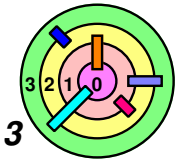
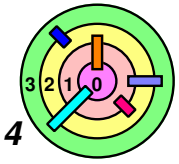
# 6.4 Multiple Disks

- RAID
  - what can be done if you lose an entire disk?



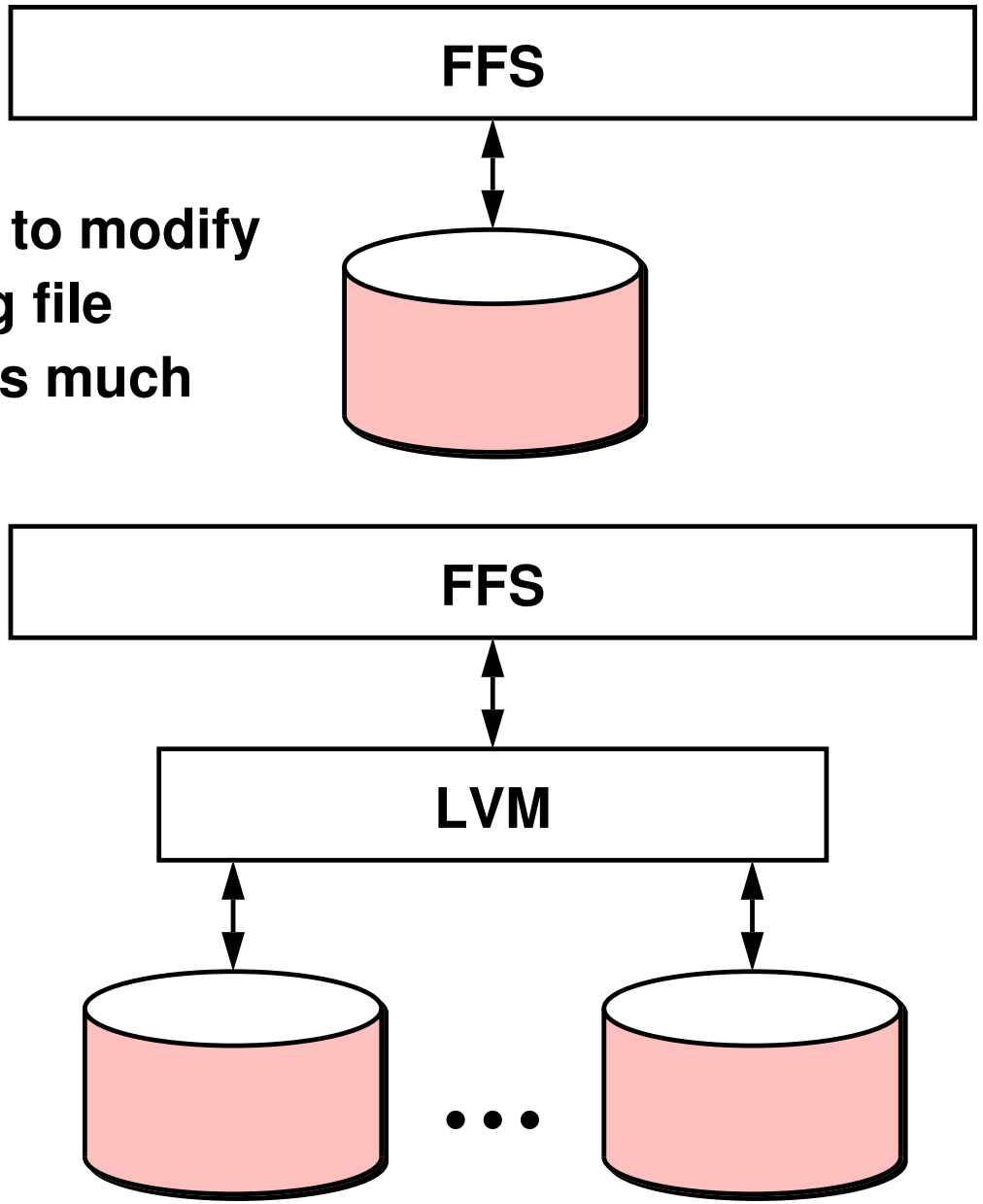
# Benefits of Multiple Disks

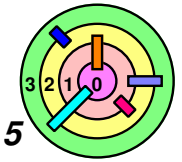
- They hold more data than one disk does
- Data can be stored *redundantly* so that if one disk fails, they can be found on another
  - increase reliability
  - increase availability
- Data can be spread across multiple drives, allowing *parallel* access
  - increase effective access time
    - access time = seek time + rotational latency + data transfer time



