# RAID

## Redundant Array of Independent Disks

Niv Dayan

We'll start at 3:10 pm

# RAID Addresses Three Problems

**Our database size exceeds one drive and we need more storage**

# RAID Addresses Three Problems

Our database size
exceeds one drive and
we need more storage

**A drive fails, and**
**we need to recover**
**its data**

# RAID Addresses Three Problems

Our database size exceeds one drive and we need more storage

A drive fails, and we need to recover its data


**We want to overcome the limits of one storage device bandwidth**

# Origin from 1988

## A Case for Redundant Arrays of Inexpensive Disks (RAID)

*David A. Patterson, Garth Gibson, and Randy H. Katz*

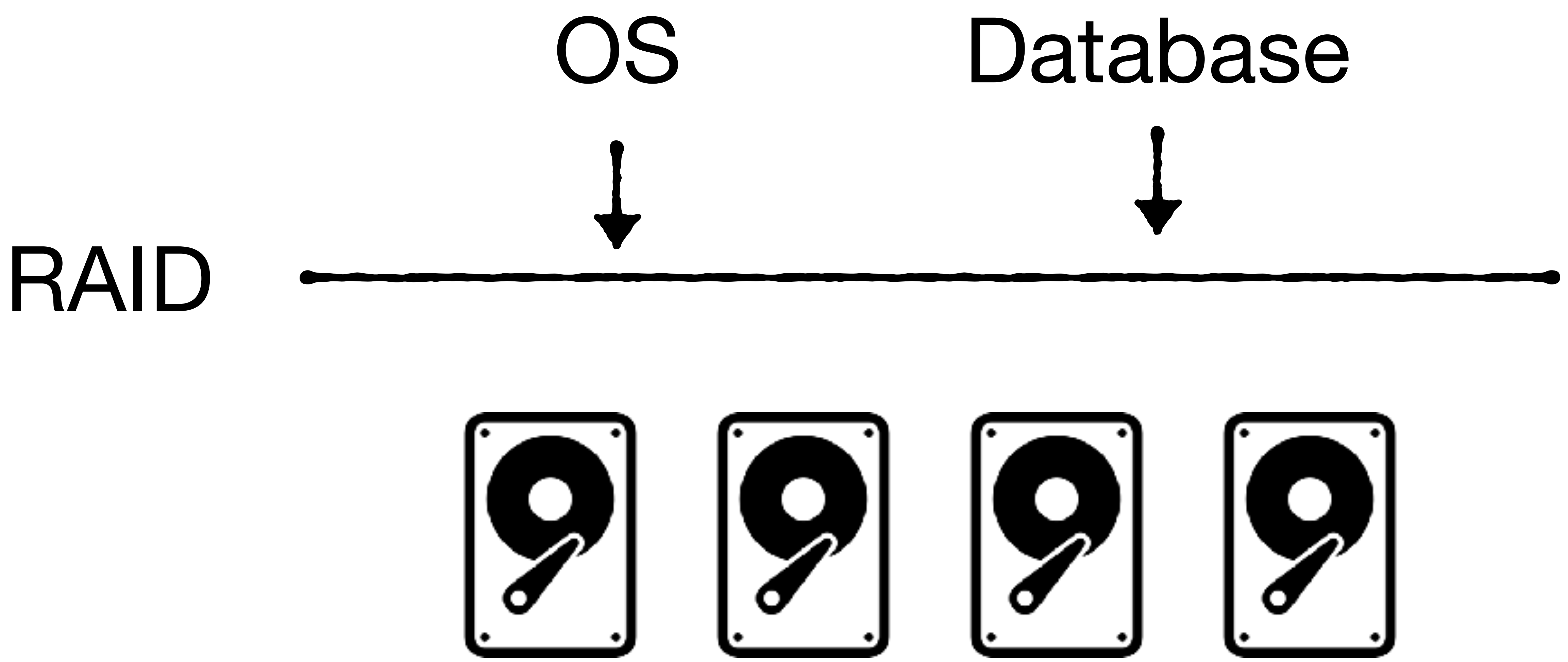
Computer Science Division
Department of Electrical Engineering and Computer Sciences
571 Evans Hall
University of California
Berkeley, CA 94720
(pattrsn@pepper.berkeley.edu)

*Abstract*

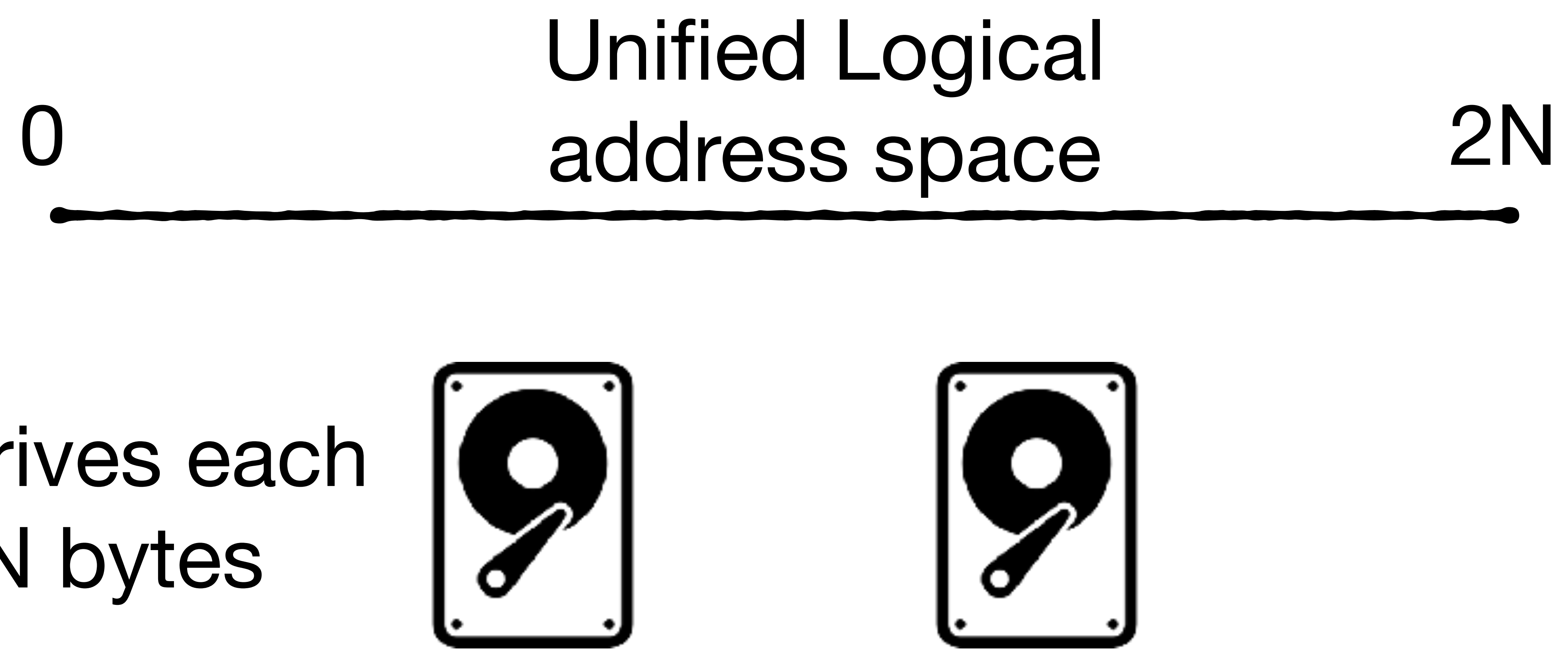
*Increasing performance of CPUs and memories will be squandered if not matched by a similar performance increase in I/O. While the capacity of Single Large Expensive Disk (SLED) has grown rapidly, the performance improvement of SLED has been modest. Redundant Arrays of Inexpensive Disks (RAID), based on the magnetic disk technology developed for personal computers, offers an attractive alternative to SLED, promising improvements of an order of magnitude in performance, reliability, power consumption, and scalability.*

*This paper introduces five levels of RAIDs, giving their relative cost/performance, and compares RAIDs to an IBM 3380 and a Fujitsu Super Eagle.*

