RAID

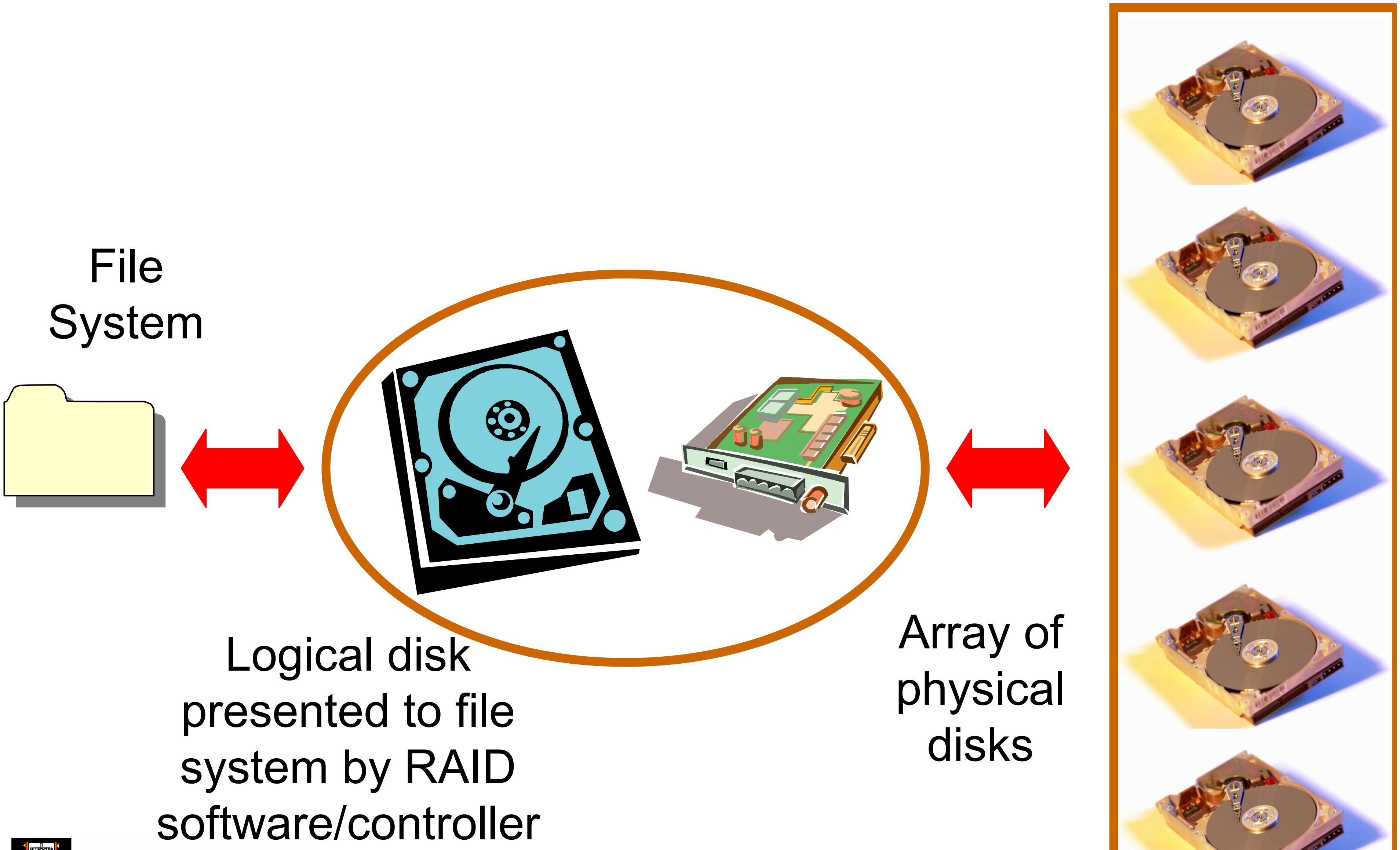
Chapter 5

RAID

- Redundant Array of Inexpensive Disks
  - Industry tends to use “Independent Disks” ☺
- Idea:
  - Use multiple disks to parallelise Disk I/O for better performance
  - Use multiple redundant disks for better availability
- Alternative to a Single Large Expensive Disk (SLED)

RAID Level

- Various configurations of multiple disks are termed a RAID Level
  - Note the Level, does not necessarily imply that one configuration is above or below another.
- We will look at RAID Levels 0 to 5
- All instances of RAID present a single logical disk to the file system.