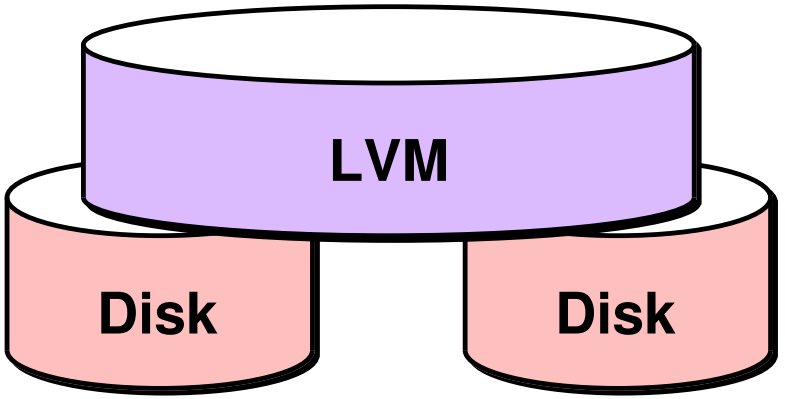
# Logical Volume Manager

- Try not to modify existing file systems much

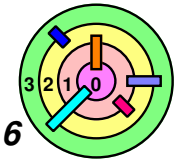


# Logical Volume Manager

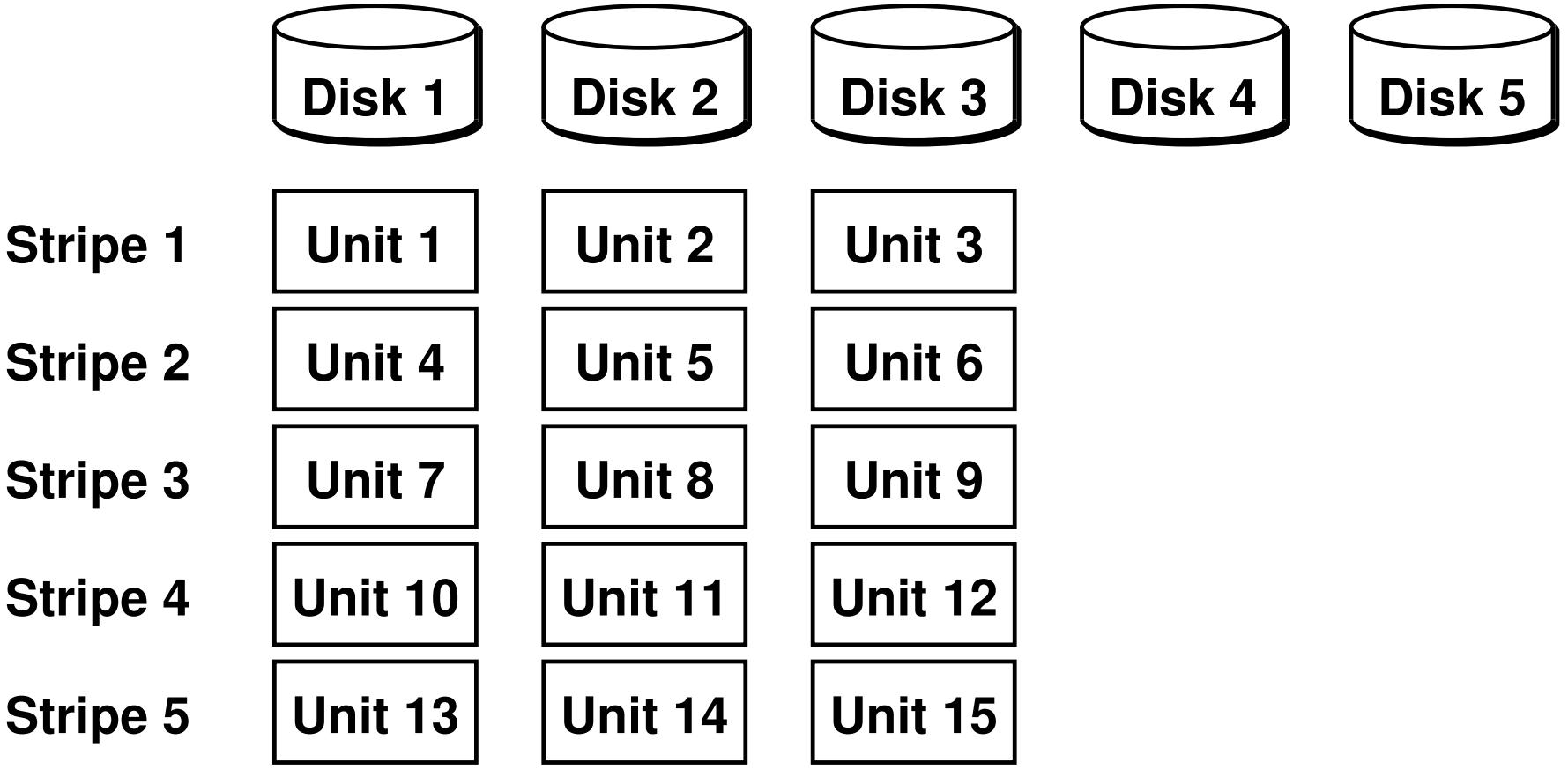



- Mirroring
  - file system writes redundantly to both disks
  - reads from one
- Spanning
  - two (or more) real disks appear to file system as one large disk
    - what are the choices and parameters?
      - pick a disk to store a file
      - break up file into pieces to increase parallelism

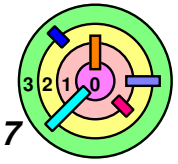


# Striping

Ex: *stripe width* = 3

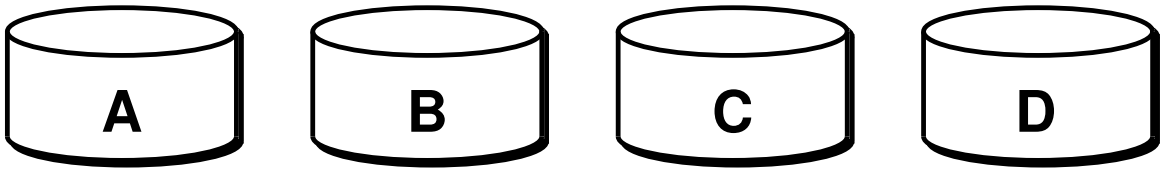



- theoretically, a *striping unit* can be a bit *(i.e., bit-interleaving)*
  - pack these bits into disk blocks and store on disk