## RAID stores data along with redundancy on multiple disks



## Expose a larger logical address space to OS

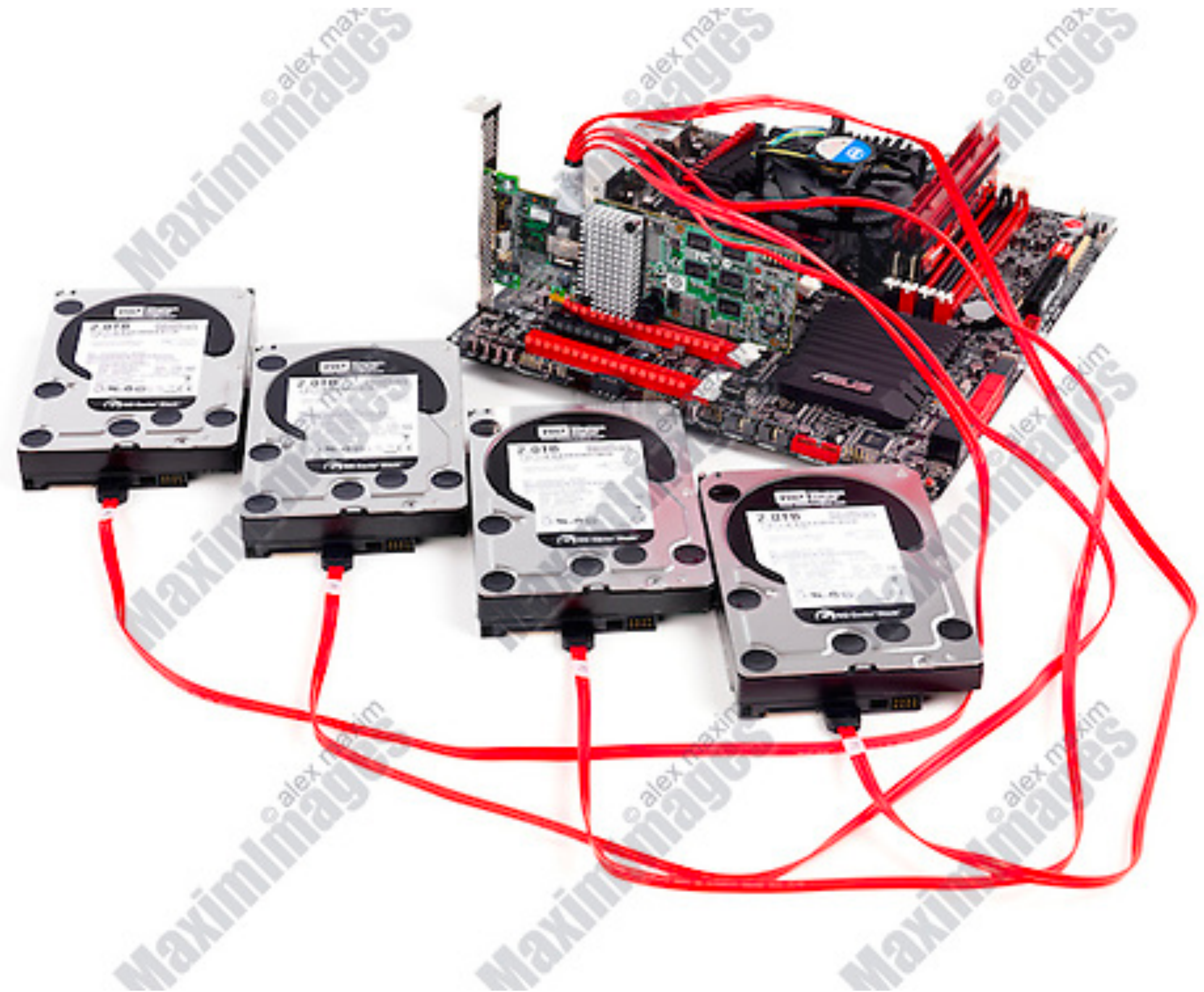



e.g., two drives each storing N bytes

**Looks to the OS like one drive, though consists of many**

# Can be implemented in…

## Hardware

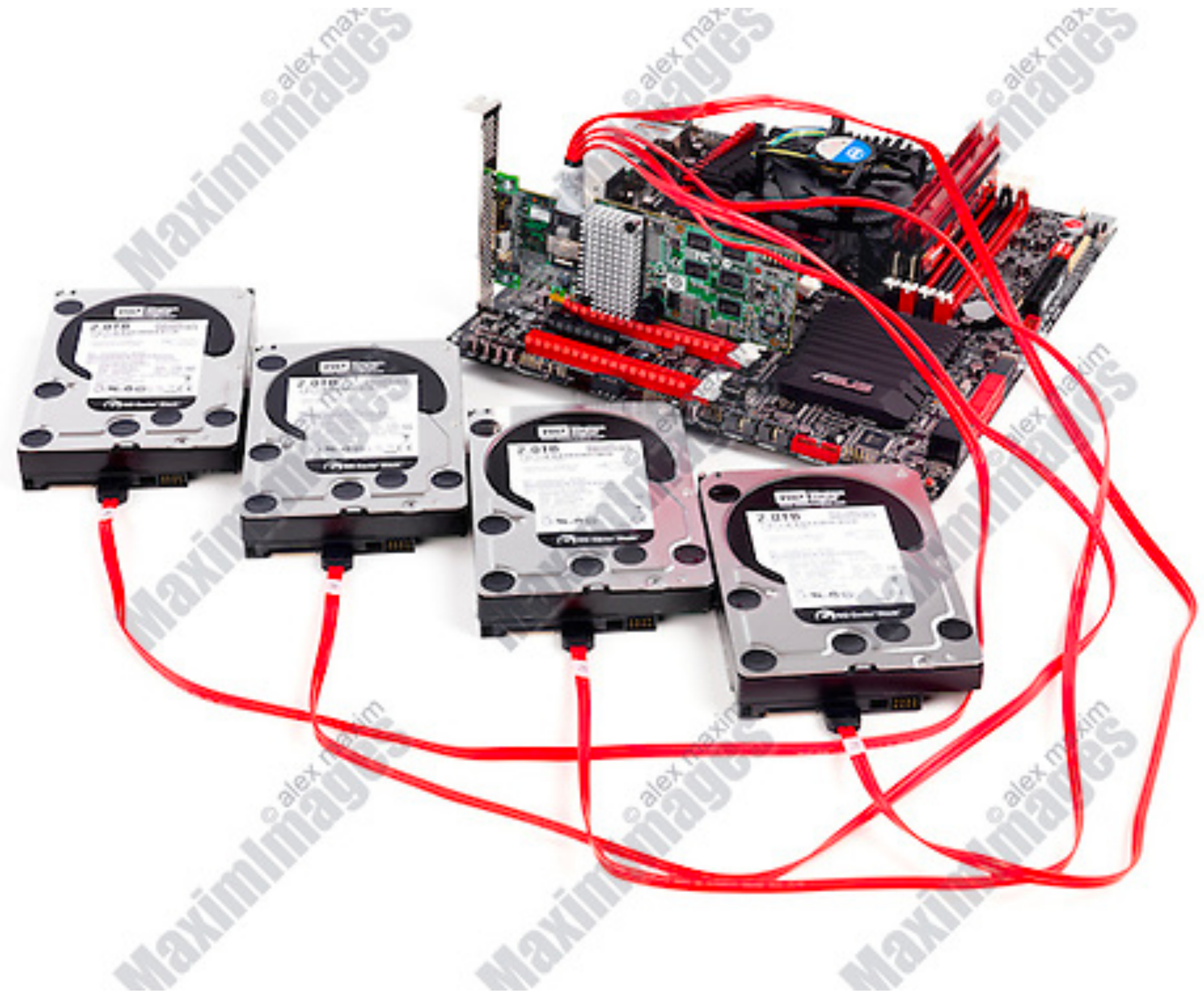


# Can be implemented in…

Hardware

Or

Software



# There are many RAID designs. We'll cover five

RAID 0 RAID 1 RAID 0+1 RAID 4 RAID 5

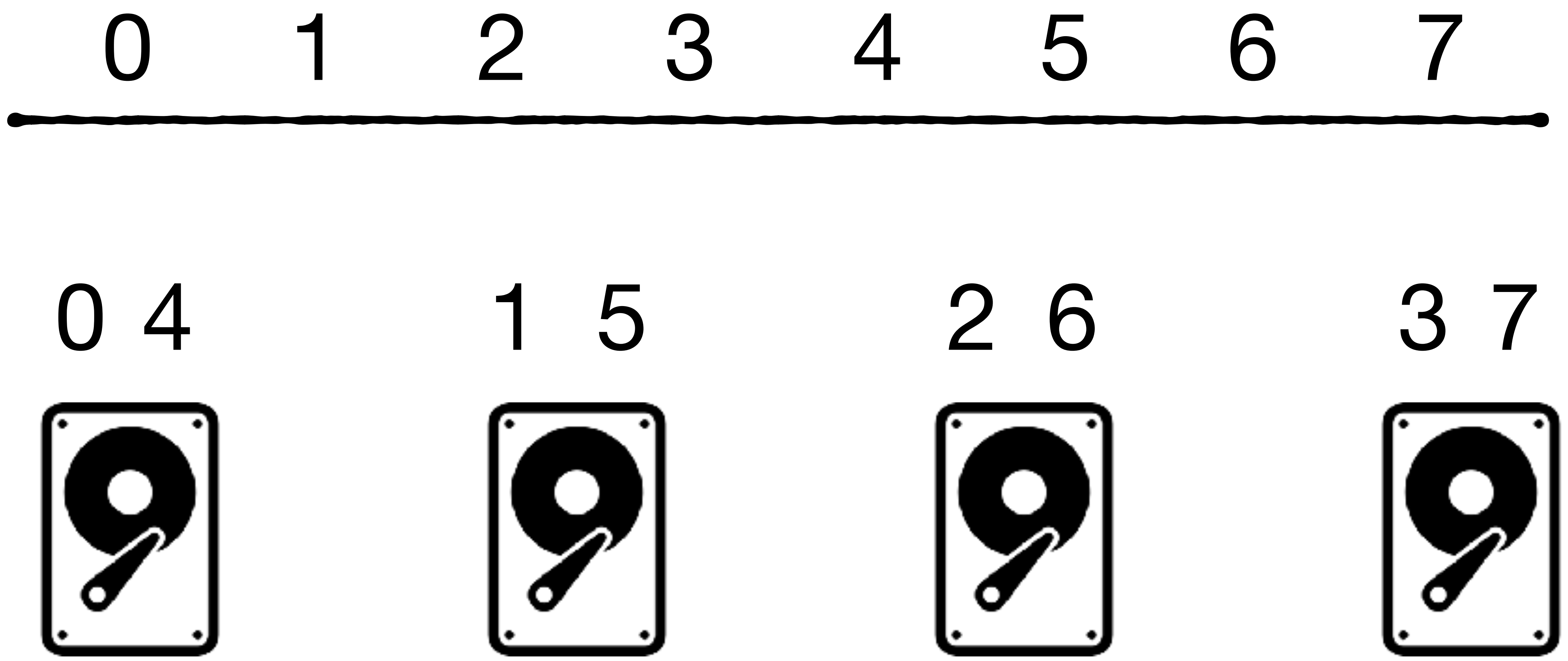
# RAID 0 - Pure striping

Stripe data in the logical address

# RAID 0 - Pure striping

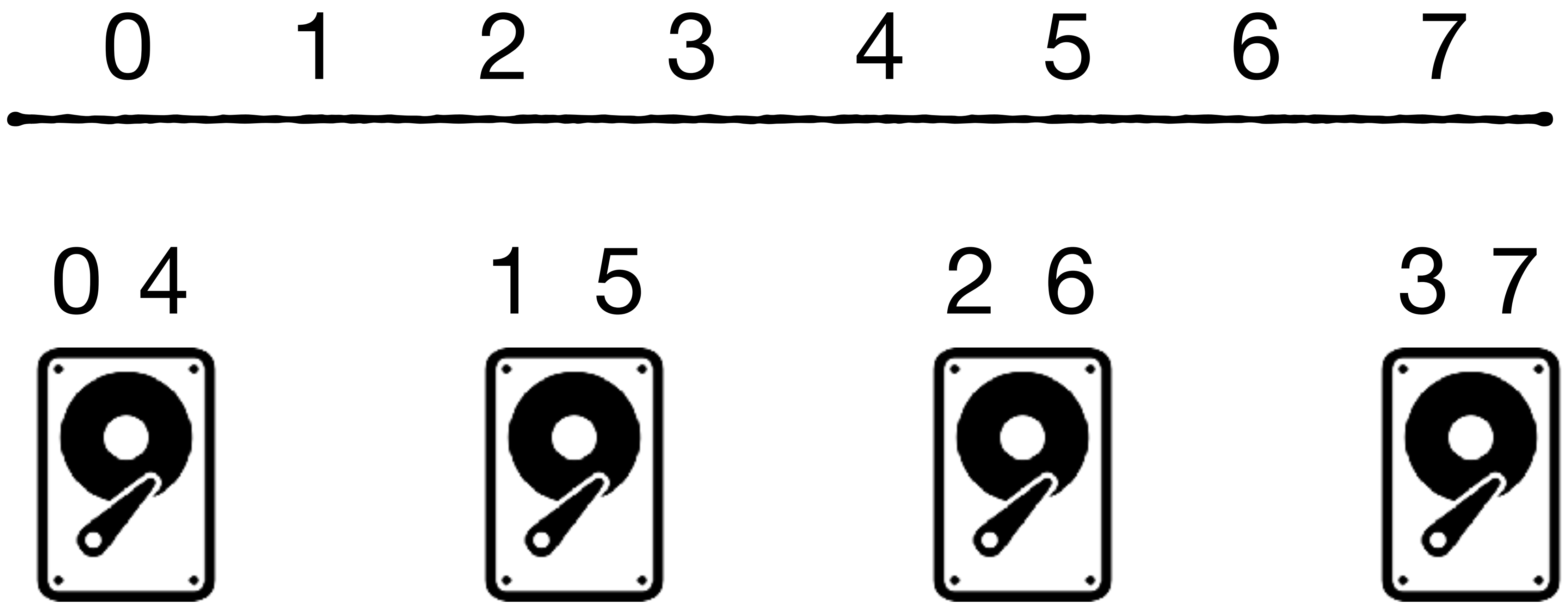
Stripe data in the logical address

Each number represents
a 512 B block address



# RAID 0 - Pure striping

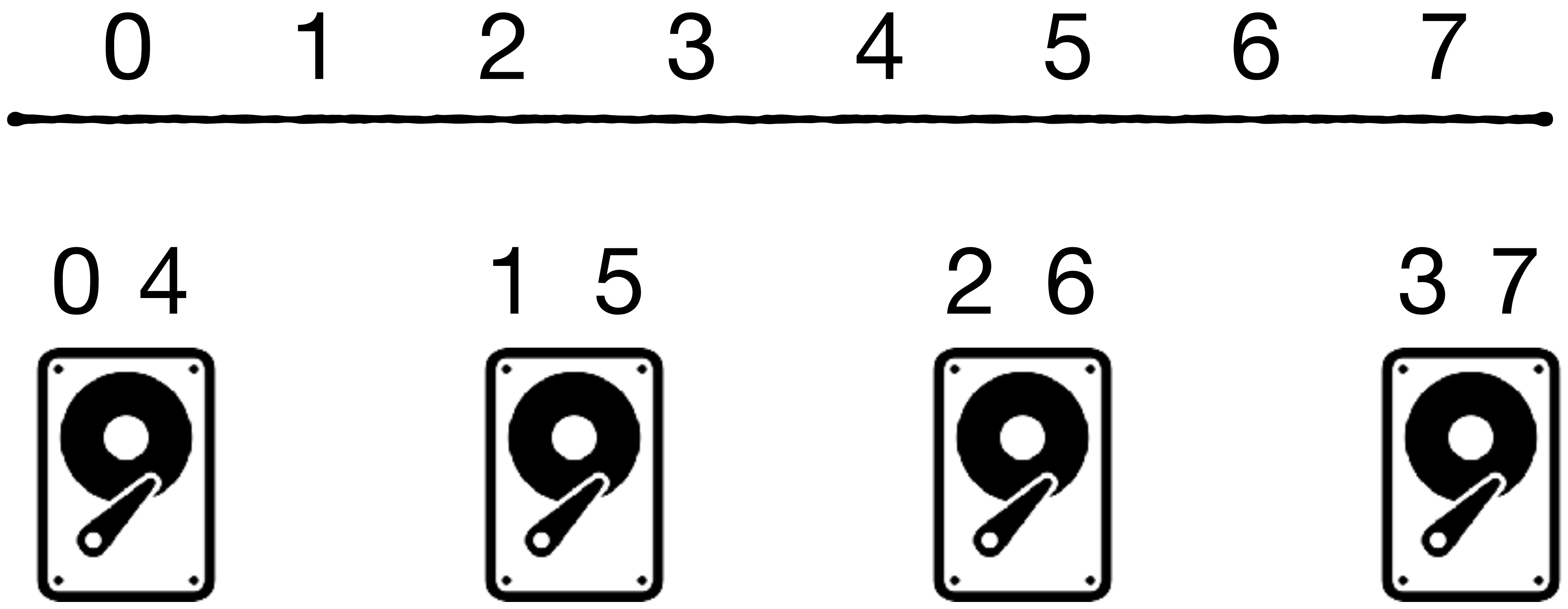
Stripe data in the logical address




1. Sequential reads / writes?
2. Random reads / writes?
3. Failure tolerance?

# RAID 0 - Pure striping

Stripe data in the logical address



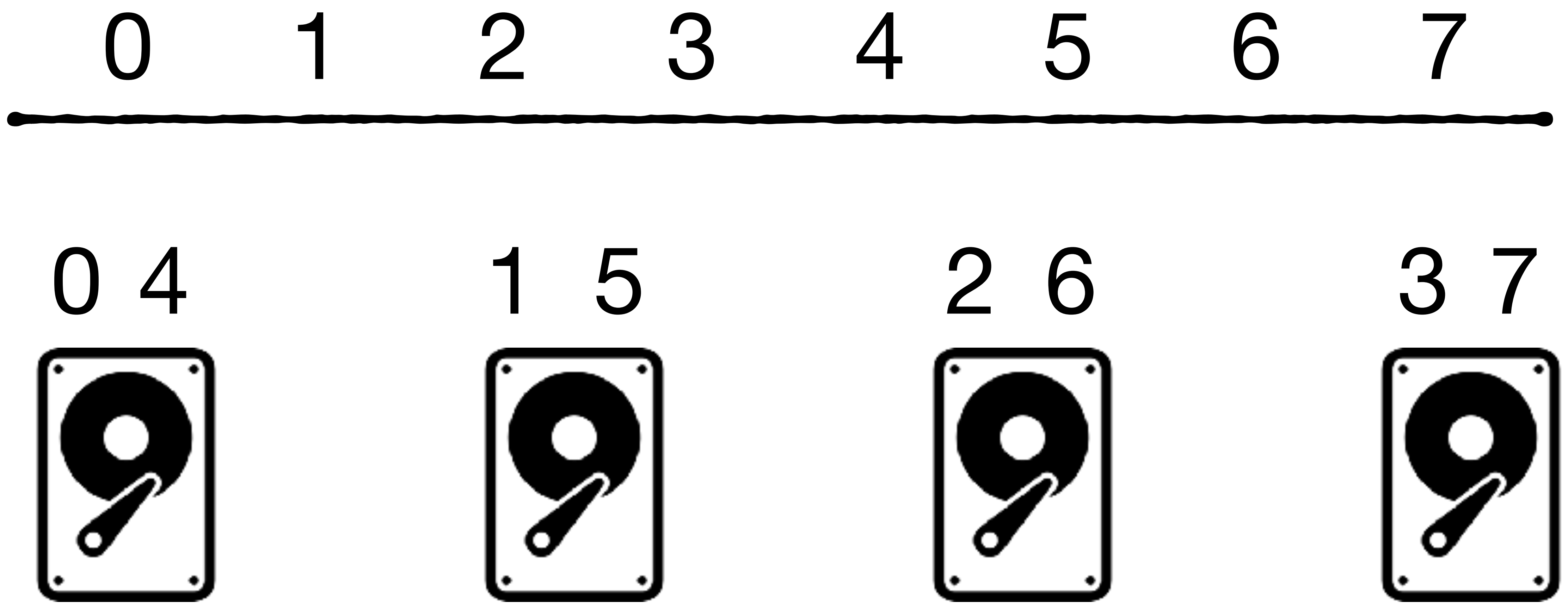

**1. Sequential reads / writes at combined bandwidth of all drives**

2. Random reads / writes?

3. Failure tolerance?

# RAID 0 - Pure striping

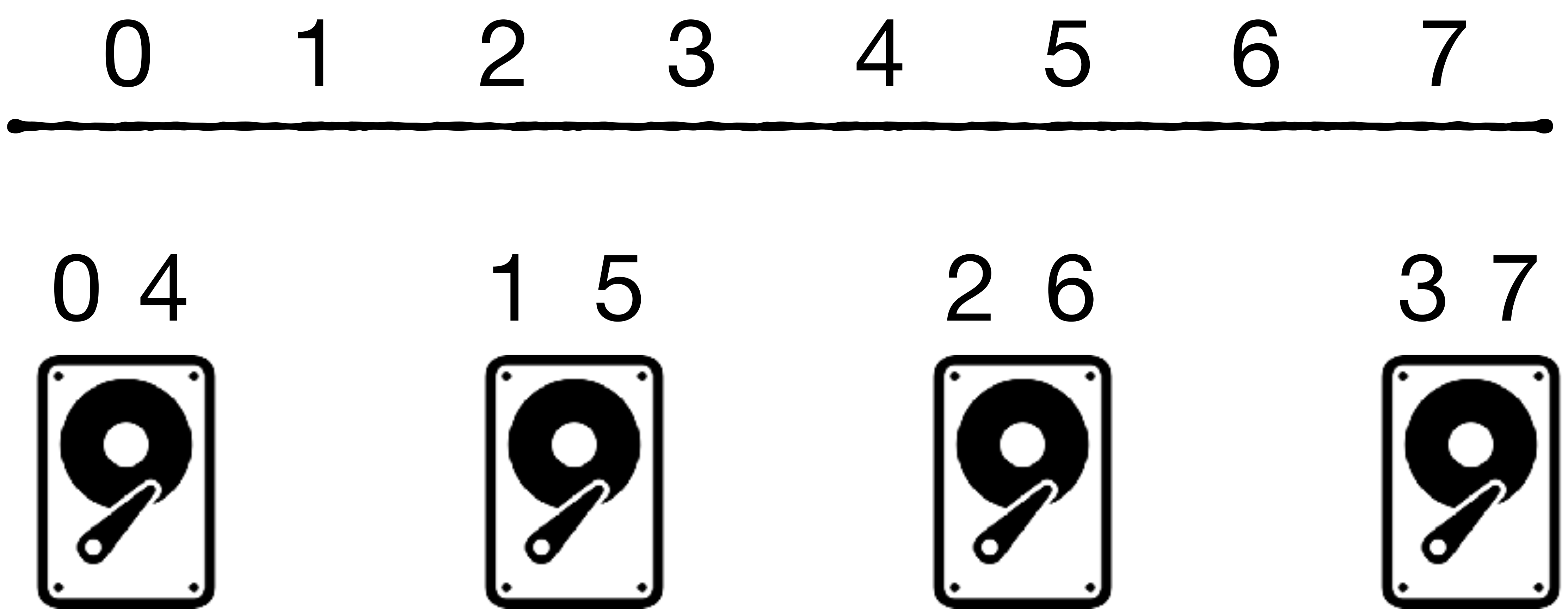
Stripe data in the logical address




1. Sequential reads / writes at combined bandwidth of all drives
2. **Faster random reads / writes due to load balancing**
3. Failure tolerance?