File System
Logical disk presented to file system by RAID software/controller
Array of physical disks

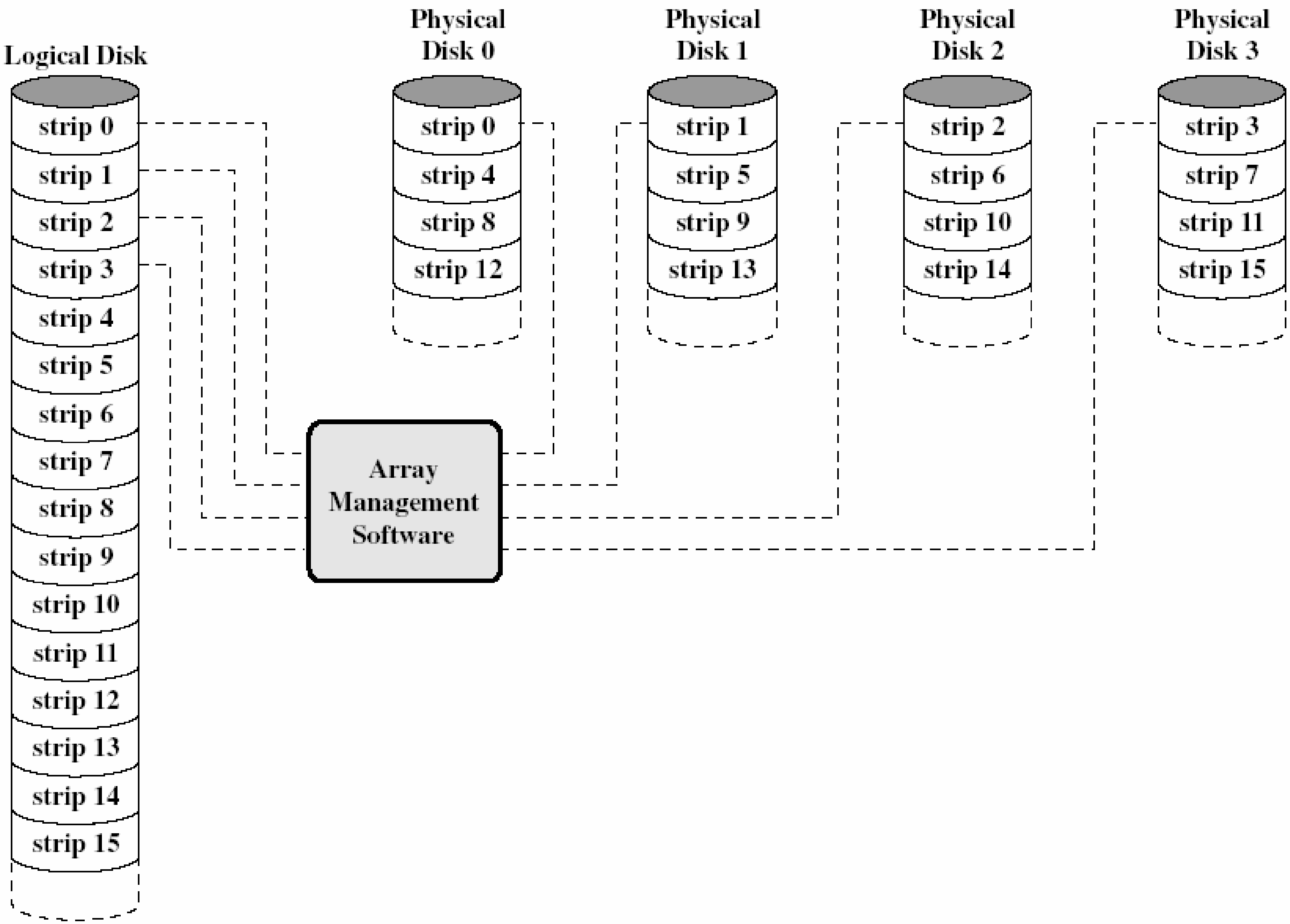


Figure 11.10 Data Mapping for a RAID Level 0 Array [MASS97]

RAID 0

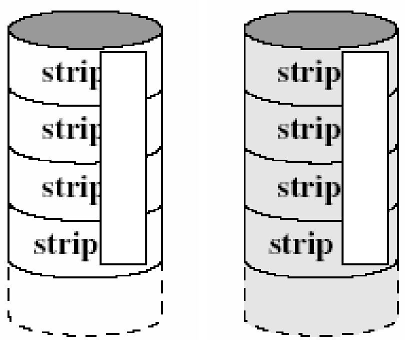
- Logical Disk divided into *strip(e)s*
  - Stripe = a fixed number of sectors
  - First strip written to disk 0
  - Consecutive strips written to different disk in the array in round-robin fashion
- Splits I/O workload across several disks
  - Best with many independent request streams
    - Avoids hotspots on a single disk
- Increases bandwidth available to/from the logical disk.

RAID 0