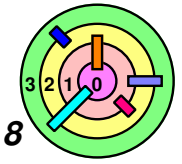


# Parallel Disks

- Advantages
  - increase parallelism
    - can retrieve blocks belonging to multiple files simultaneously if they are on different disks



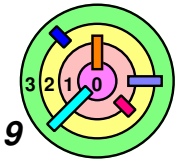

    - reduced access time if a block is spread over multiple disks
      - seek in parallel, wait for rotational latency in parallel
- Disadvantages
  - higher variance
    - average is just part of the story
  - worse reliability (later)
  - heterogenious disks



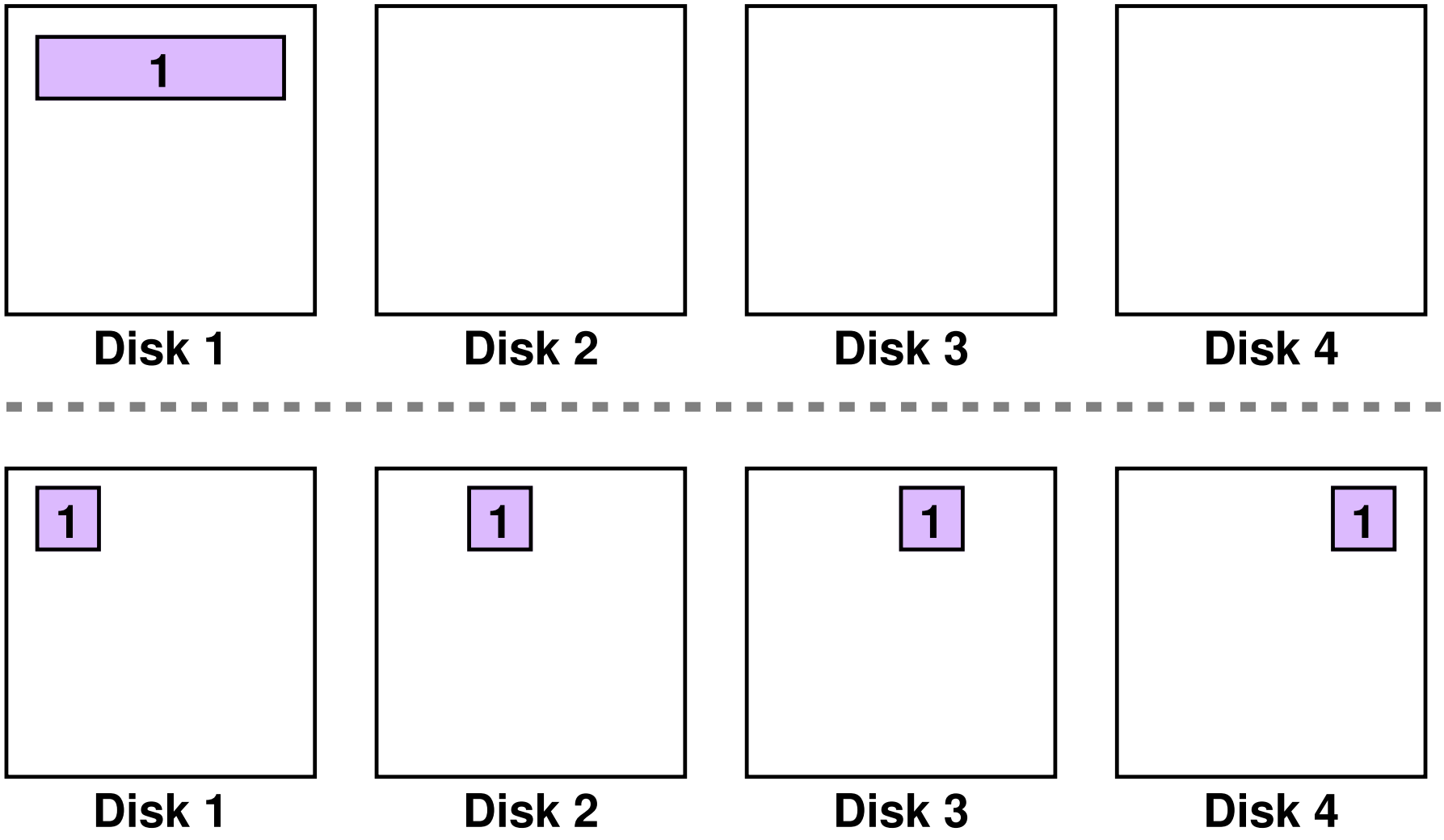
# How To Stripe?

- How to stripe?
  - what's the best *stripe width* (i.e., across how many disks)?
  - how large should be the *striping unit* (i.e., how much to put on one disk before moving to the next disk)?

