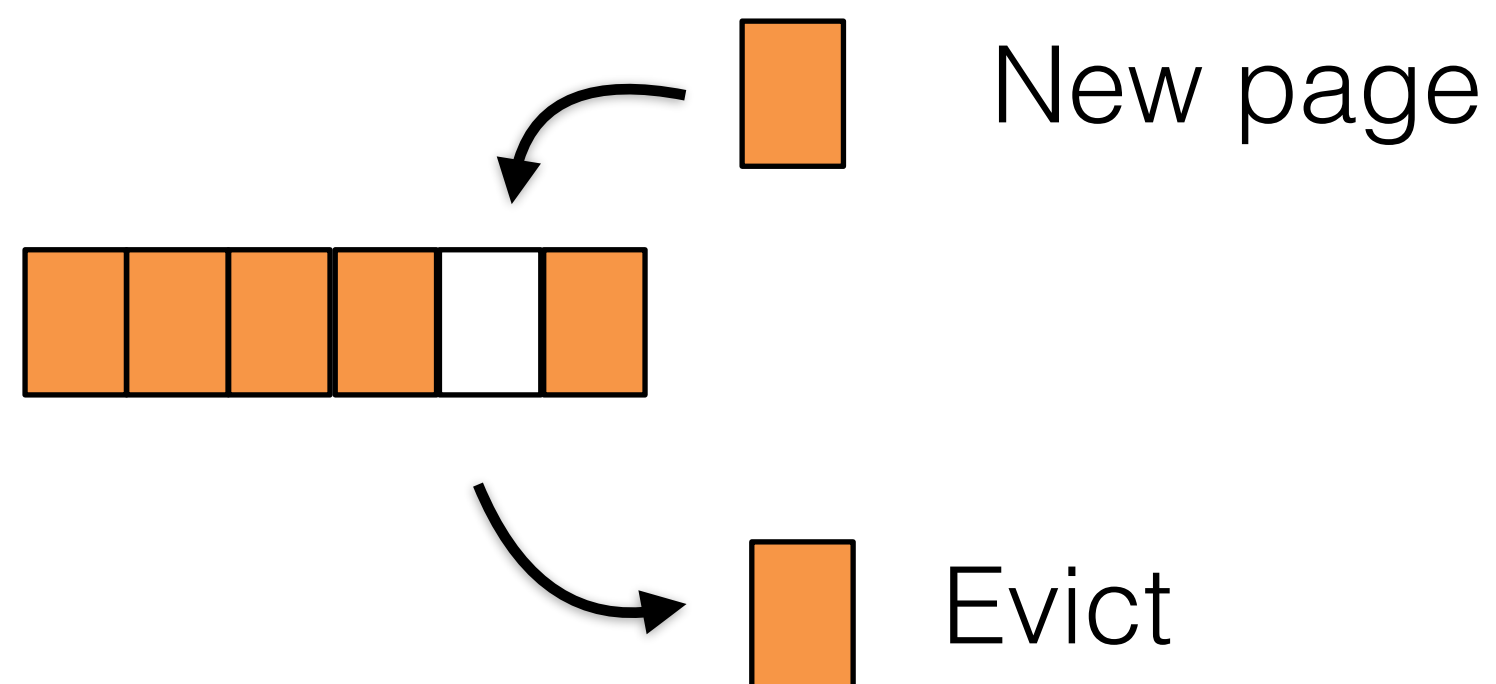
What's the best buffer management strategy for this case and why?

Would you also use this strategy if the reads are heavily skewed? Why or why not?

## Question 6

Consider a database that's subject to uniformly random reads.

What's the best buffer management strategy for this case and why?