- Not really true RAID
  - No redundancy
- RAID 0 is less reliable than a SLED
  - Example: Assume MTBF of 10000 hours
  - MTBF of the array is MTBF divided by the number of disks
    - A 4 disk array would have an MTBF of 2500 hours

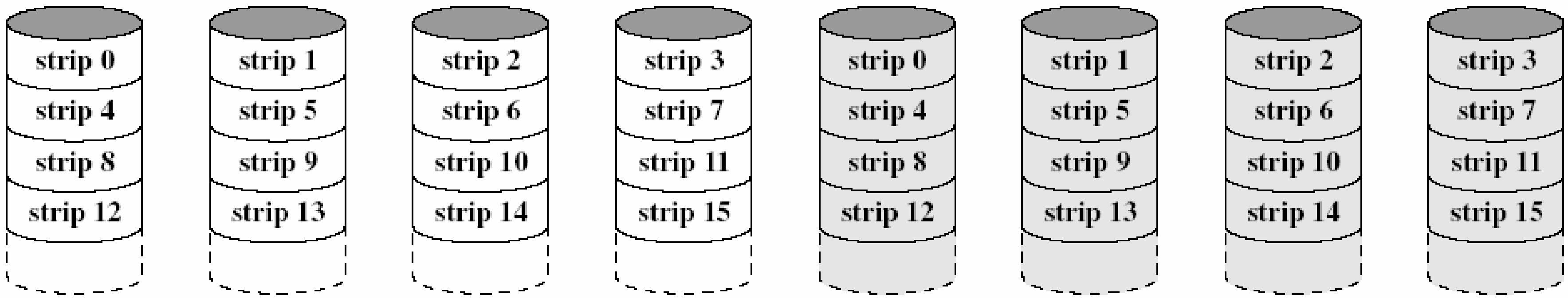
RAID 1

- Each strip is written to two disks
  - Also termed Mirroring (true RAID 1)
- Provides redundancy
  - If disk fails, we can use the copy
- Read performance can double
  - To fetch some blocks, we send half the requests to one disk set, and the other half to the other
- Write performance stays the same
  - A logical write results in two parallel writes to real disks