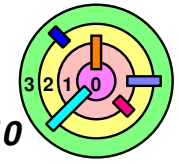
- *Concurrency Factor:* how many requests are available to be executed at once?
  - one request in queue at a time
    - concurrency factor = 1
    - e.g., one single-threaded application placing one request at a time
  - many requests in queue
    - concurrency factor > 1
    - e.g., multiple threads placing file-system requests
  - the larger the concurrency factor, the less important striping is
    - research has shown that, in general, performance is better with *larger striping unit*



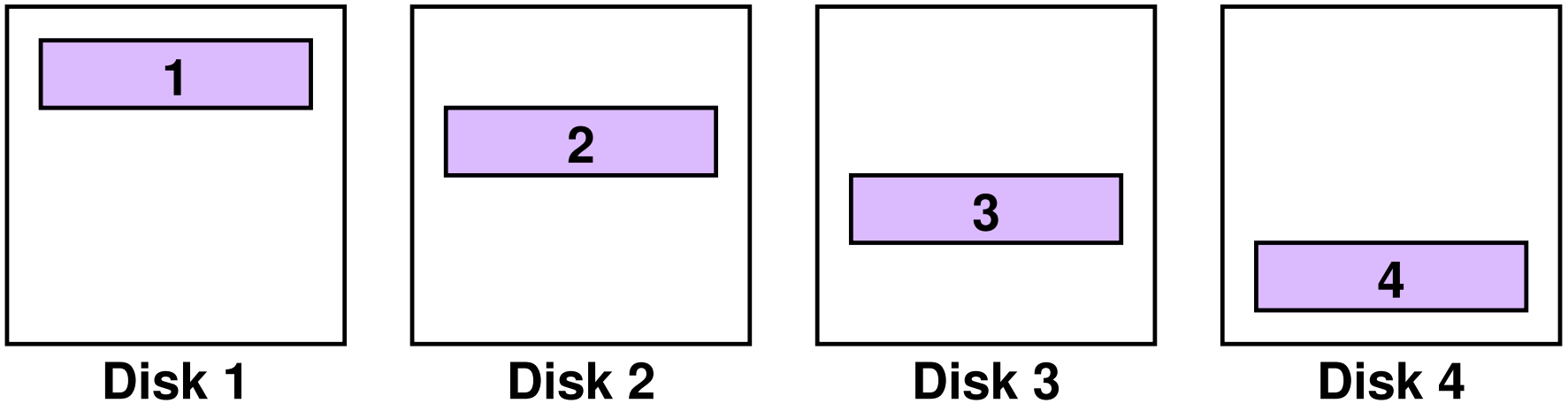
# Striping Unit Size

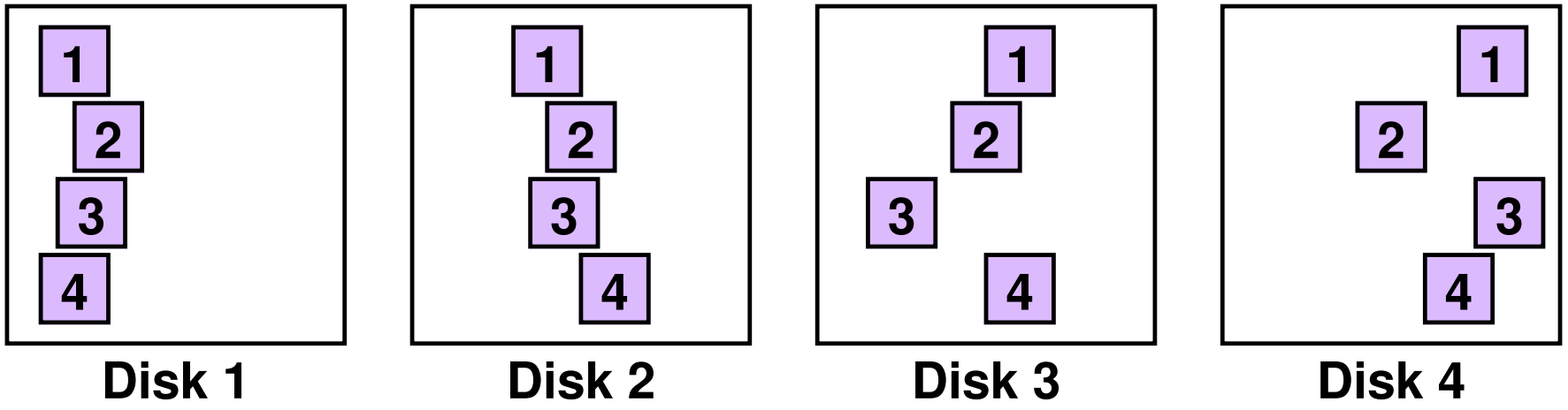



- Bottom solution seems better with concurrency factor of 1
  - data transfer time is 1/4 of the solution on the top

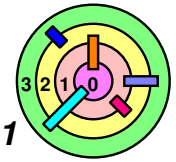


# Striping Unit Size



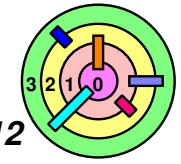




- The above two cases have *same total data transfer time*
  - can *reduce total access time* with a *larger striping unit*



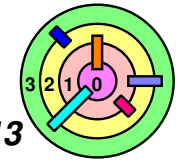
# Striping: The Effective Disk

- Improved effective transfer speed
  - parallelism
- No improvement in seek and rotational delays
  - sometimes worse
- A *system* depending on *N* disks is much more likely to fail than one depending on one disk
  - how to study reliability?
    - failure modes can be complex
    - simple model: assume that failures are *i.i.d.*
      - *i.i.d.: independent* and *identically* distributed
      - clearly not true, but such simplication allows us to gain some insight
  - assuming i.i.d., if probability of one disk failing is *f*
    - in an *N*-disk system, probability of no disk failure is $(1 - f)^N$
    - probability of *N*-disk *system* failing is $(1 - (1 - f)^N)$