### **Random eviction.**

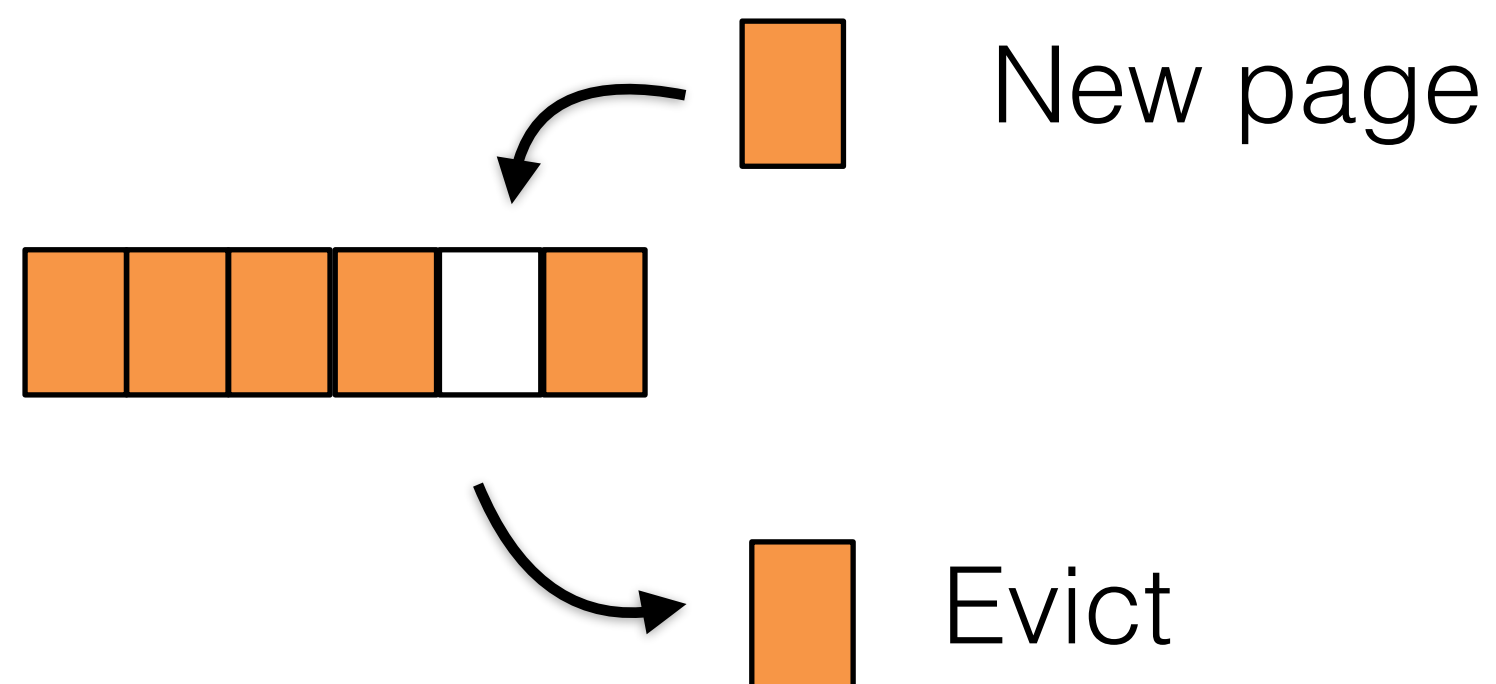
- (1) Allows 100% hash table utilization.
- (2) Fastest since hitting reads incur no collisions
- (3) There is no metadata so we can set the hash table to be slightly larger.

Would you also use this strategy if the reads are heavily skewed? Why or why not?

## Question 6

Consider a database that's subject to uniformly random reads.

What's the best buffer management strategy for this case and why?



### **Random eviction.**

- (1) Allows 100% hash table utilization.
- (2) Fastest since hitting reads incur no collisions
- (3) There is no metadata so we can set the hash table to be slightly larger.

Would you also use this strategy if the reads are heavily skewed? Why or why not?

**No because it might evict hot pages. Clock or LRU are better for this.**

## Question 7

Give pros and cons for managing buffer pool at the unit of entries rather than pages.

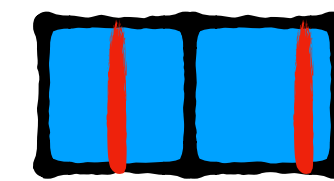
Pros:

Cons:

## Question 7

Give pros and cons for managing buffer pool at the unit of entries rather than pages.

Pros: (1) Some hot entries may exist on otherwise cold pages. Buffering entries allows us to fit more of the hot working set into memory. This is great for reads.



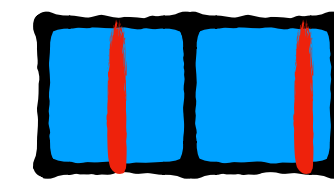
Cons:



## Question 7

Give pros and cons for managing buffer pool at the unit of entries rather than pages.

Pros: (1) Some hot entries may exist on otherwise cold pages. Buffering entries allows us to fit more of the hot working set into memory. This is great for reads.



Cons:

(1) Evicting a dirty entry requires reading its page first and then rewriting it

(2) Heavier metadata overheads (e.g., to track which page each entry belongs to, etc)

See you on Tuesday **in-person** for the start of indexing