# RAID 0 - Pure striping

Stripe data in the logical address



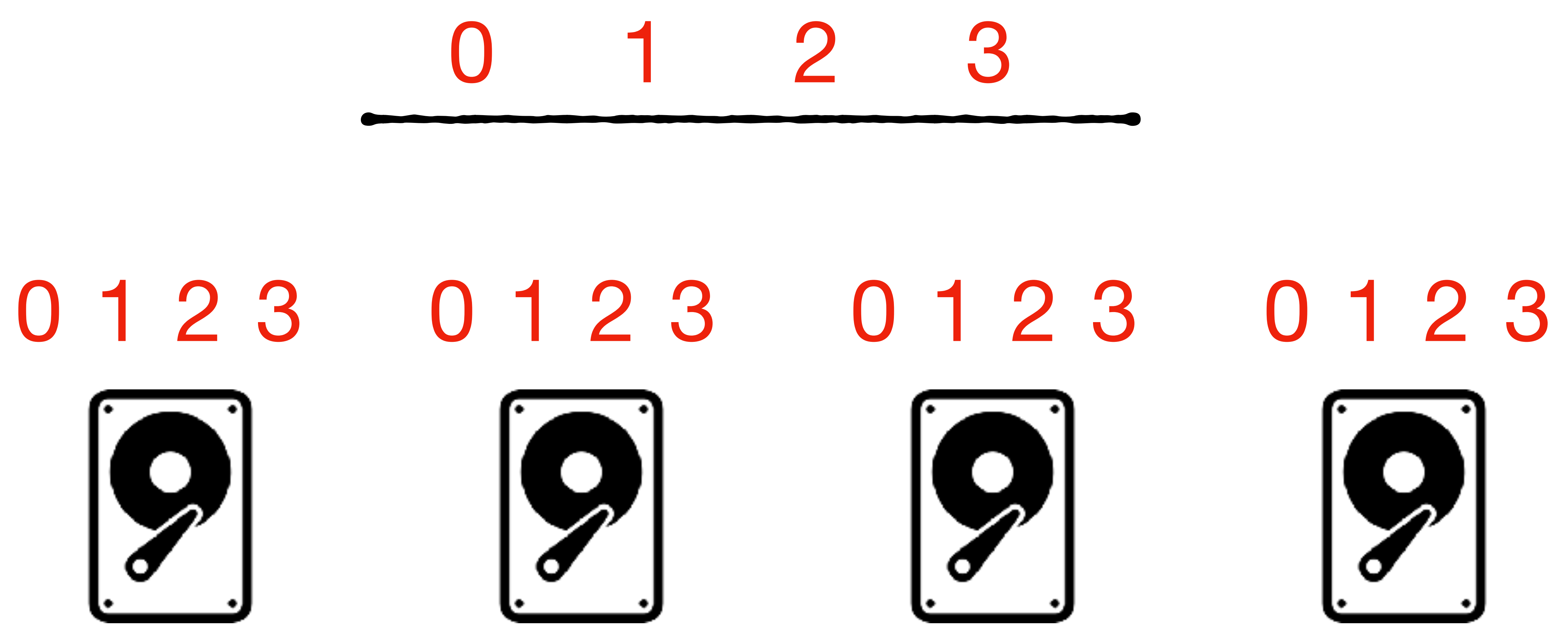

1. Sequential reads / writes at combined bandwidth of all drives
2. Faster random reads / writes due to load balancing
3. **Failure tolerance? No redundancy. If one disk fails, we lose data.**

RAID 0 **RAID 1** RAID 0+1 RAID 4 RAID 5

# RAID 1 - Mirroring

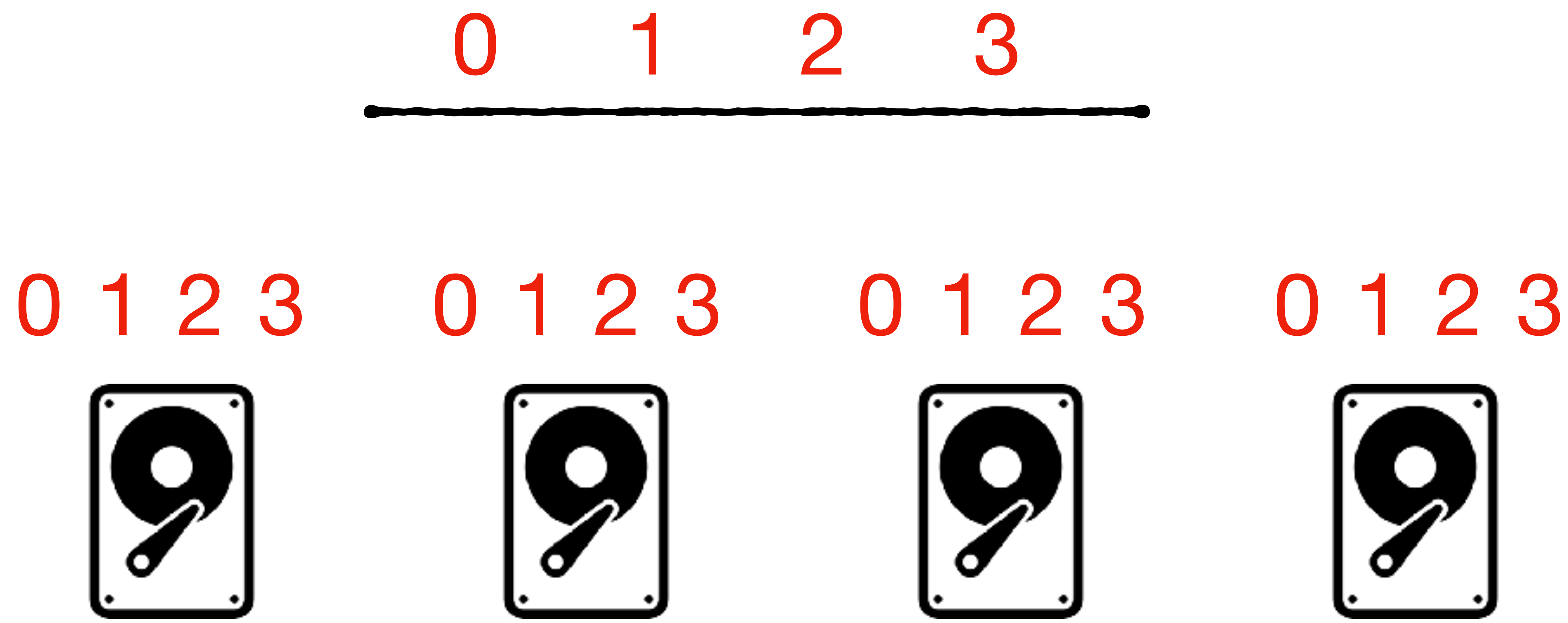
One drive is mirrored on as many drives as we have

Each number represents
a 512 B block address



# RAID 1 - Mirroring

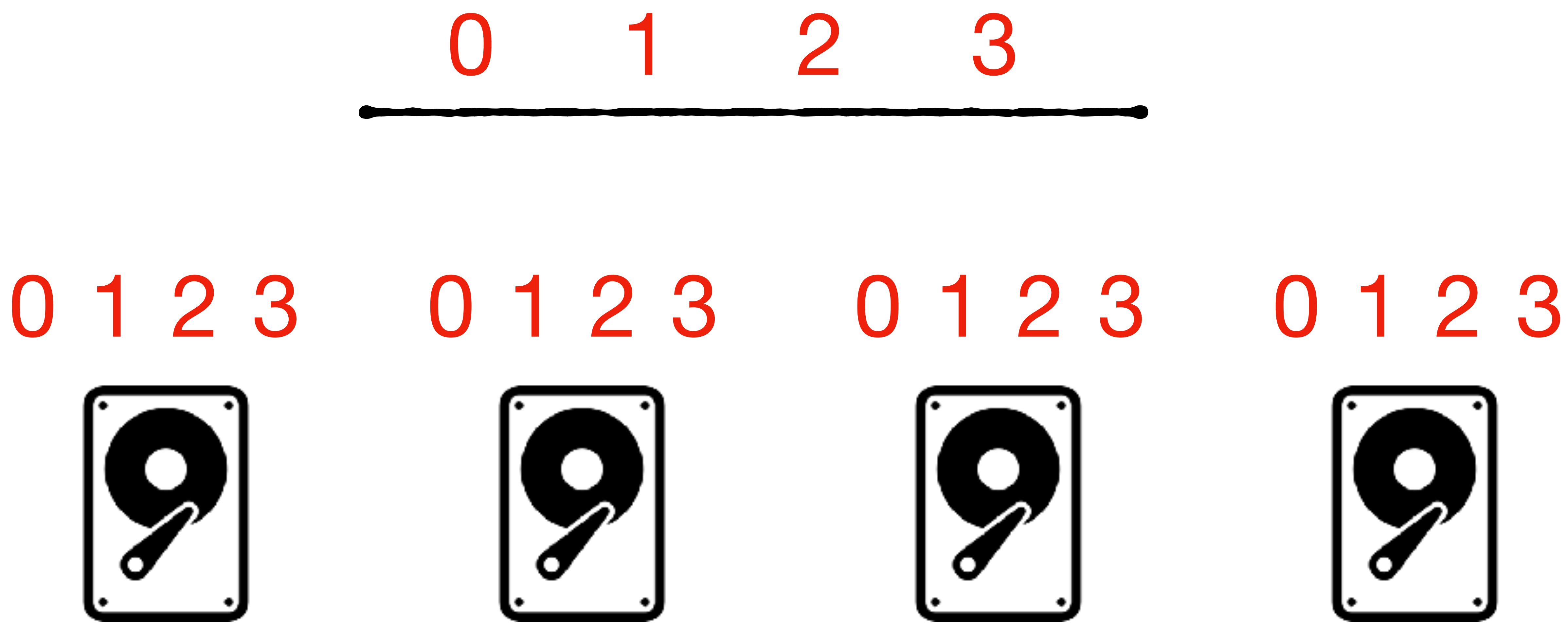
One drive is mirrored on as many drives as we have




1. Sequential writes?
2. Sequential reads?
3. Storage capacity?

## RAID 1 - Mirroring

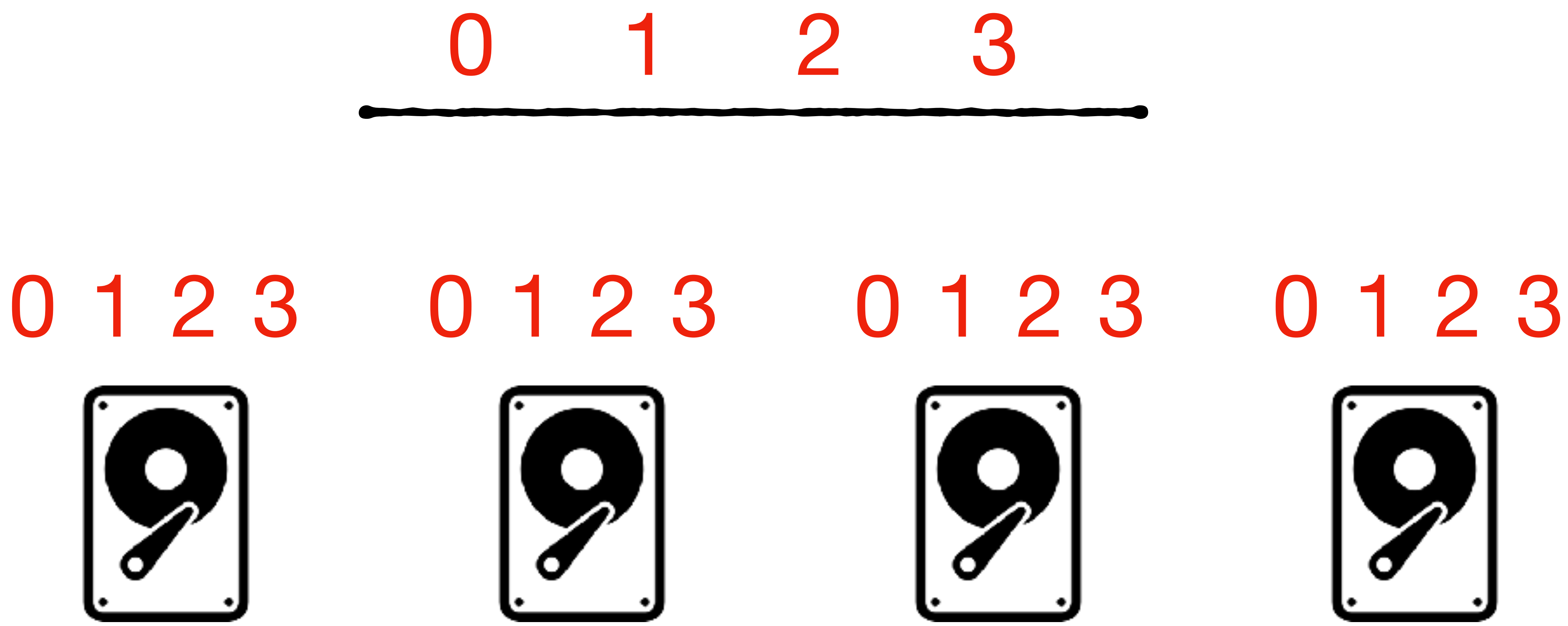
One drive is mirrored on as many drives as we have




1. **Sequential writes in the bandwidth of a single drive**
2. Sequential reads?
3. Storage capacity?

## RAID 1 - Mirroring

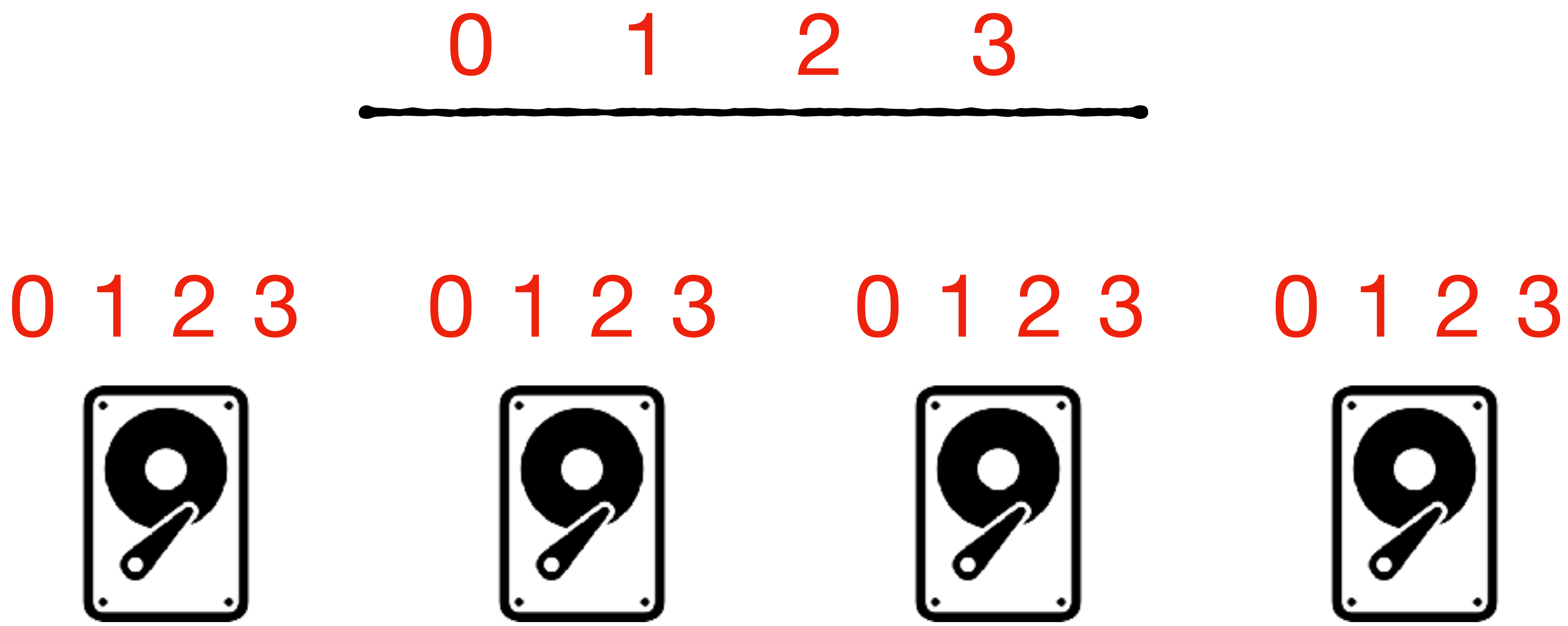
One drive is mirrored on as many drives as we have




1. Sequential writes in the bandwidth of a single drive
2. **Sequential reads in the combined bandwidth of all drives**
3. Storage capacity?