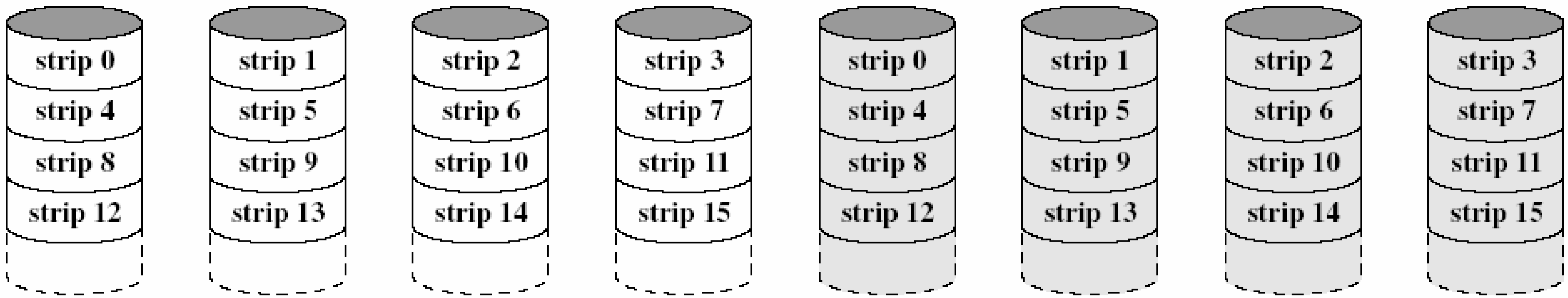
RAID 0+1,1+0, 01, 10

- With striping, sometimes termed RAID 0+1,1+0, 01, 10
- Diagram RAID 0+1
  – Two striped sets (RAID 0)
  – Mirror the two sets (RAID 1)
- Alternative RAID 1+0
  – Many sets mirror pairs (RAID 1)
  – Strip across all sets (RAID 0)
- Splits workload across more disks
- Write performance stays the same
  – A logical write results in two parallel writes to real disks