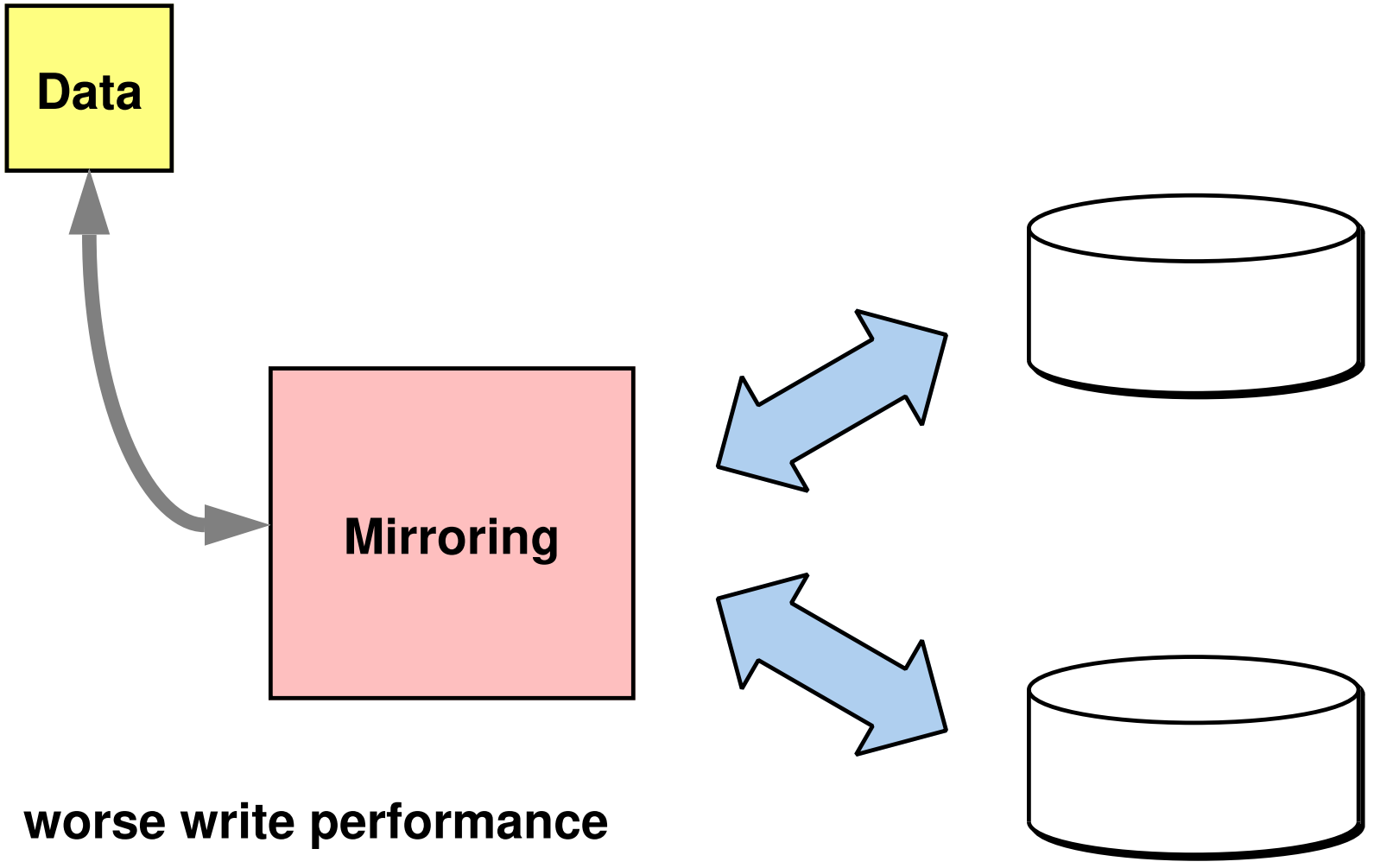


# RAID to the Rescue

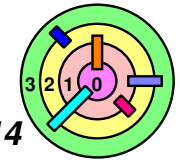
- ***RAID:*** **Redundant Array of Inexpensive Disks**
  - **(as opposed to Single Large Expensive Disk: SLED)**
  - **combine *striping* with *mirroring***
  - **5 different variations originally defined**
    - **RAID level 1 through RAID level 4 developed by IBM**
    - **RAID level 5 developed by UC Berkeley**
      - **RAID level 0: pure striping (numbering extended later)**
    - **RAID level 1: pure mirroring**