# RAID 1 - Mirroring

One drive is mirrored on as many drives as we have



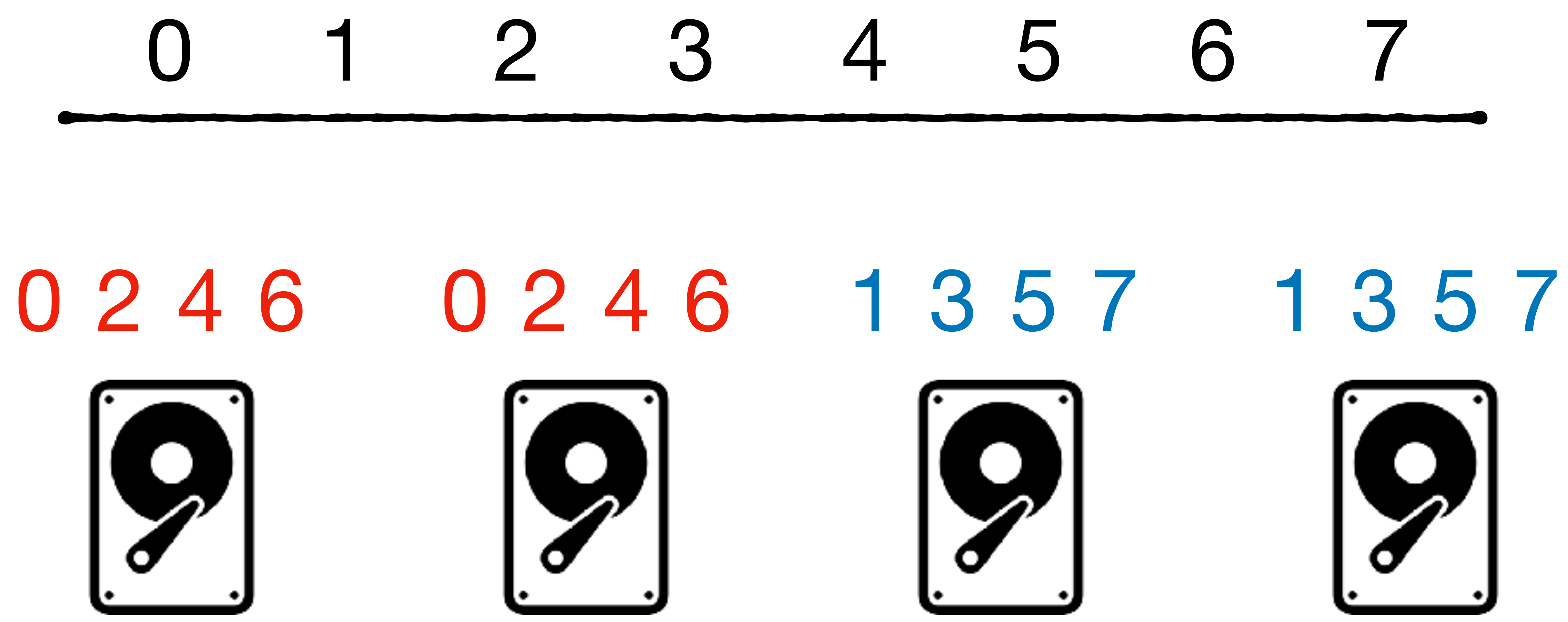

1. Sequential writes in the bandwidth of a single drive
2. Sequential reads in the combined bandwidth of all drives
3. **Storage capacity? Costs a lot…**

RAID 0 RAID 1 **RAID 0+1** RAID 4 RAID 5

# RAID 0+1 - Striping and Mirroring

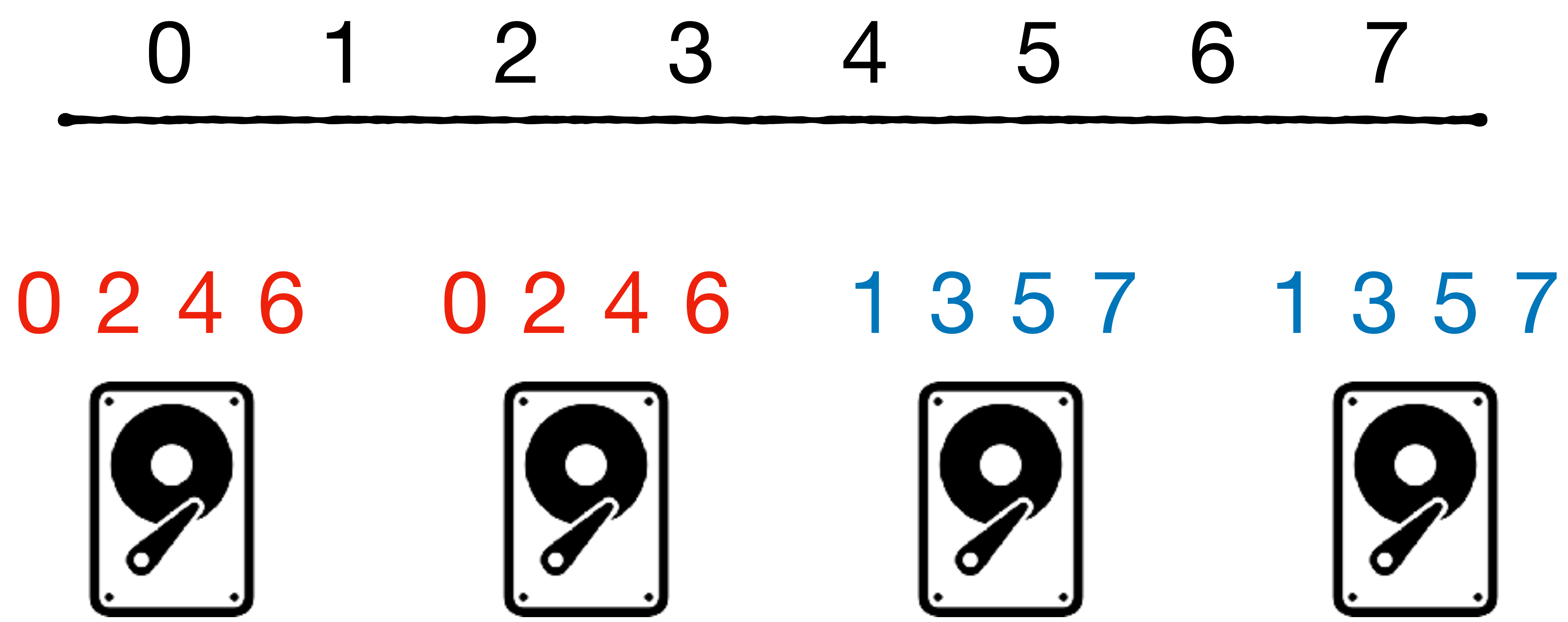
We stripe and mirror data at the same time

Each number represents
a 512 B block address




(This example assumes one mirror per disk, though this is in principle configurable)

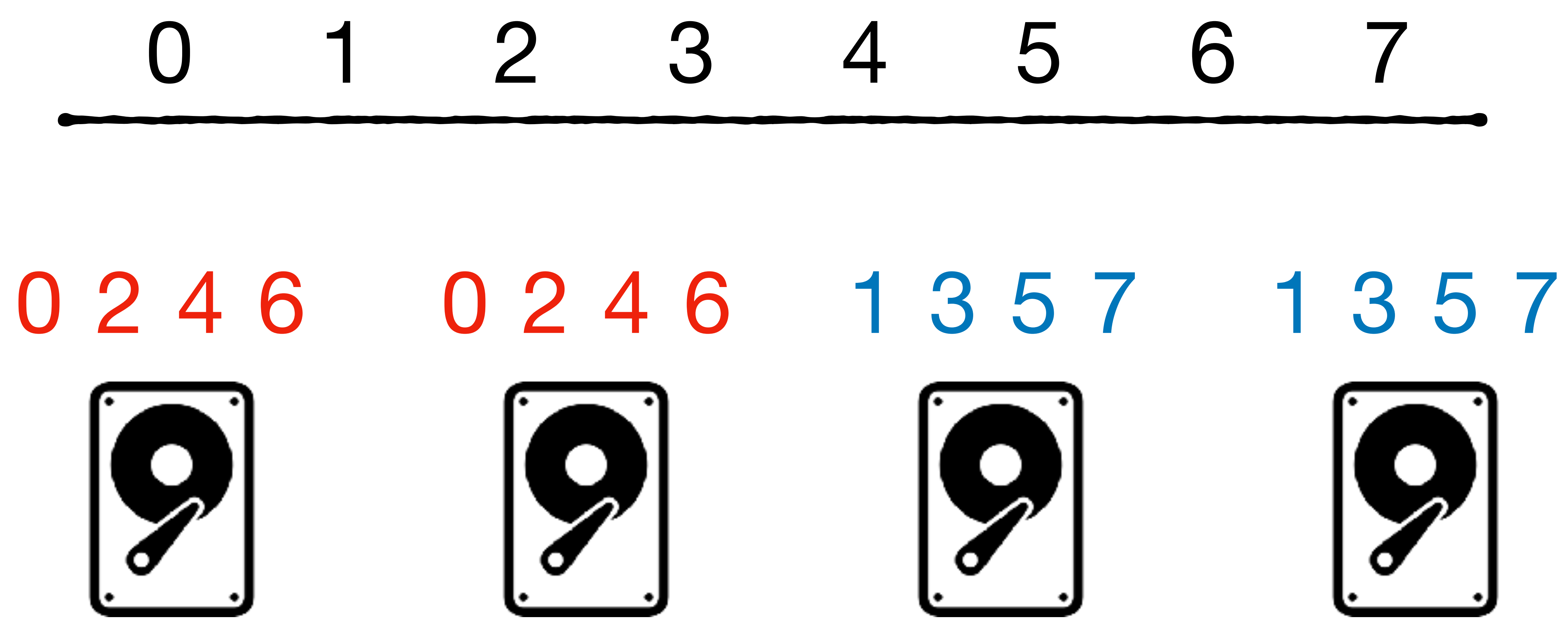
# RAID 0+1 - Striping and Mirroring




Assume N drives and X mirrors per drive

1. Sequential writes?
2. Sequential reads?
3. Storage capacity?

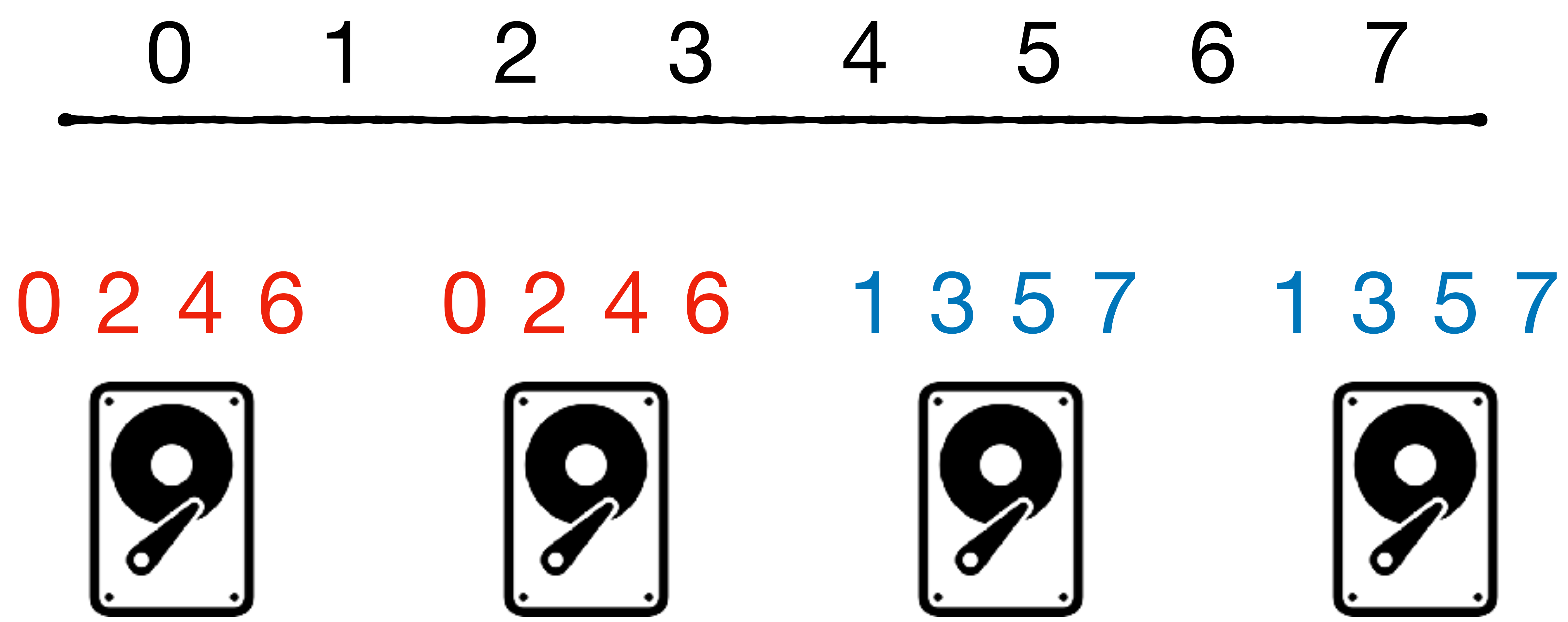
# RAID 0+1 - Striping and Mirroring




Assume N total drives and X drives per mirror group

1. **Sequential writes at combined bandwidth of N/X**
2. Sequential reads?
3. Storage capacity?

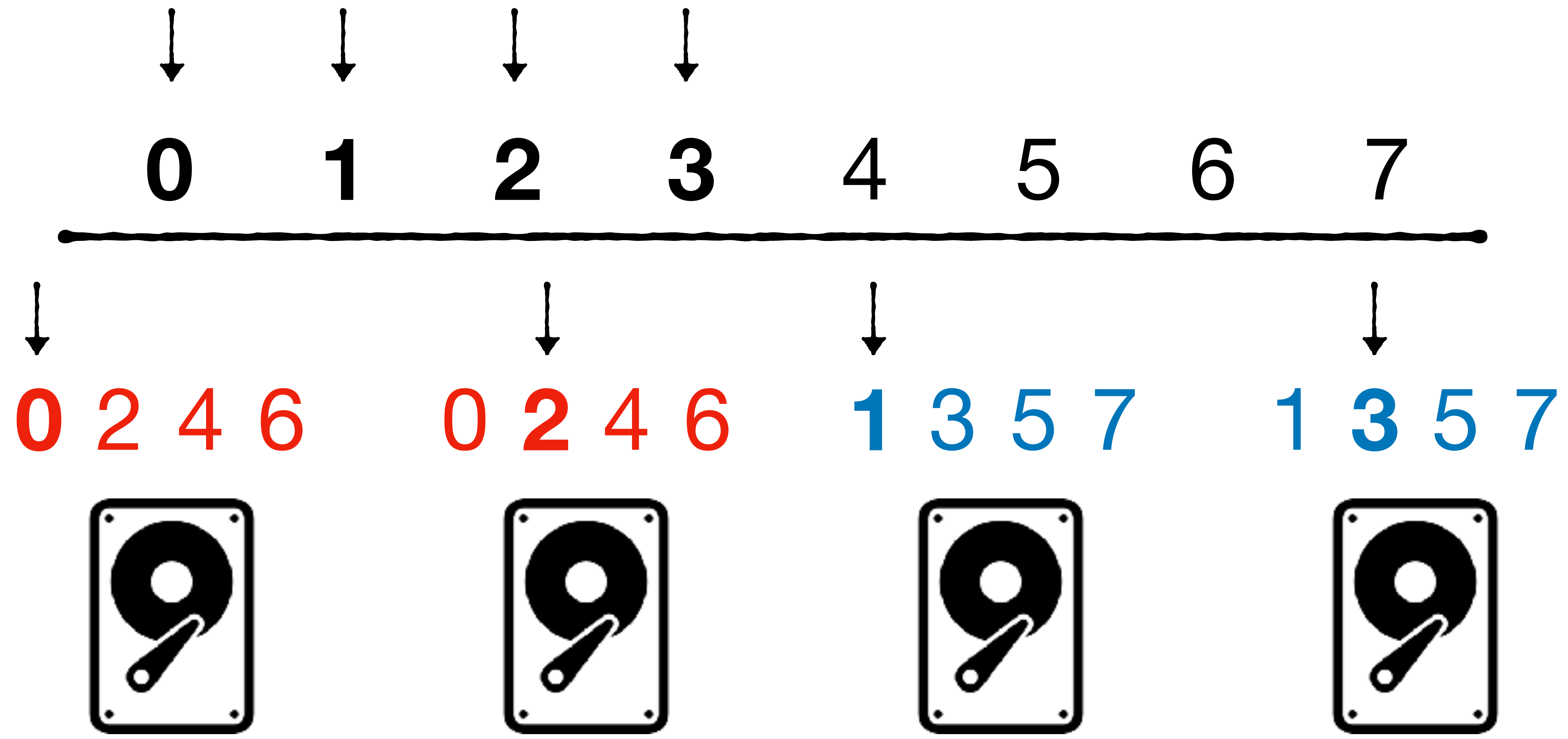
# RAID 0+1 - Striping and Mirroring




Assume N total drives and X drives per mirror group

1. Sequential writes at combined bandwidth of N/X
2. **Sequential reads at combined bandwidth of N**
3. Storage capacity?