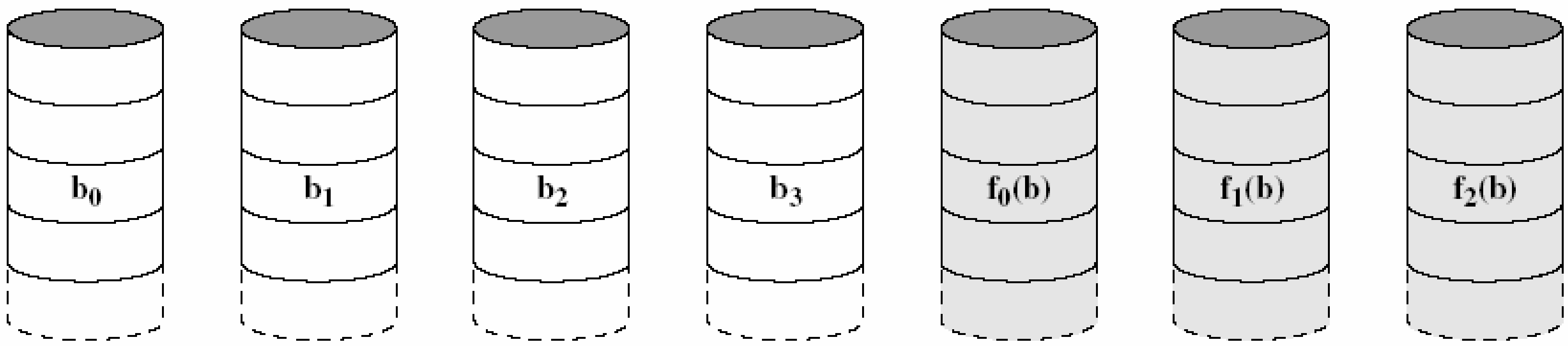
RAID 1

- However
  - RAID 1 requires twice as many disks



RAID 2

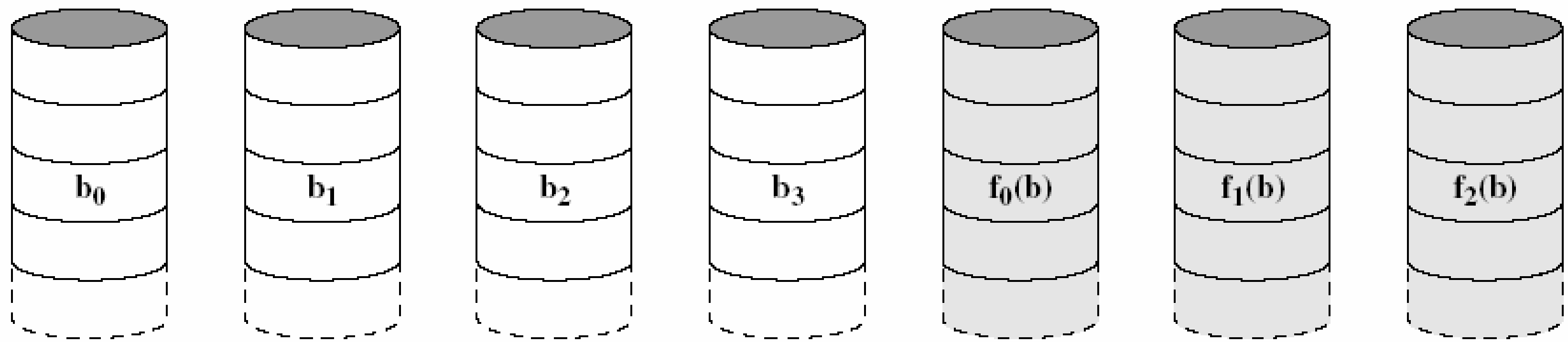
- Example: split data into 4-bit nibbles
- Write each bit to a separate disk
  - Use synchronised spindles to ensure each bit is available at the same time
- Additionally, write 3 Hamming code (ECC) bits to 3 extra disks
  - Hamming code can correct a single bit error



THE UNIVERSITY OF NEW SOUTH WALES

RAID 2

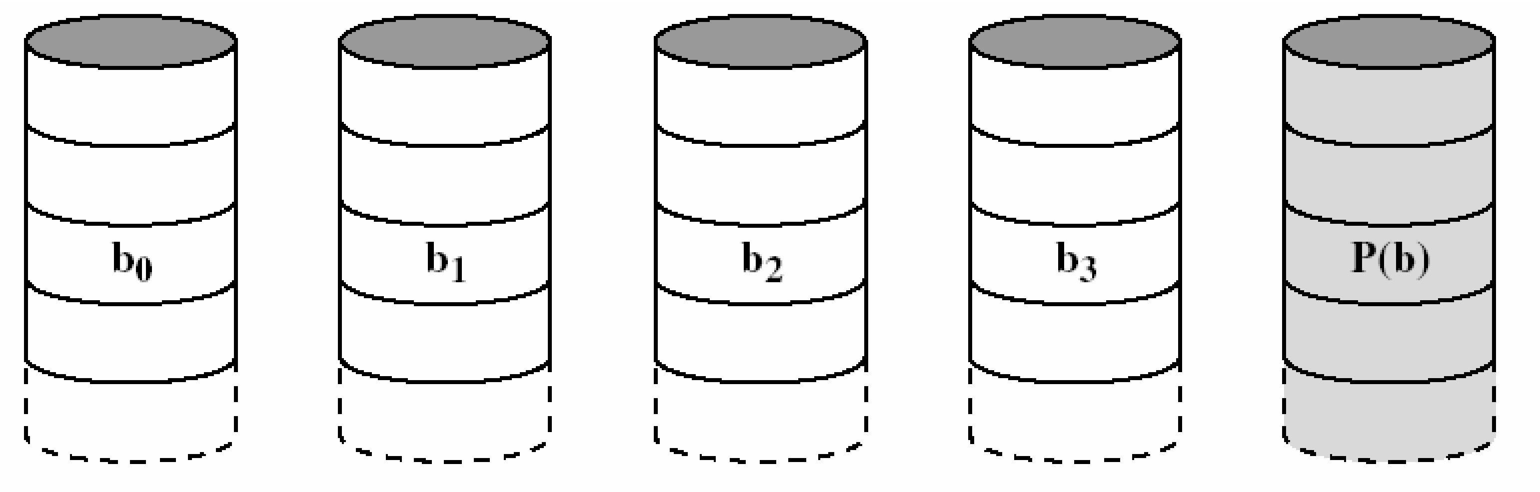
- Makes more sense with more drives
  - 38 drives (32-bit words, with 6-bit ECC)
  - Still 19% storage overhead
- Disadvantage – needs synchronised spindles
- Not used