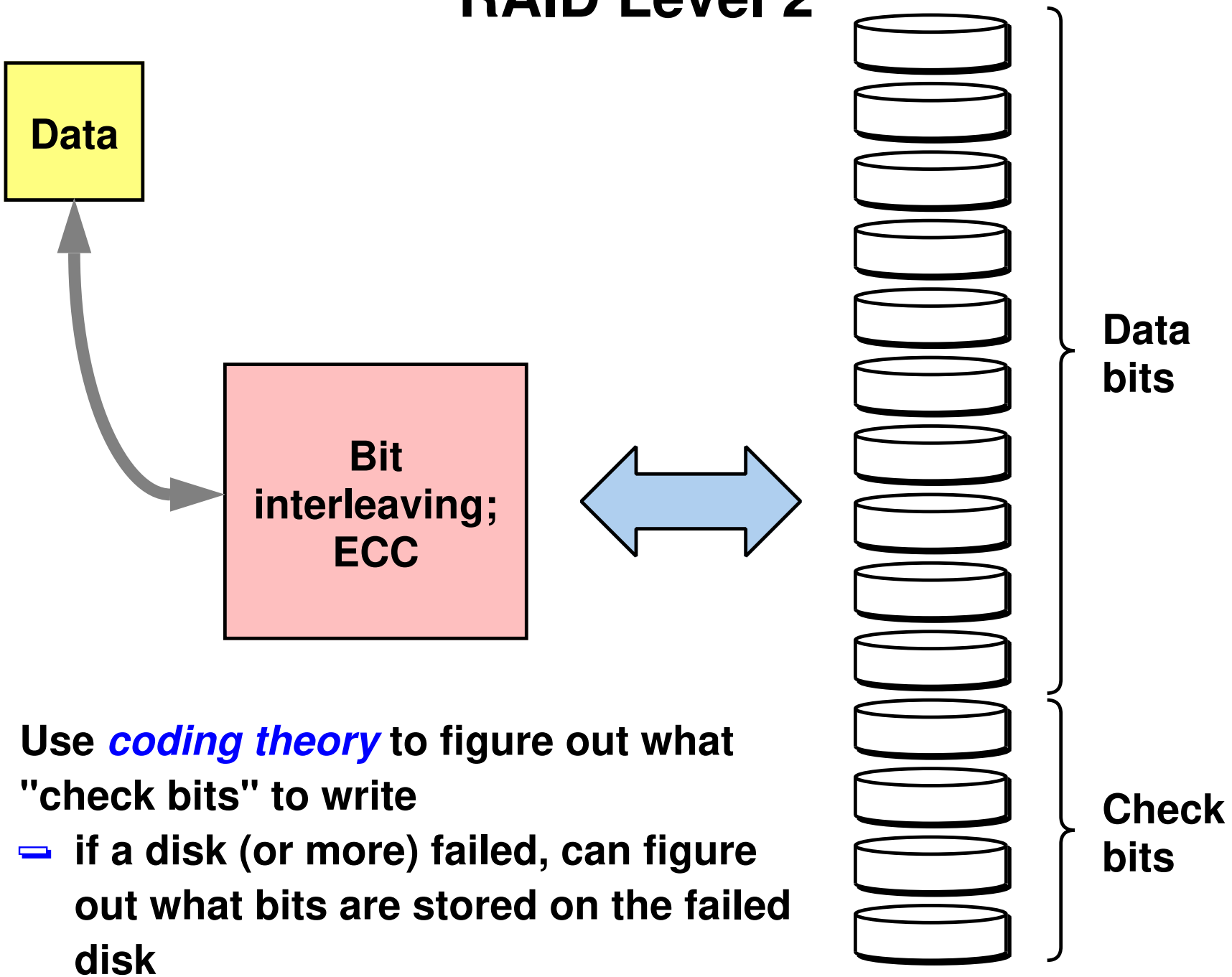
# RAID Level 1



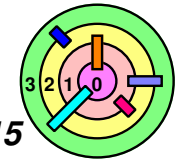

- worse write performance
- can split read traffic
- doing repair reduces performance