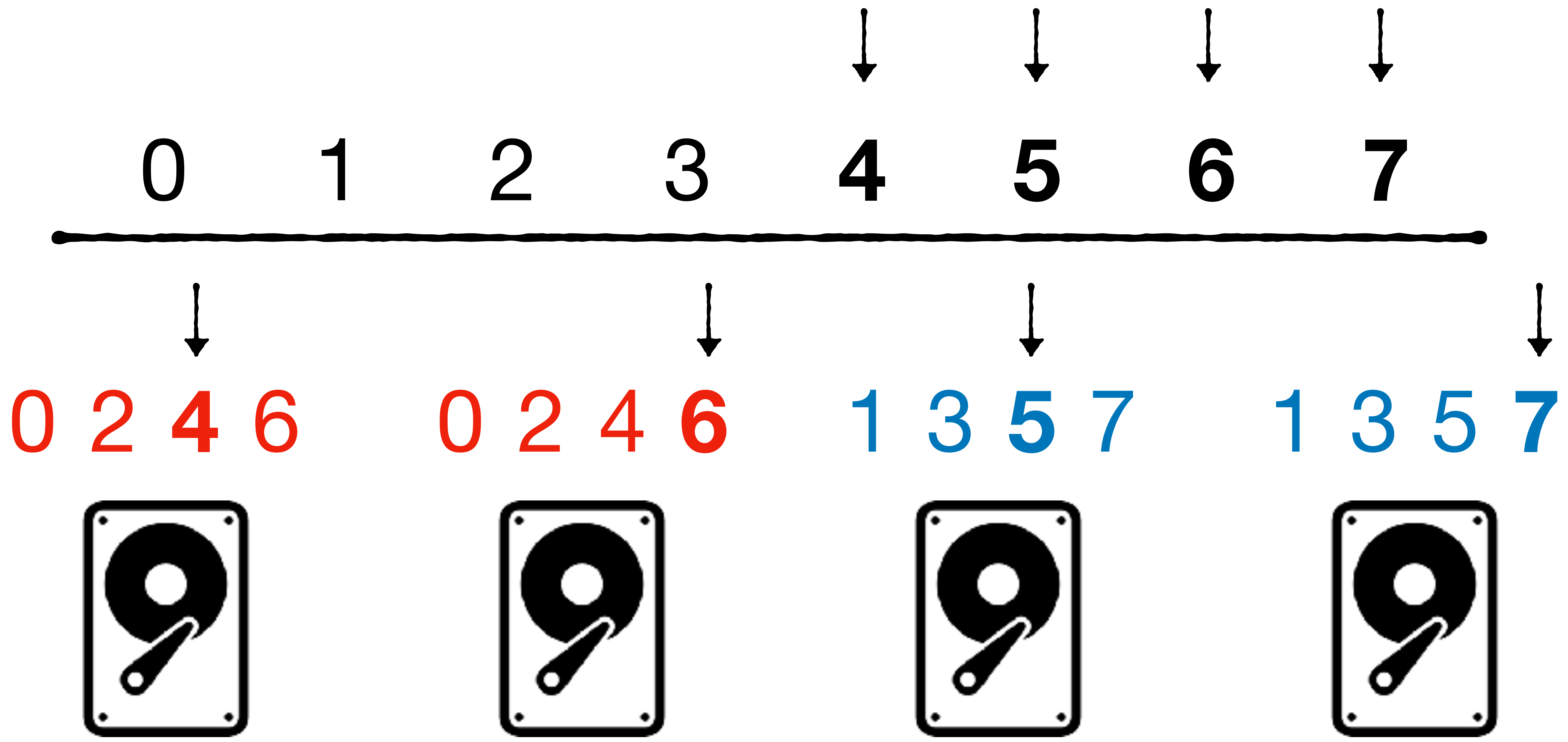
# RAID 0+1 - Striping and Mirroring




Assume N total drives and X drives per mirror group

1. Sequential writes at combined bandwidth of N/X
2. **Sequential reads at combined bandwidth of N**
3. Storage capacity?

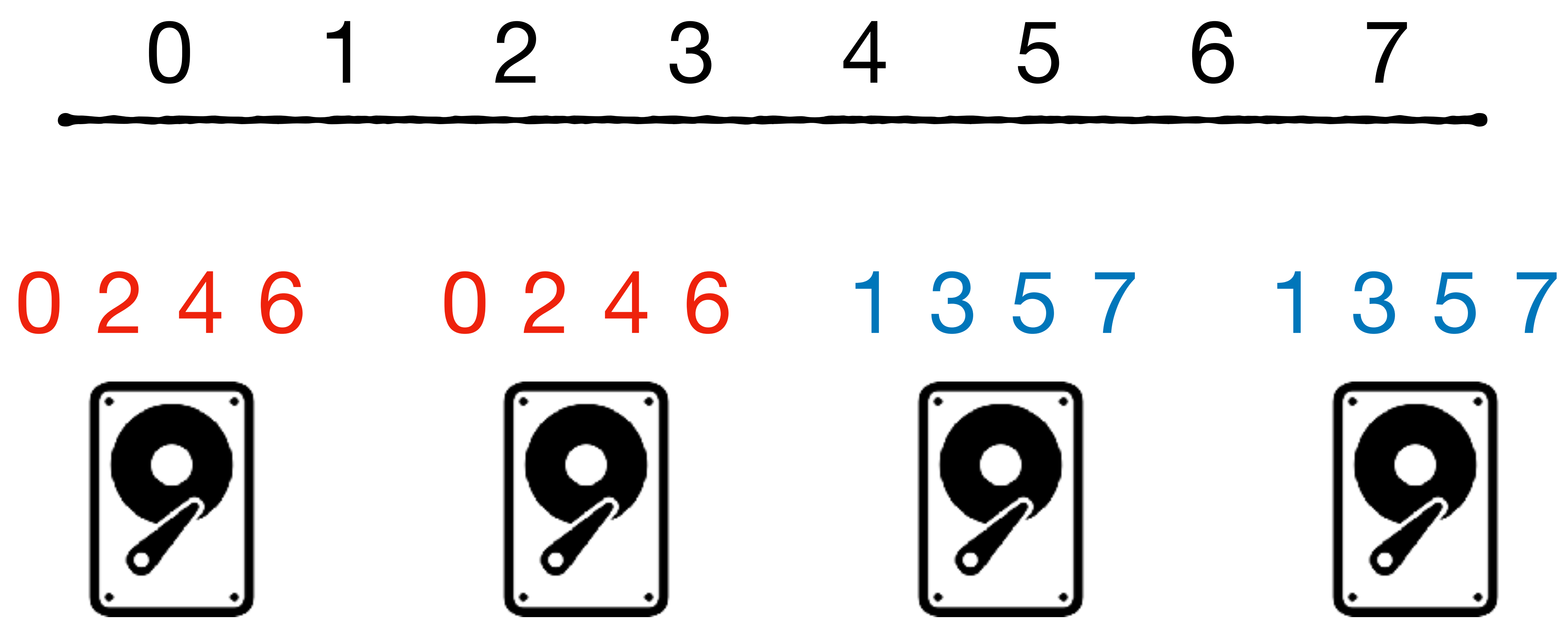
# RAID 0+1 - Striping and Mirroring




Assume N total drives and X drives per mirror group

1. Sequential writes at combined bandwidth of N/X
2. **Sequential reads at combined bandwidth of N**
3. Storage capacity?

## RAID 0+1 - Striping and Mirroring




Assume N total drives and X drives per mirror group

1. Sequential writes at combined bandwidth of N/X
2. Sequential reads at combined bandwidth of N
3. **Storage capacity? 1/X of total capacity**

RAID 0 RAID 1 RAID 0+1 **RAID 4** RAID 5

# RAID 4

## Goal: using less than 50% of the storage capacity to allow recovery

# RAID 4

Goal: using less than 50% of the storage capacity to allow recovery

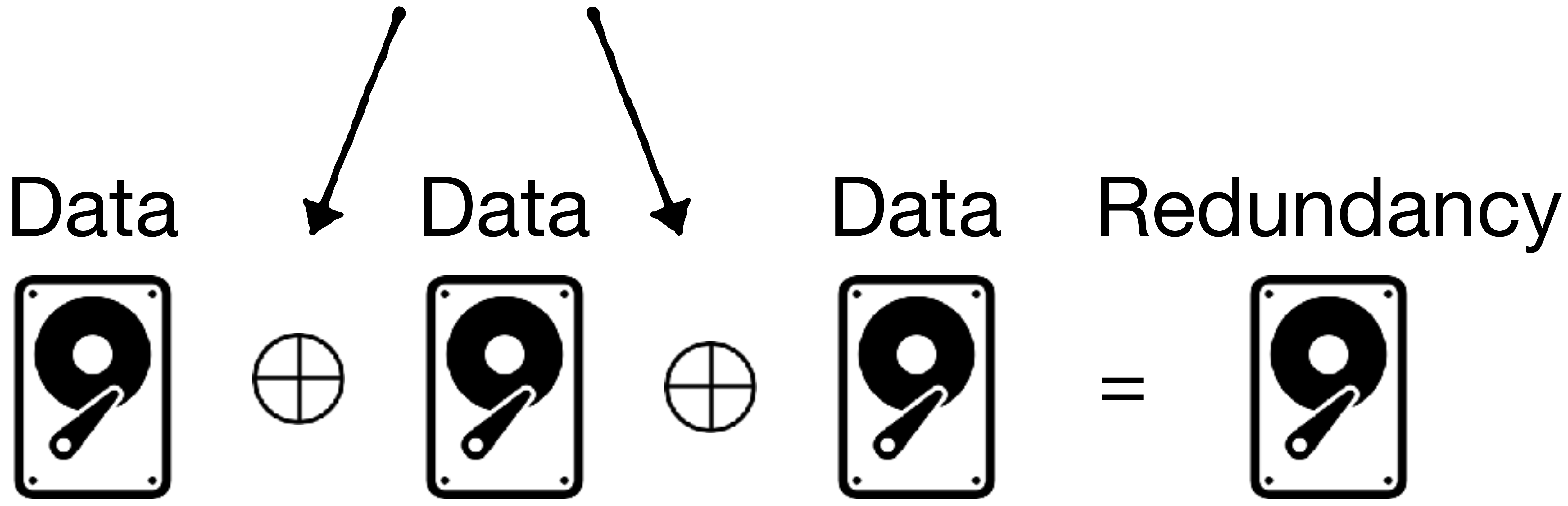
**Challenge: how can we use one drive to recover any drive that fails?**

# RAID 4

Goal: using less than 50% of the storage capacity to allow recovery

Challenge: how can we use one drive to recover any drive that fails?

## XOR Operator



# XOR Operator

| | | | | |
|---|---|---|---|---|
| **Input 1** | 0 | 0 | 1 | 1 |

$$\oplus$$

| | | | | |
|---|---|---|---|---|
| **Input 2** | 0 | 1 | 0 | 1 |

$$=$$

| | | | | |
|---|---|---|---|---|
| **Parity** | 0 | 1 | 1 | 0 |

# XOR Operator

| | | | | |
|---|---|---|---|---|
| **Input 1** | 0 | 0 | 1 | 1 |
| | | $\oplus$ | | |
| **Input 2** | 0 | 1 | 0 | 1 |
| | | $=$ | | |
| **Parity** | 0 | 1 | 1 | 0 |

If number of 1s in an input column is even, the result is 0. If odd, it is 1.

# XOR Operator

| | | | | |
|---|---|---|---|---|
| **Input 1** | 0 | 0 | 1 | 1 |
| | | $\oplus$ | | |
| **Input 2** | 0 | 1 | 0 | 1 |
| | | = | | |
| | 0 | 1 | 1 | 0 |

**Parity:** the condition of the number of items in a set, particularly the number of bits per byte or word, being either even or odd: used as a means for detecting certain errors.

Dictionary definition

# Parity can help recover any input

**Suppose we lost input 2**

| | | | | |
|---|---|---|---|---|
| Input 1 | 0 | 0 | 1 | 1 |
| Parity | 0 | 1 | 1 | 0 |

# Parity can help recover any input

**Suppose we lost input 2**

| | | | | | |
|---|---|---|---|---|---|
| Input 1 | 0 | 0 | 1 | 1 | |
| | | $\oplus$ | | | |
| Parity | 0 | 1 | 1 | 0 | |
| | | $=$ | | | |
| Input 2 | 0 | 1 | 0 | 1 | **Recovered** |

# Parity can help recover any input

**Or suppose we lost input 1**

| | | | | |
|---|---|---|---|---|
| Input 2 | 0 | 1 | 0 | 1 |
| Parity | 0 | 1 | 1 | 0 |

# Parity can help recover any input

**Or suppose we lost input 1**

| | | | | |
|---|---|---|---|---|
| Input 2 | 0 | 1 | 0 | 1 |
| | | $\oplus$ | | |
| Parity | 0 | 1 | 1 | 0 |
| | | = | | |
| Input 1 | 0 | 0 | 1 | 1 |

**Recovered**

# XOR is commutative and associative

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Input 1** | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | | | | $\oplus$ | | | | |
| **Input 2** | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | | | | $\oplus$ | | | | |
| **Input 3** | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | | | | = | | | | |
| **Parity** | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |

# XOR is commutative and associative

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Input 1** | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | | | | $\oplus$ | | | | |
| **Input 2** | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | | | | $\oplus$ | | | | |
| **Input 3** | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | | | | = | | | | |
| **Parity** | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |

**Parity can recover any input, as long as we also have all the other inputs**

# What happens if we xor input 1 with input 2 twice?

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Input 1** | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | | | | | $\oplus$ | | | |
| **Input 2** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** |
| | | | | | $\oplus$ | | | |
| **Input 2** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** |

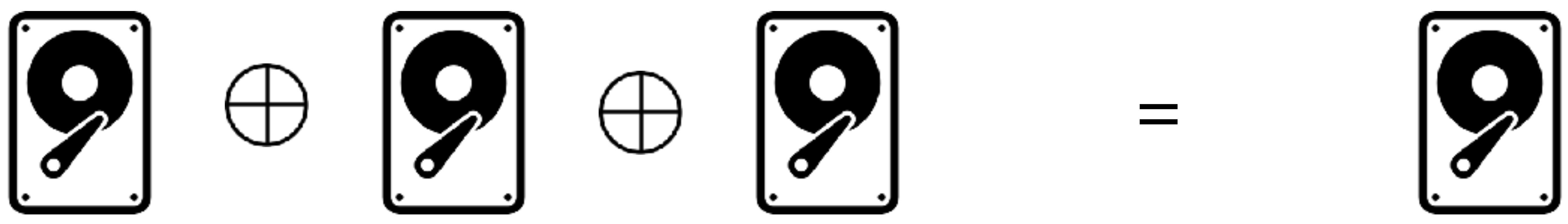
# XORing with something twice cancels it

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Input 1** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** |
| | | | | $\oplus$ | | | | |
| **Input 2** | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | | | | $\oplus$ | | | | |
| **Input 2** | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | | | | = | | | | |
| **Input 1** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** |