TH
NE

RAID 3

- Like RAID 2, but instead of ECC, use a single parity bit.
- Can only detect a single error, not correct it
  - Unless we know which bit is wrong



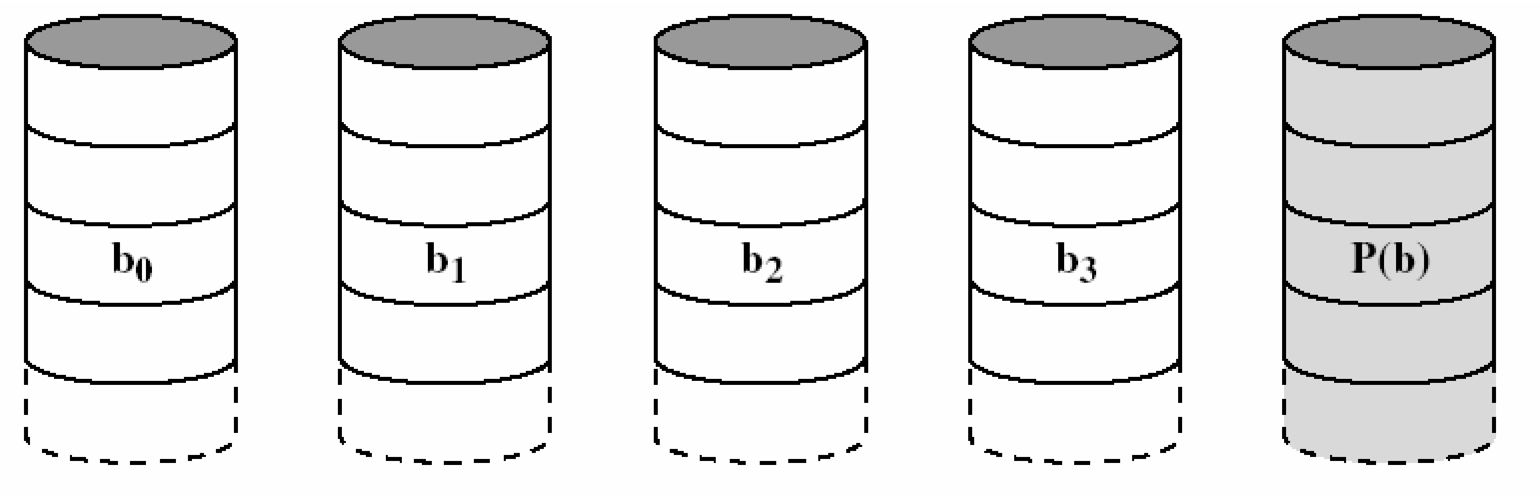
Quick Look At Parity

Disk 1	Disk 2	Disk 3	Disk 4	Parity
1	0	1	0	0

What is the lost bit?