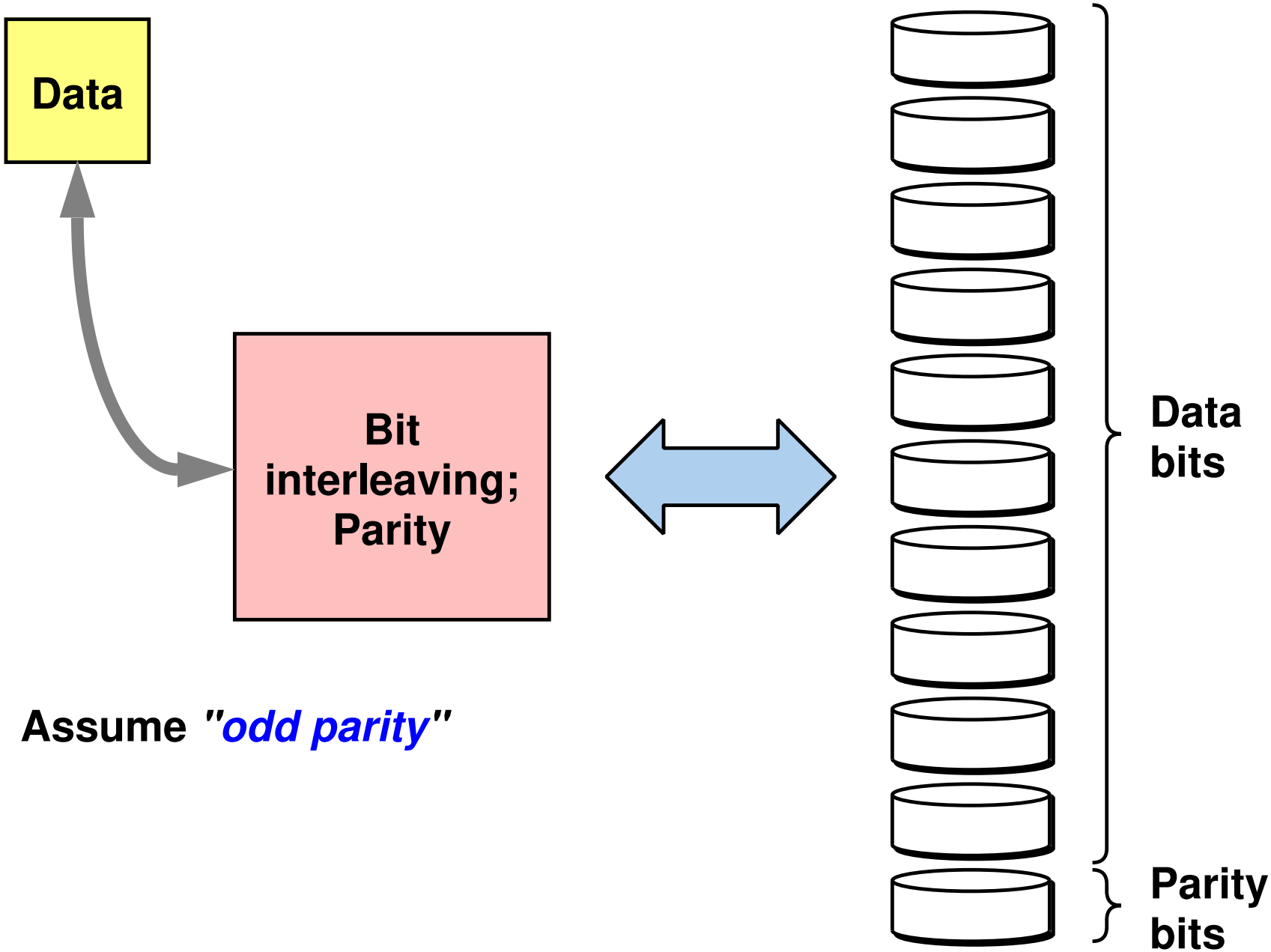
# RAID Level 2




- Use *coding theory* to figure out what "check bits" to write
  - if a disk (or more) failed, can figure out what bits are stored on the failed disk