# RAID 4 - Parity Drive

Given N drives, stripe on N-1 drives and use 1 drive for parity

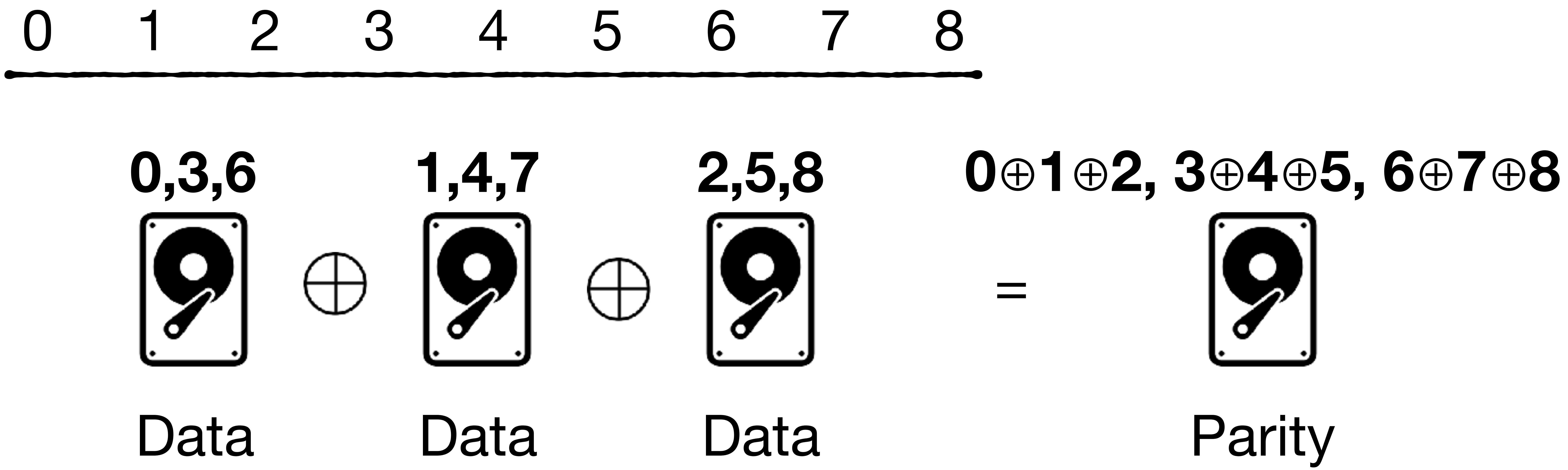
0 1 2 3 4 5 6 7 8



Data Data Data Parity

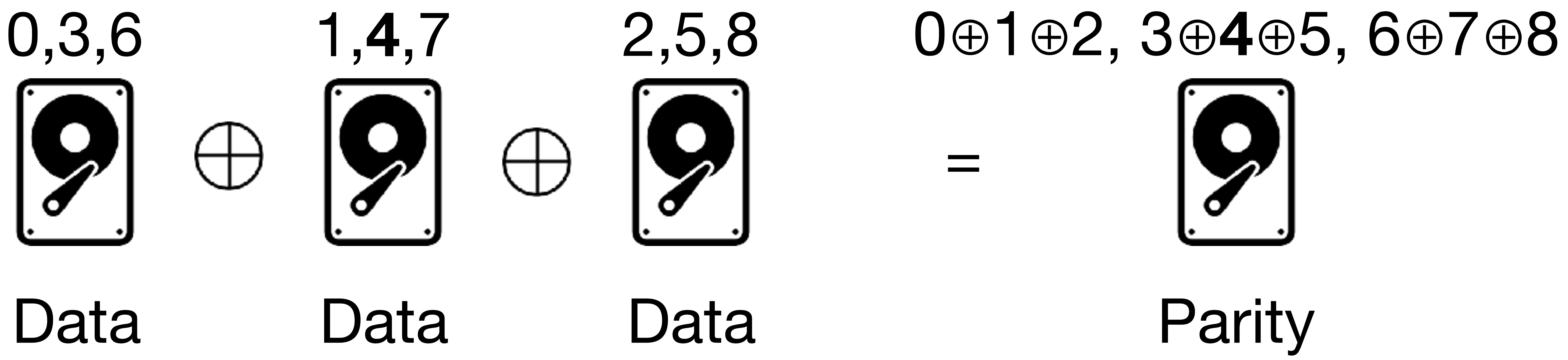
# RAID 4 - Parity Drive

Given N drives, stripe on N-1 drives and use 1 drive for parity



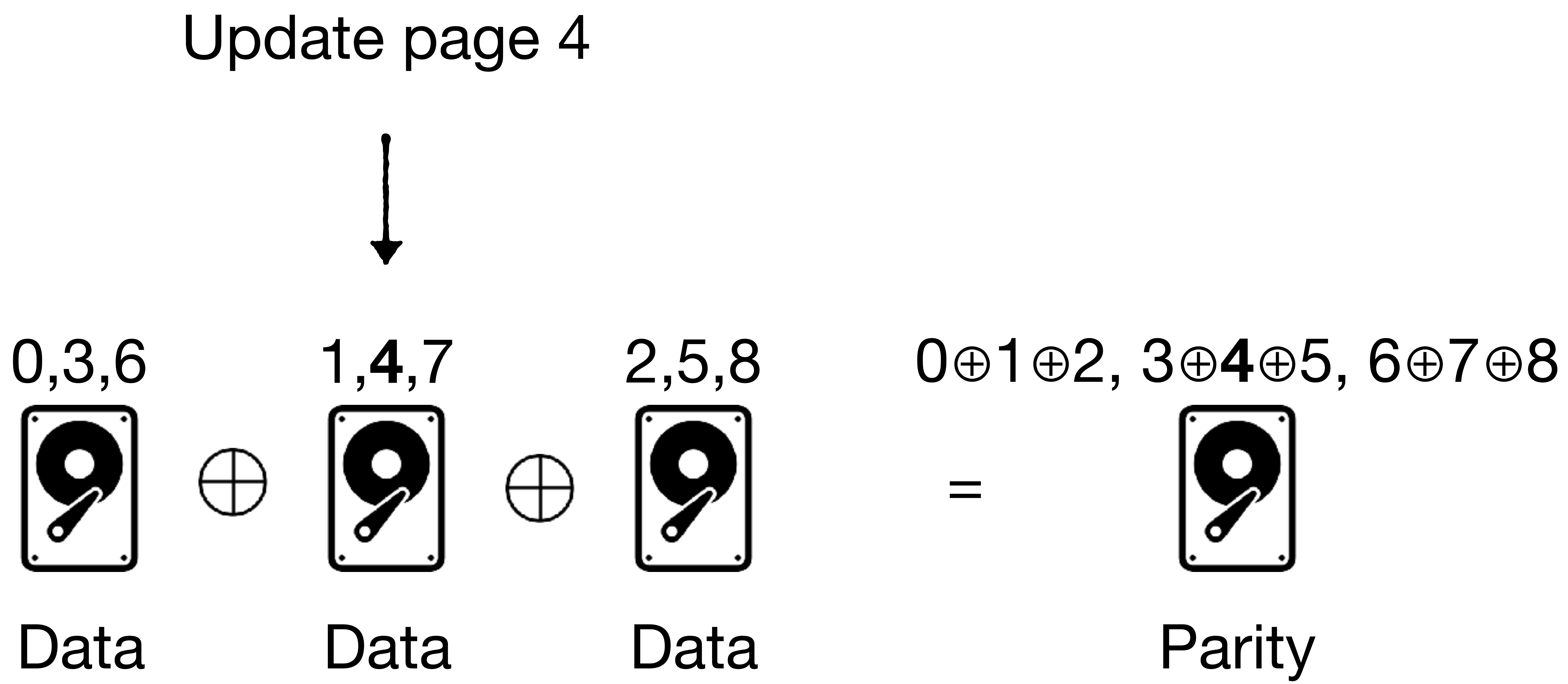
# RAID 4 - Parity Drive

**It’s easy to update a whole stripe.**



# RAID 4 - Parity Drive

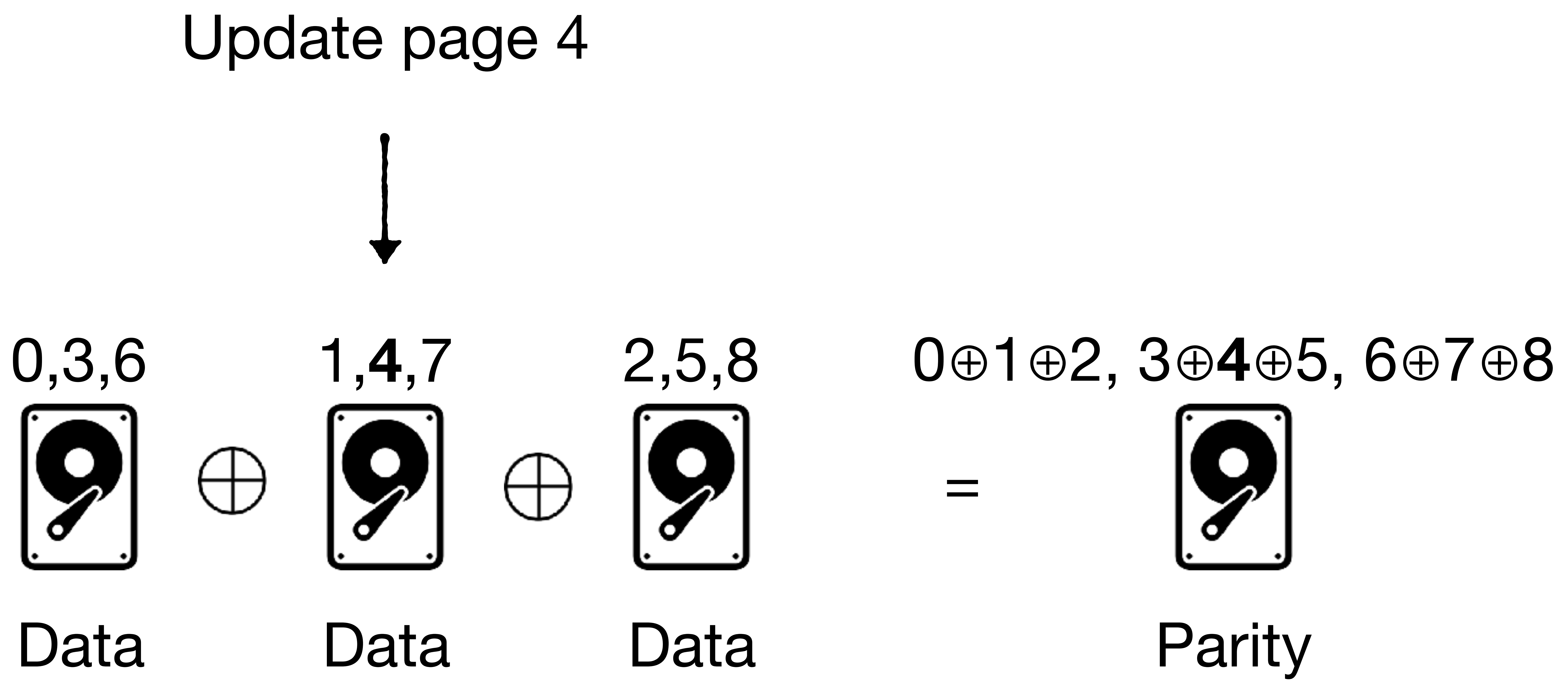
It’s easy to update a whole stripe. **How about updating one page?**



## RAID 4 - Parity Drive

It’s easy to update a whole stripe. How about updating one page?

**Option 1: read all pages at matching offset and regenerate parity:**

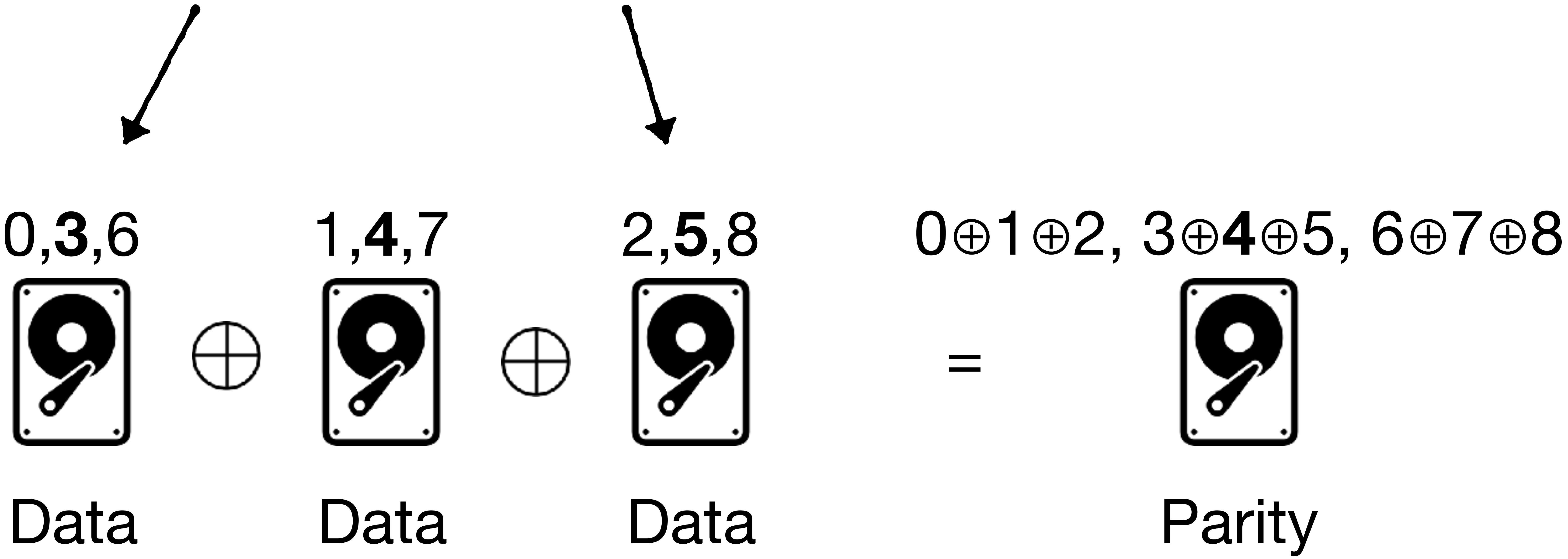


## RAID 4 - Parity Drive

It’s easy to update a whole stripe. How about updating one page?

**Option 1: read all pages at matching offset and regenerate parity:**

$$\mathbf{3 \oplus 4_{new} \oplus 5 = (3\oplus4\oplus5)_{new}}$$

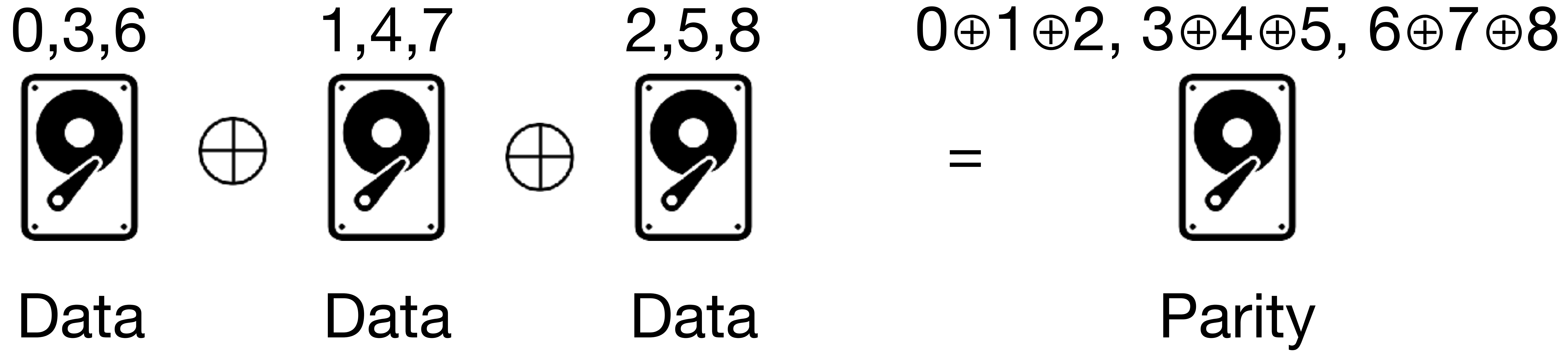


# RAID 4 - Parity Drive

It’s easy to update a whole stripe. How about updating one page?

Option 1: read all pages at matching offset and regenerate parity:

**Cost: N-2 reads & 2 writes**



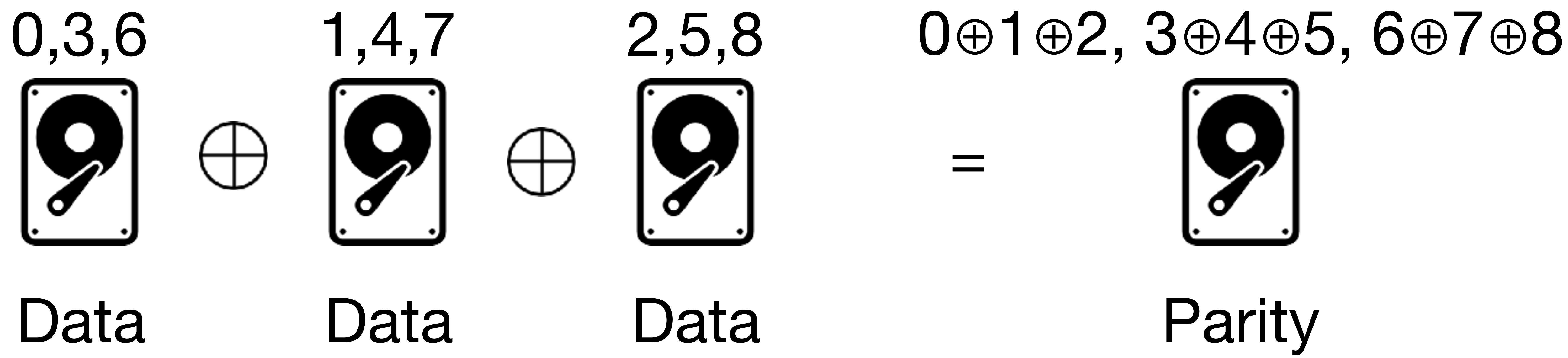
## RAID 4 - Parity Drive

It’s easy to update a whole stripe. How about updating one page?