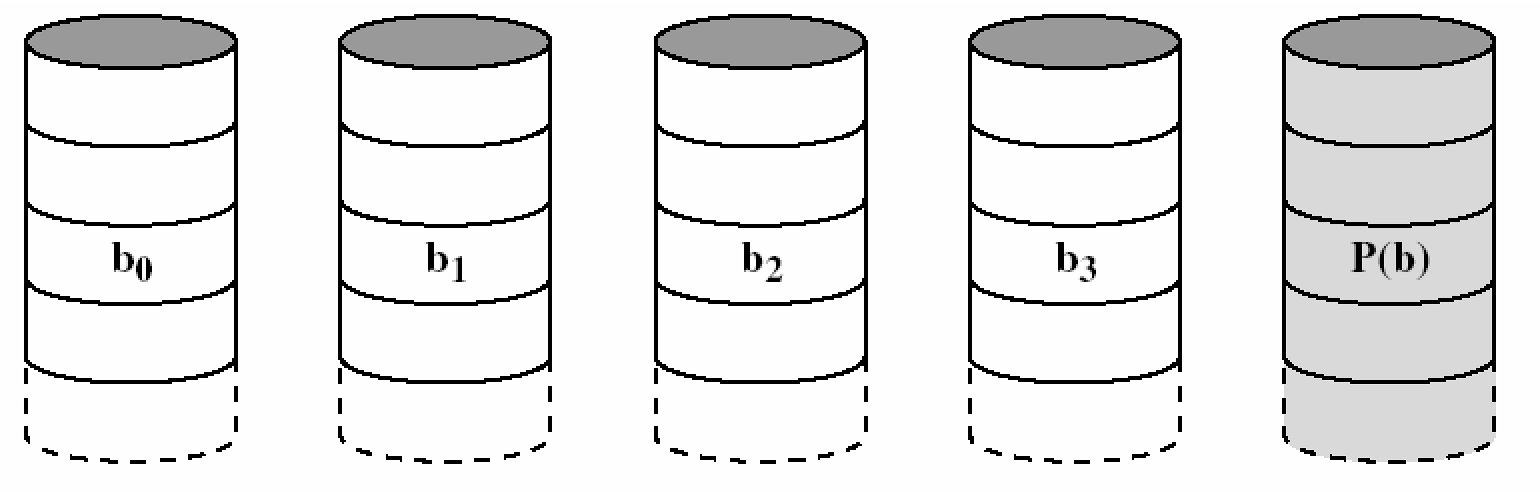
RAID 3

- Fortunately, if a disk fails, we know which bit is "wrong" and can use the parity bit to recover it
- Advantage:
  - Only need a single extra disk to implement RAID 3
- Can handle failure of complete disk