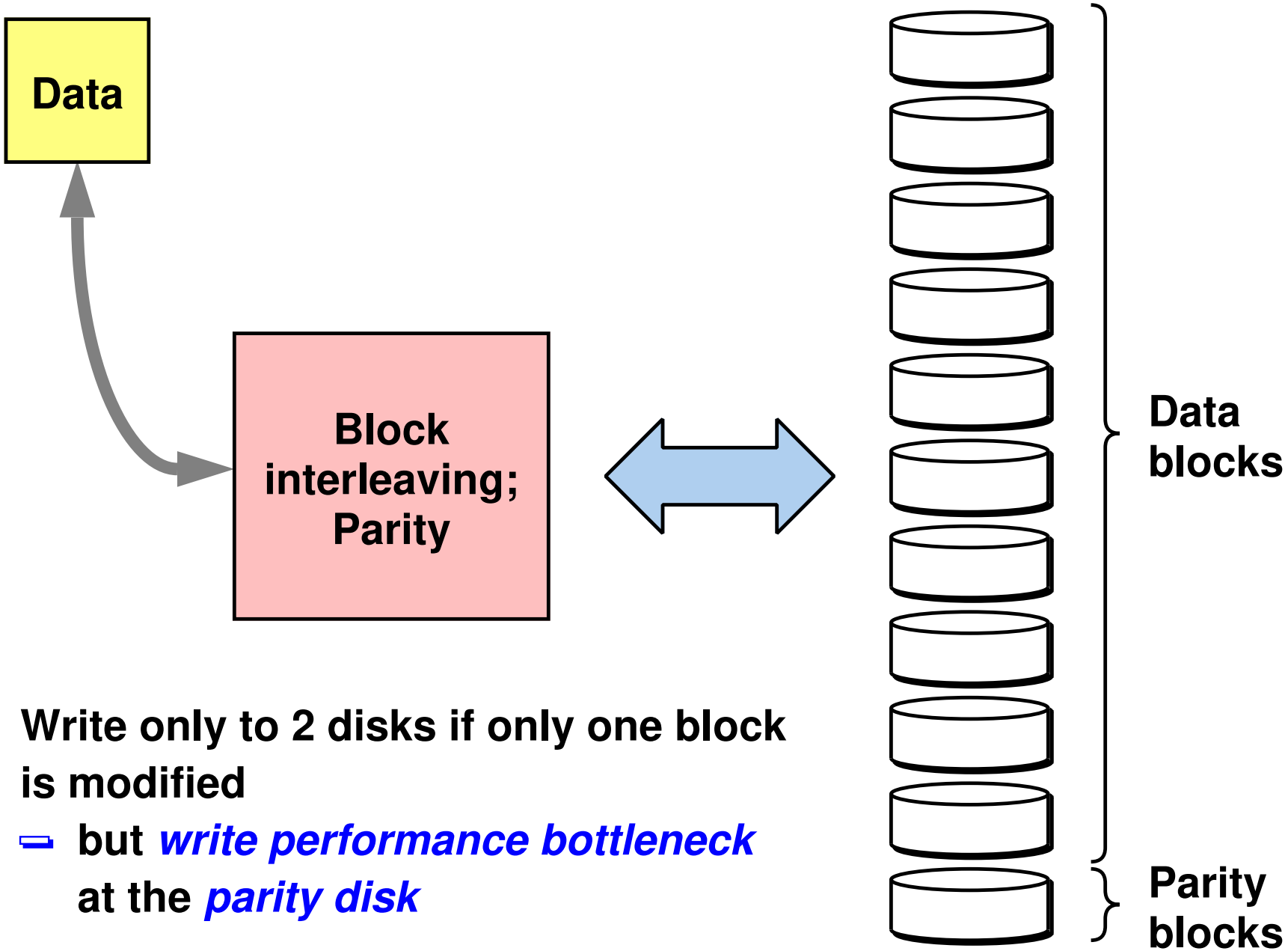


# RAID Level 3

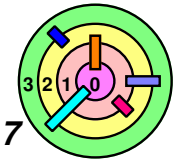



⇨ Assume *"odd parity"*

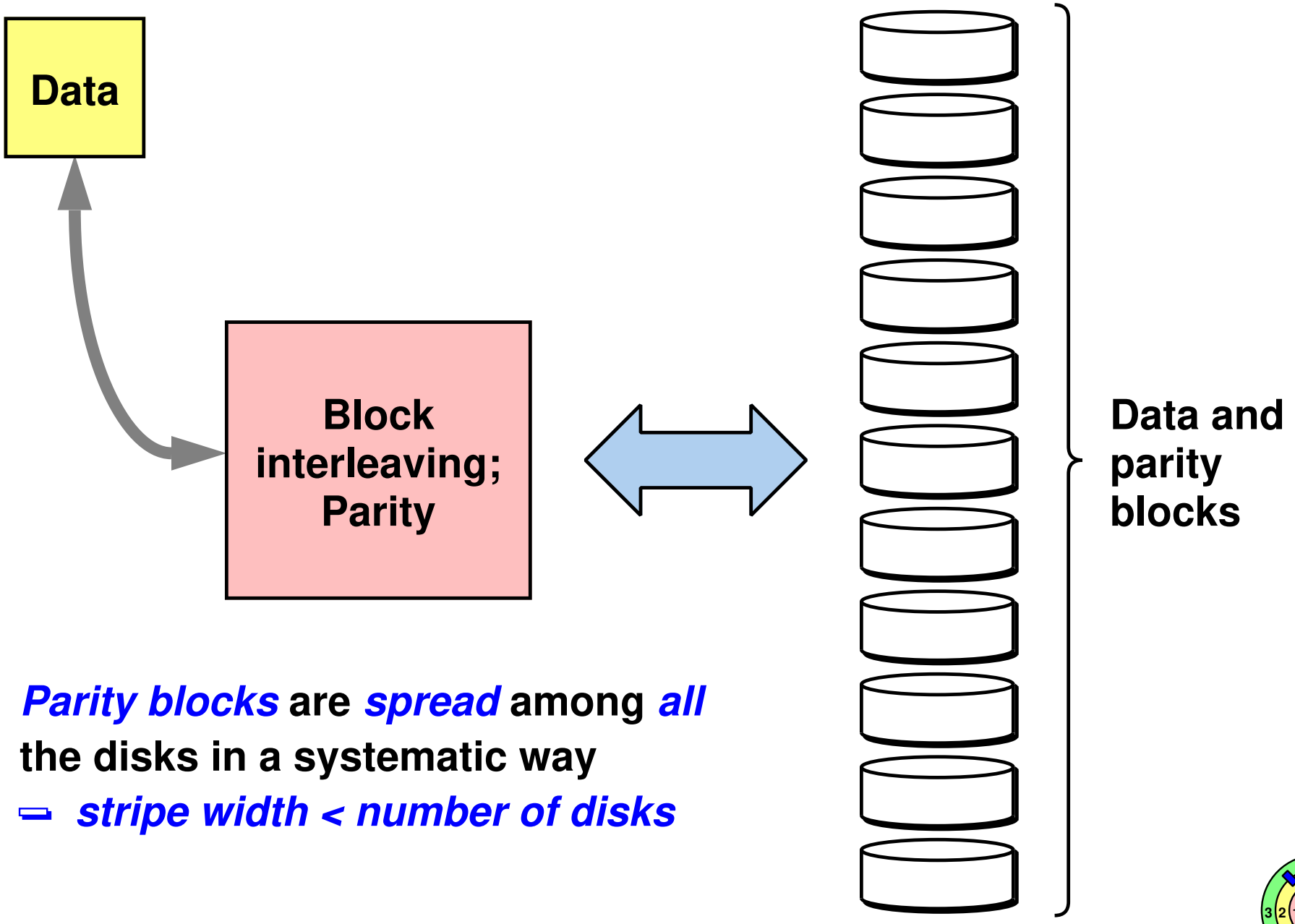
# RAID Level 4




- Write only to 2 disks if only one block is modified
  - but *write performance bottleneck* at the *parity disk*

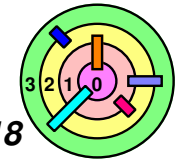


# RAID Level 5



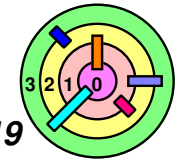

⇨ ***Parity blocks*** **are** ***spread*** **among** ***all*** **the disks in a systematic way**

⇨ ***stripe width < number of disks***



# RAID 4 vs. RAID 5

- Lots of small writes
  - RAID 5 is best
- Mostly large writes
  - multiples of stripes
  - either is fine
- Expansion
  - add an additional disk or two
  - RAID 4: add them and recompute parity
  - RAID 5: add them, recompute parity, shuffle data blocks among all disks to reestablish check-block pattern
- Write performance
  - RAID 4: parity disk have workload multiple of other disks
  - RAID 5: same workload on all disks on the average
- One disk failure
  - RAID 4: parity disk have workload multiple of other disks
  - RAID 5: work load spread out more evenly



# Beyond RAID 5

- RAID 6
  - like RAID 5, but additional parity
  - handles two failures
- Cascaded RAID
  - RAID 1+0 (RAID 10)
    - striping across mirrored drives
  - RAID 0+1
    - two striped sets, mirroring each other