Option 1: read all pages at matching offset and regenerate parity:

**Cost: N-2 reads & 2 writes**

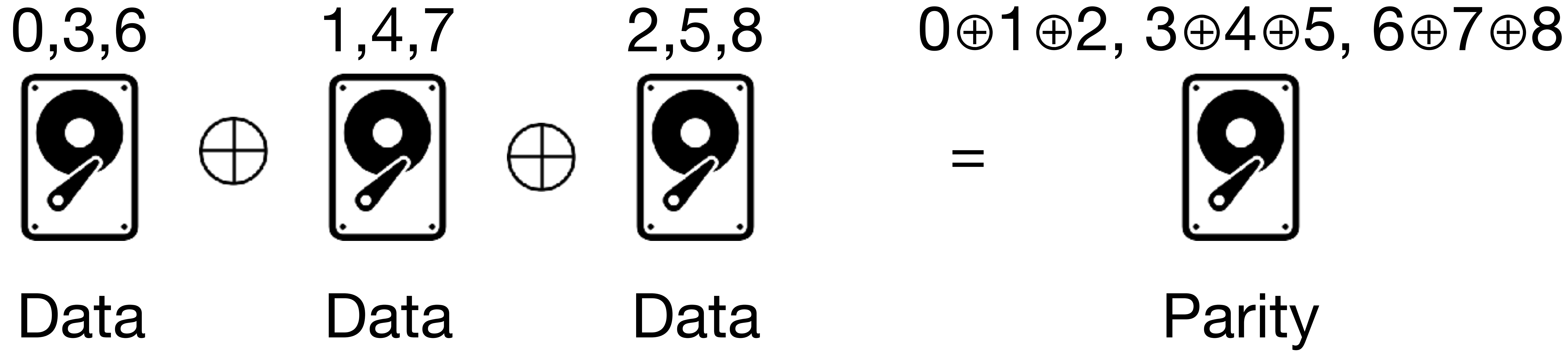
**Can we do better?**



# RAID 4 - Parity Drive

It's easy to update a whole stripe. How about updating one page?

**Option 2: read only the original value and the parity**

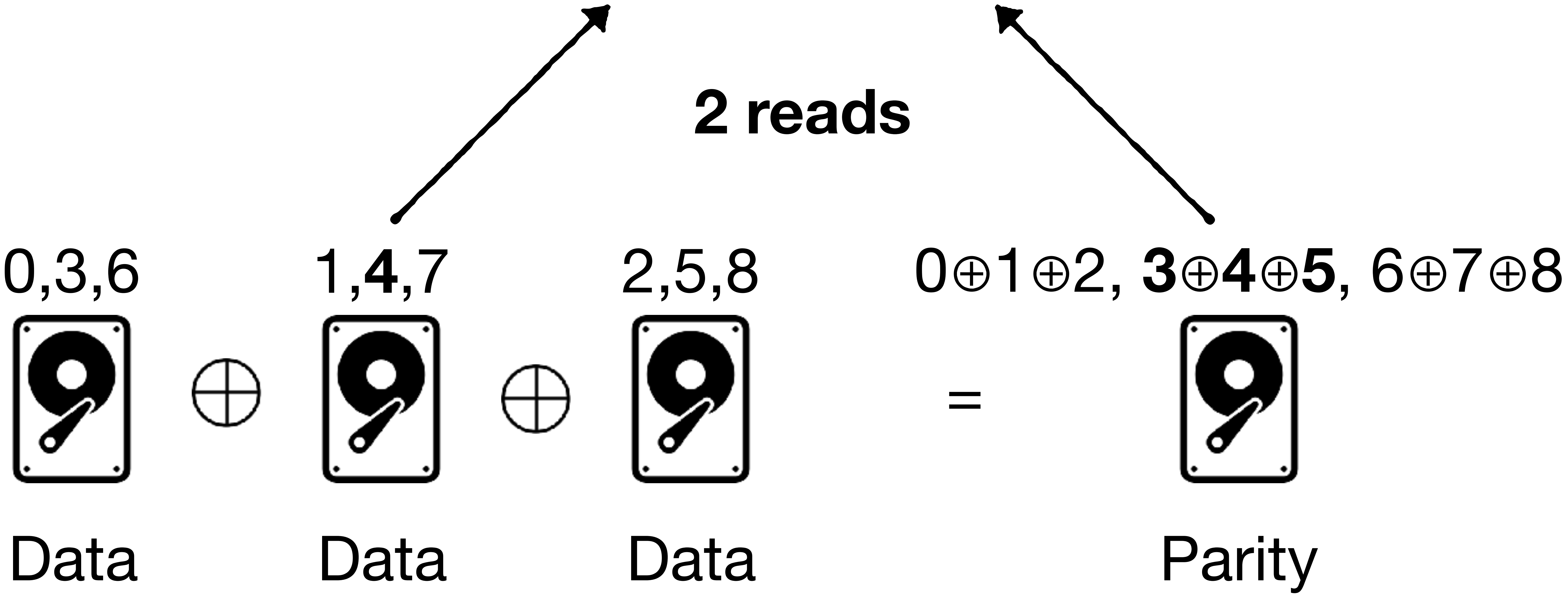


## RAID 4 - Parity Drive

It's easy to update a whole stripe. How about updating one page?

Option 2: read only the original value and the parity

**Update page 4:** $\mathbf{4_{new} \oplus 4_{old} \oplus 3{\oplus}4{\oplus}5_{old} = 3{\oplus}4{\oplus}5_{new}}$



# RAID 4 - Parity Drive

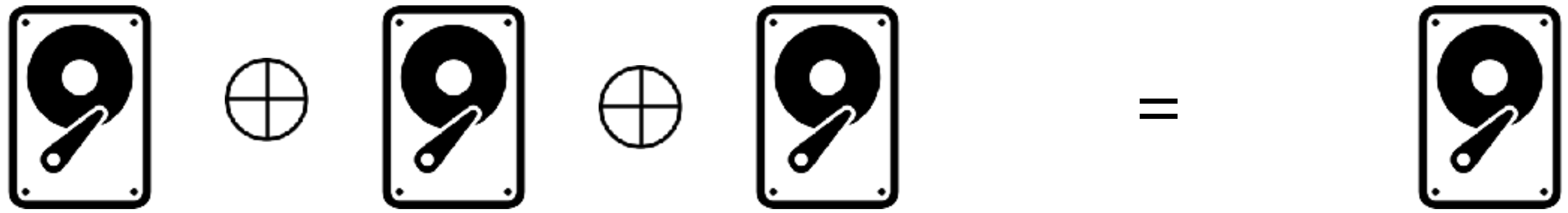
It’s easy to update a whole stripe. How about updating one page?

Option 2: read only the original value and the parity

Update page 4: $4_{new} \oplus 4_{old} \oplus 3\oplus4\oplus5_{old} = 3\oplus4\oplus5_{new}$

**2 writes**

0,3,6 | 1,**4**,7 | 2,5,8 | $0\oplus1\oplus2$, $\mathbf{3\oplus4\oplus5}$, $6\oplus7\oplus8$



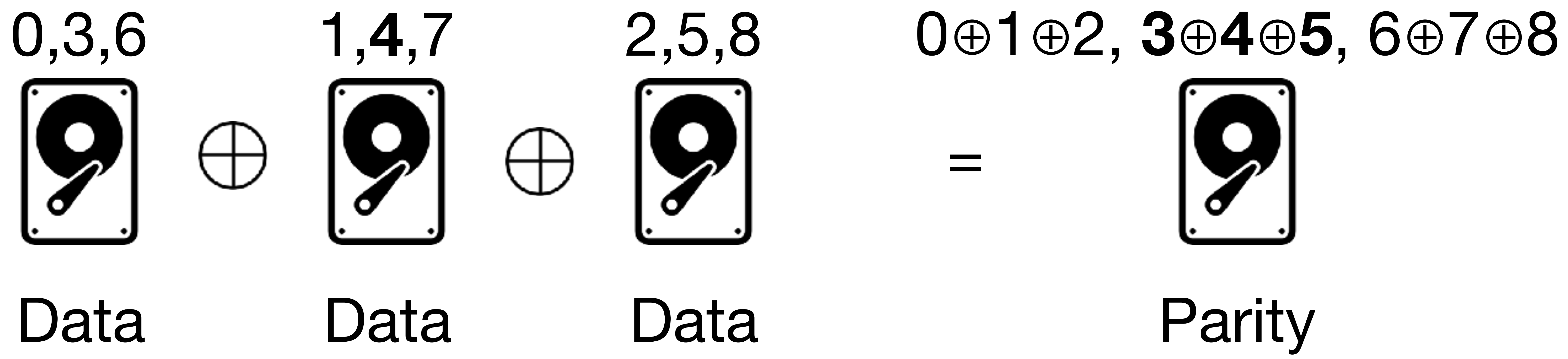
Data | Data | Data | Parity

# RAID 4 - Parity Drive

It’s easy to update a whole stripe. How about updating one page?

Option 2: read only the original value and the parity

**Cost: 2 reads & 2 writes**



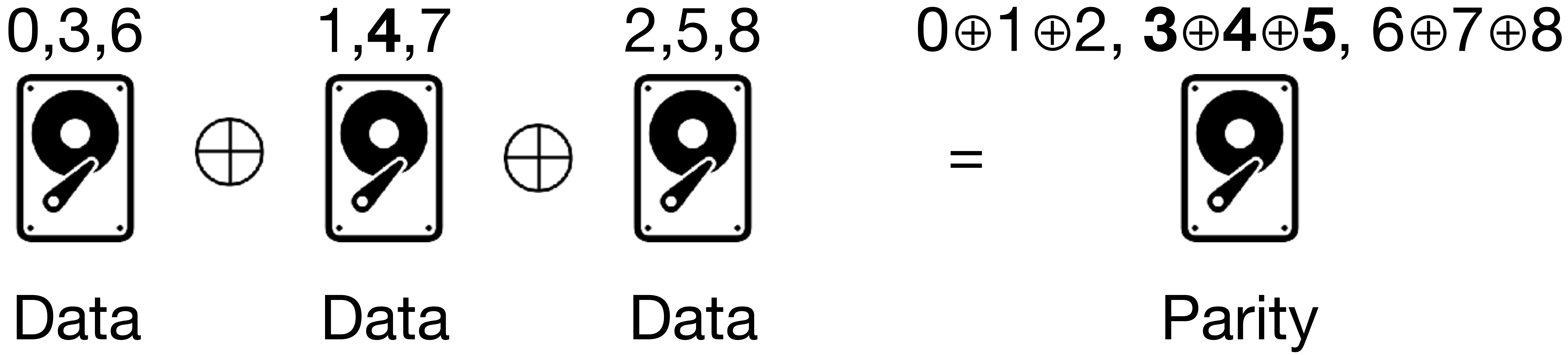
# RAID 4 - Parity Drive

It’s easy to update a whole stripe. How about updating one page?

Option 2: read only the original value and the parity

Cost: 2 reads & 2 writes

**Any problems?**



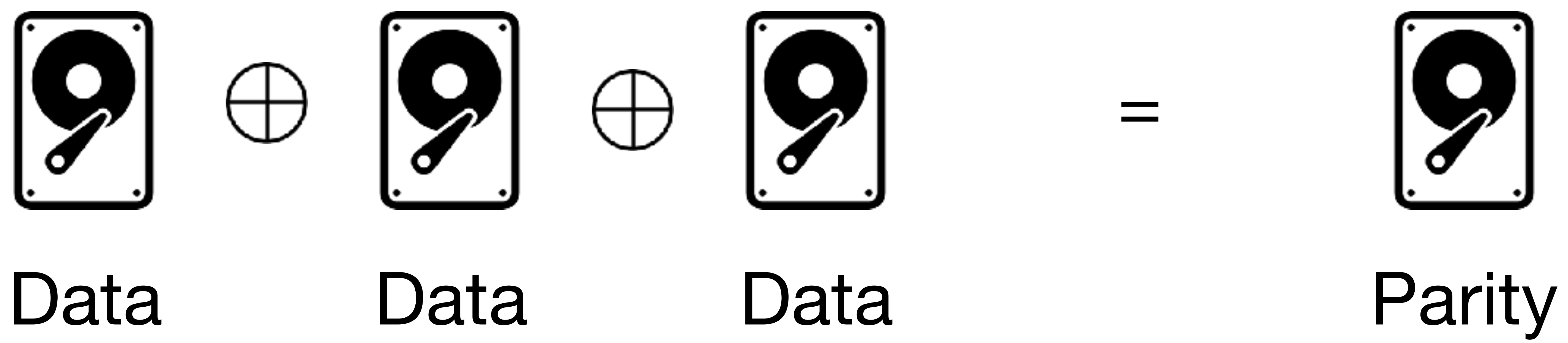
# RAID 4 - Parity Drive

It’s easy to update a whole stripe. How about updating one page?

Option 2: read only the original value and the parity

Cost: 2 reads & 2 writes

**The parity drive becomes a bottleneck for random writes**



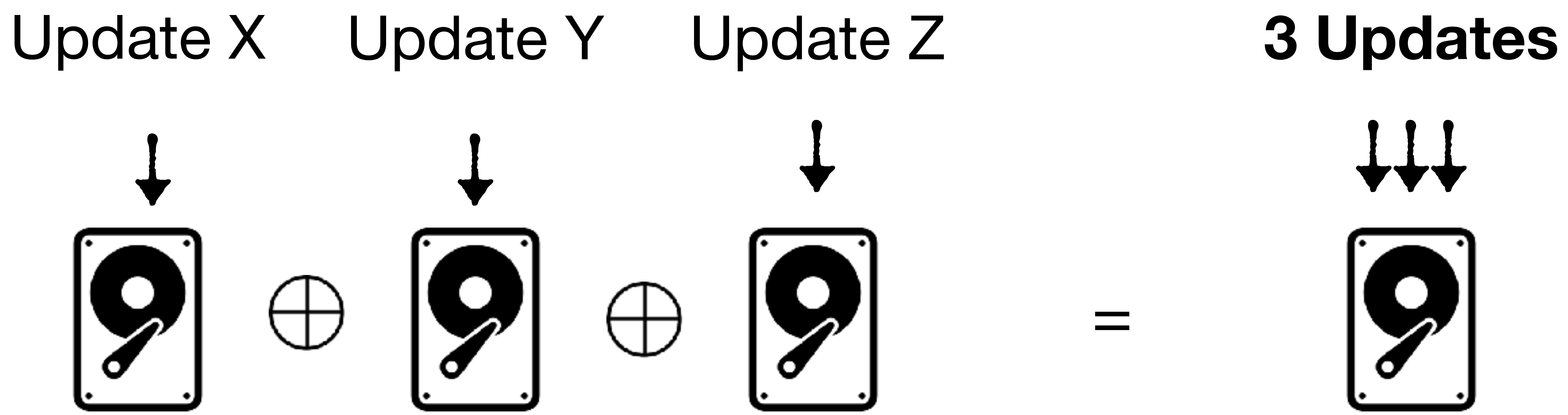
# RAID 4 - Parity Drive

It’s easy to update a whole stripe. How about updating one page?

Option 2: read only the original value and the parity

Cost: 2 reads & 2 writes

**The parity drive becomes a bottleneck for random writes**



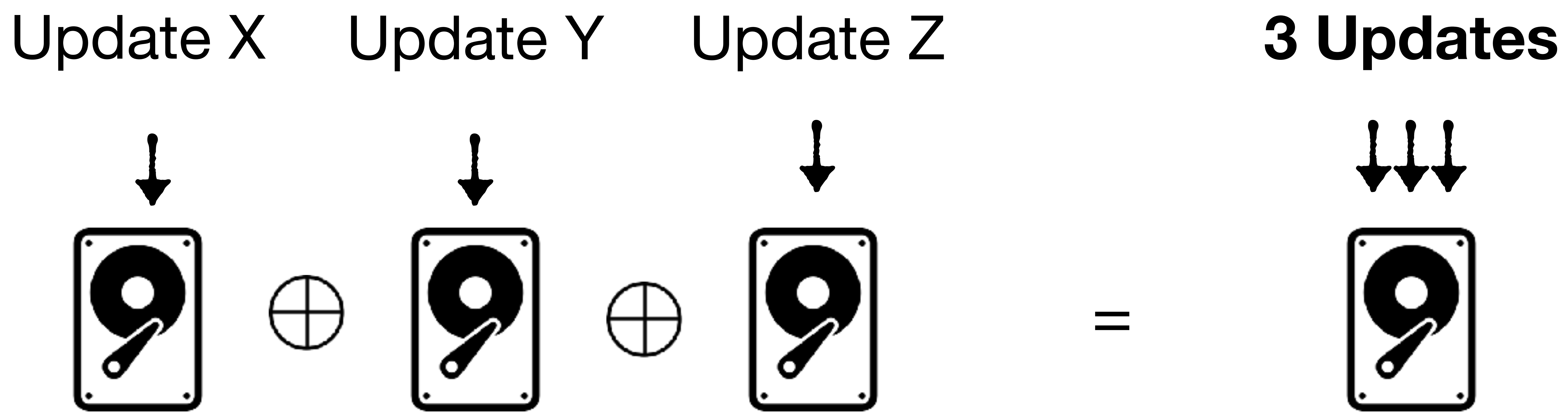
# RAID 4 - Parity Drive

It's easy to update a whole stripe. How about updating one page?

Option 2: read only the original value and the parity

Cost: 2 reads & 2 writes

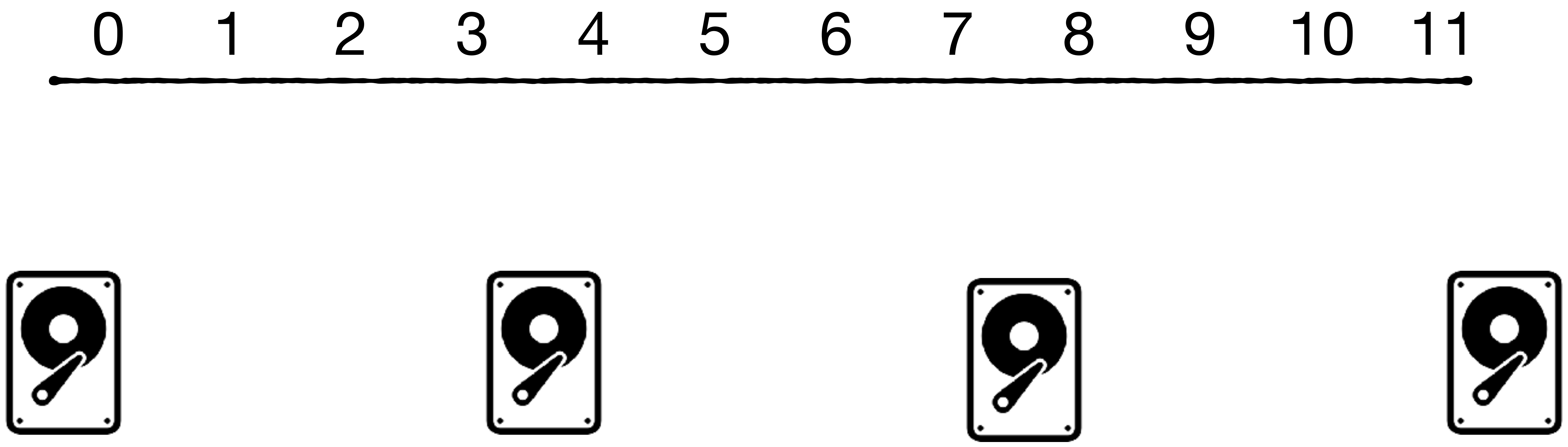
**The parity drive becomes a bottleneck for random writes**




**Can we better distribute this load among the drives?**

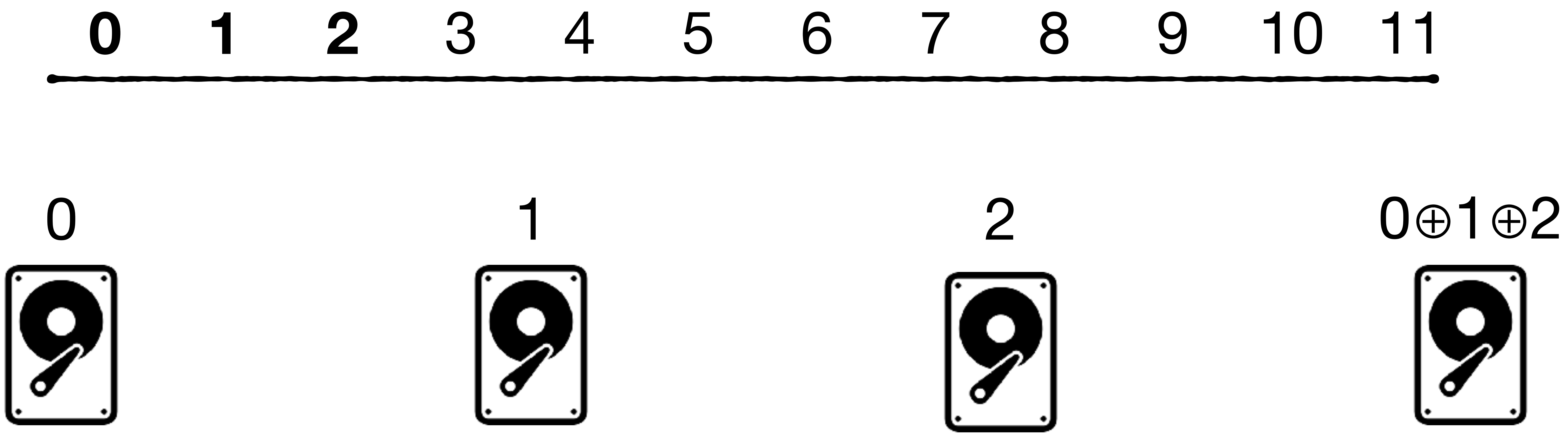
# RAID 5

Striping & distributed parity



# RAID 5

Striping & distributed parity



# RAID 5

Striping & distributed parity

0 1 2 **3** **4** **5** 6 7 8 9 10 11

0,3

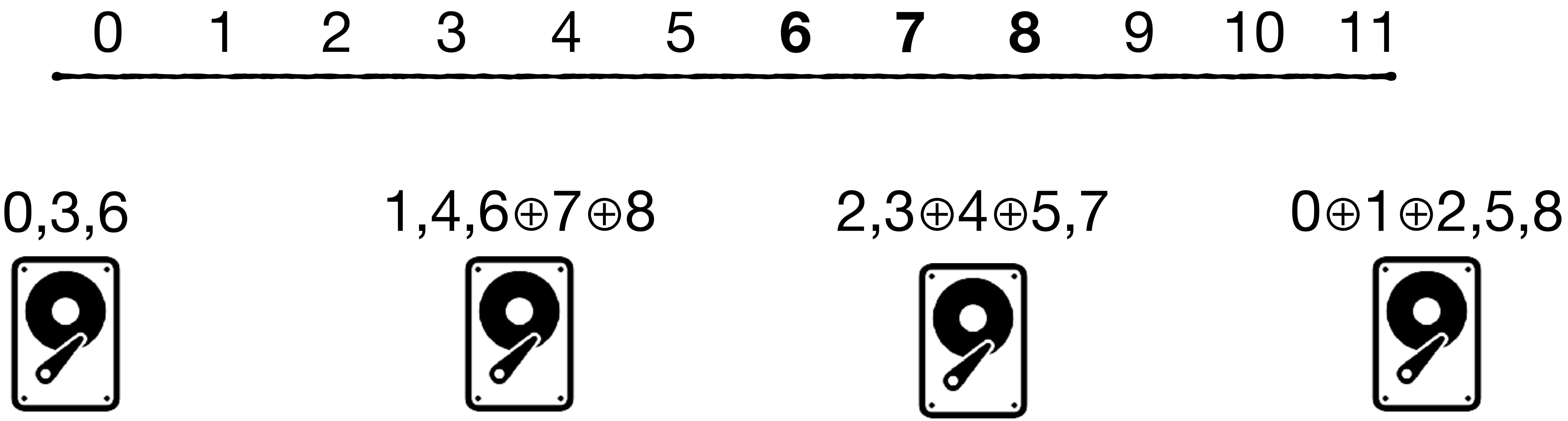
1,4

$2, 3 \oplus 4 \oplus 5$

$0 \oplus 1 \oplus 2, 5$

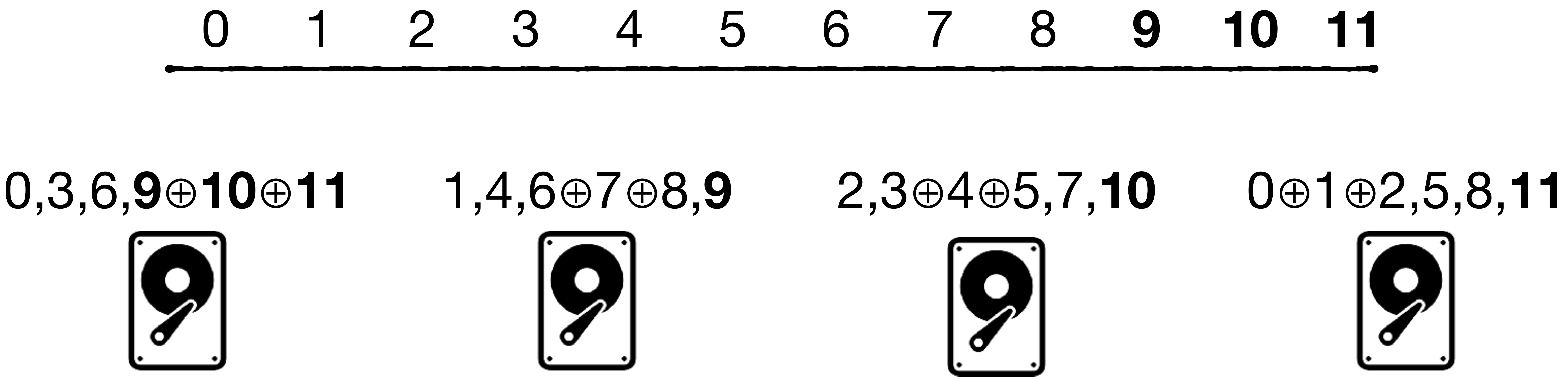
# RAID 5

Striping & distributed parity



**RAID 5**

Striping & distributed parity



# RAID 5 - Summary

**Does not waste most of the storage capacity**

# RAID 5 - Summary

Does not waste most of the storage capacity

**Sequential writes at (N-1)/N of max bandwidth**

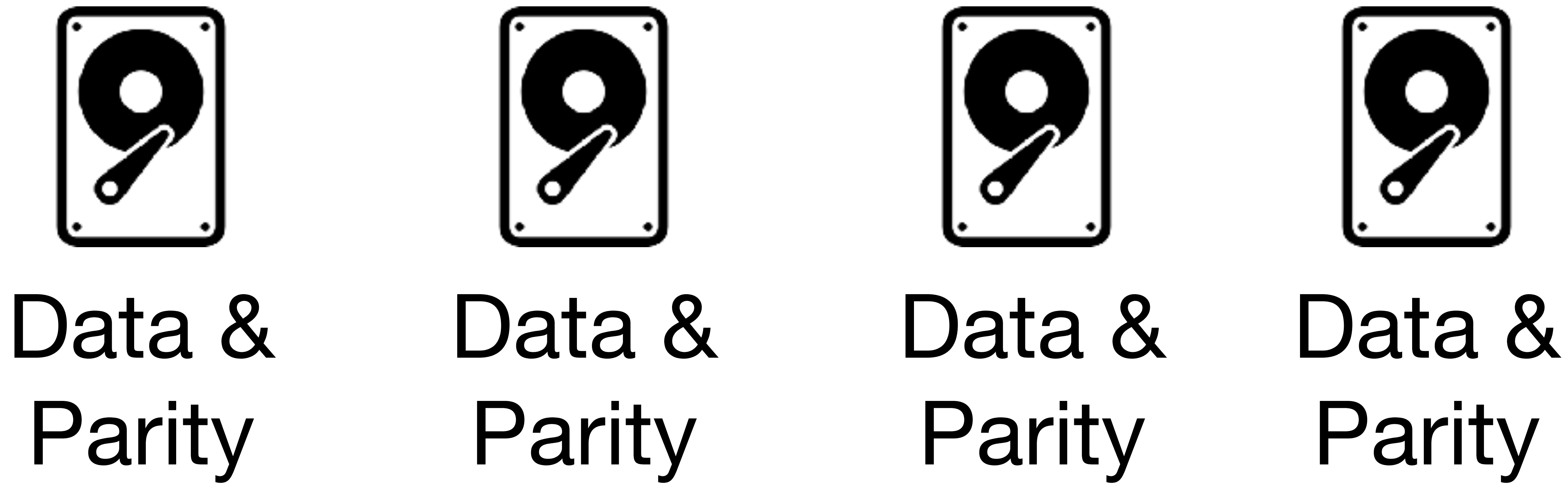
Data & Parity | Data & Parity | Data & Parity | Data & Parity

# RAID 5 - Summary

Does not waste most of the storage capacity

Sequential writes at (N-1)/N of max bandwidth

**Random Reads have less flexibility than with mirroring**



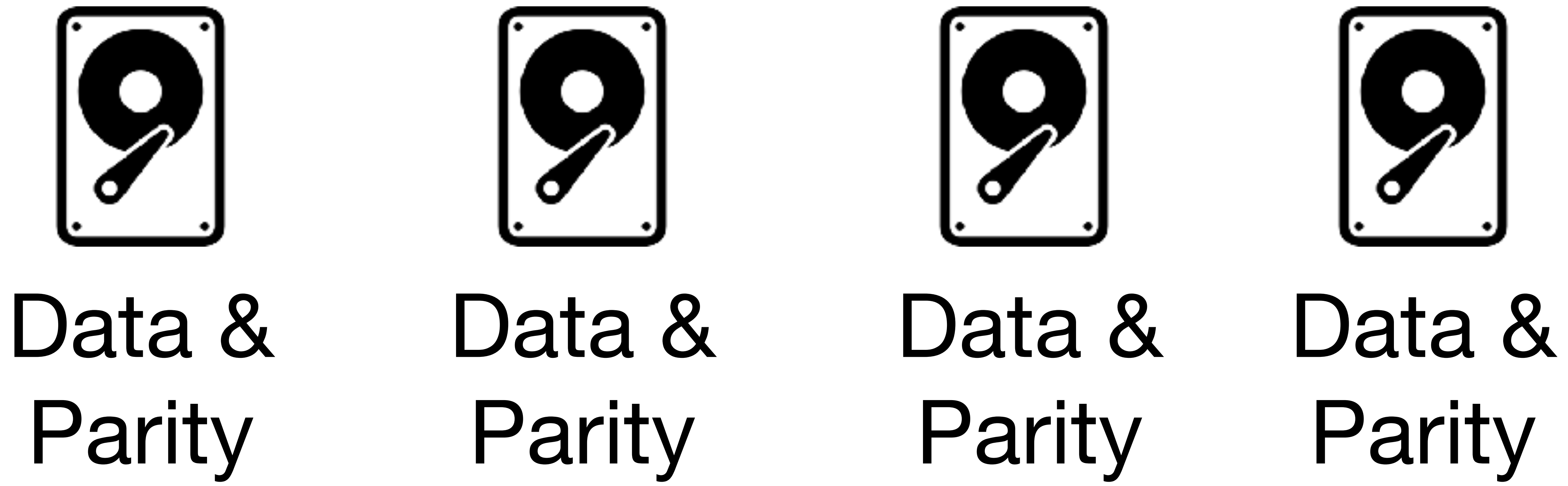
# RAID 5 - Summary

Does not waste most of the storage capacity

Sequential writes at (N-1)/N of max bandwidth

Random Reads have less flexibility than with mirroring

**Fast sequential reads**



# RAID 5 - Summary

Does not waste most of the storage capacity

Sequential writes at (N-1)/N of max bandwidth

Random Reads have less flexibility than with mirroring

Fast sequential reads

**A random write requires 2 reads and 2 writes**

# RAID 5 - Summary

Does not waste most of the storage capacity

Sequential writes at (N-1)/N of max bandwidth

Random Reads have less flexibility than with mirroring

Fast sequential reads

A random write requires 2 reads and 2 writes

**Random writes load is evenly distributed on all drives**

Data & Parity | Data & Parity | Data & Parity | Data & Parity

# RAID 5 - Summary

**What if two or more drives fail at the same time?**

Data & Parity | Data & Parity | Data & Parity | Data & Parity

# RAID 5 - Summary

What if two or more drives fail at the same time?

**Simultaneous drive failure becomes more probable with more drives**

# RAID 5 - Summary

What if two or more drives fail at the same time?

Simultaneous drive failure becomes more probable with more drives

**RAID 5 and RAID 4 only protect against one drive failing**

# RAID 5 - Summary

What if two or more drives fail at the same time?

Simultaneous drive failure becomes more probable with more drives

RAID 5 and RAID 4 only protect against one drive failing

**RAID 6 can handle two drives failing simultaneously**

**Next week:  buffer pools & row-stores**