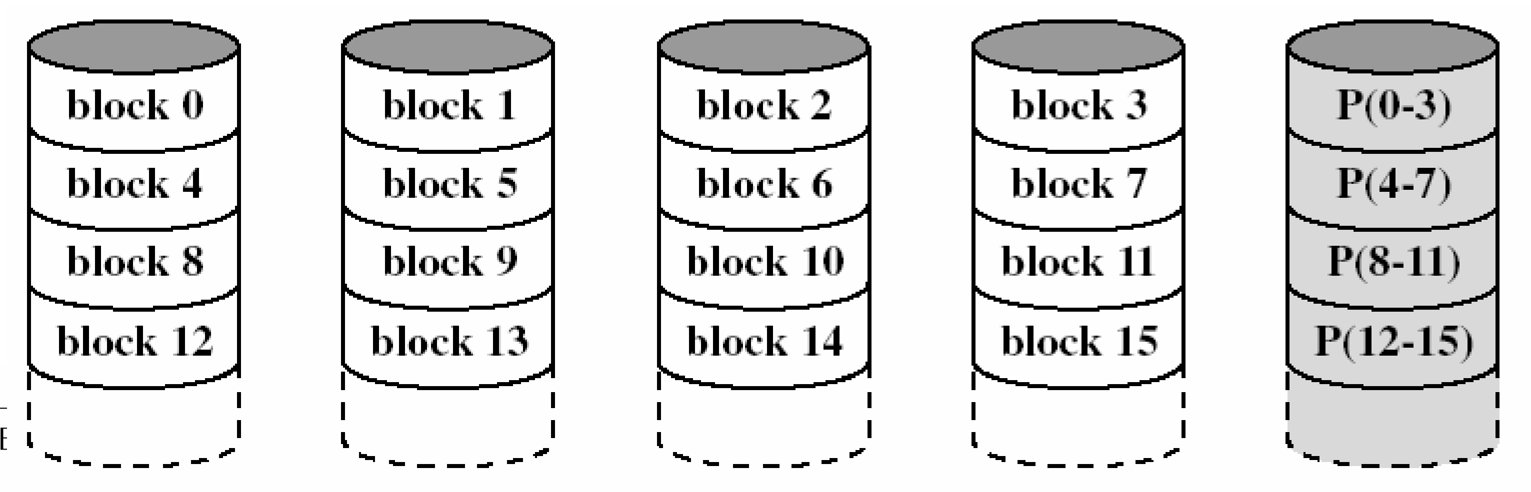
RAID 3

- Disadvantage:
  – Synchronised spindles
  – Fast for reading contiguous data, but does not improve performance for independent small requests
    - Each drive seeks together



RAID 4

- Parity computed on a block basis
  - Block 0-3 XOR'd together to generate a parity block
    - P block(x) = Block0(x) ⊗ Block1(x) ⊗ Block2(x) ⊗ Block3(x)
  - Parity stored on an extra disk
- Only needs one extra disk to implement
- Can handle failure of a single disk



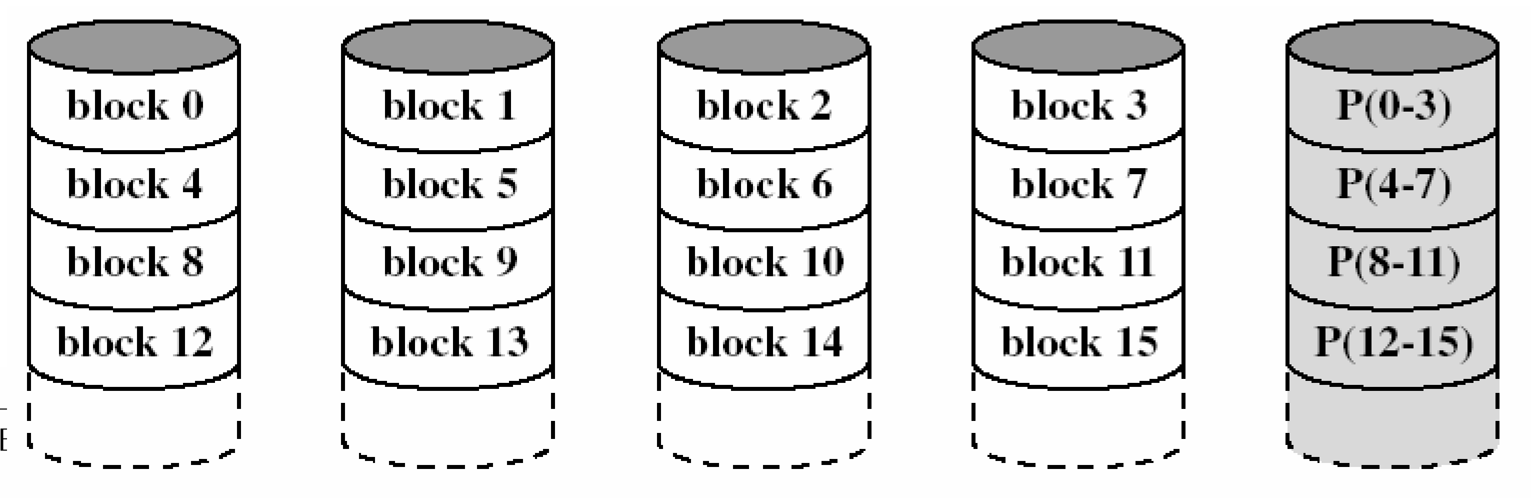
Examining the first byte in each block

	Byte 0
Block 0	`011010011`
Block 1	`111111010`
Block 2	`010000001`
Block 3	`001010100`
Parity	`111111100`

What is the lost byte?

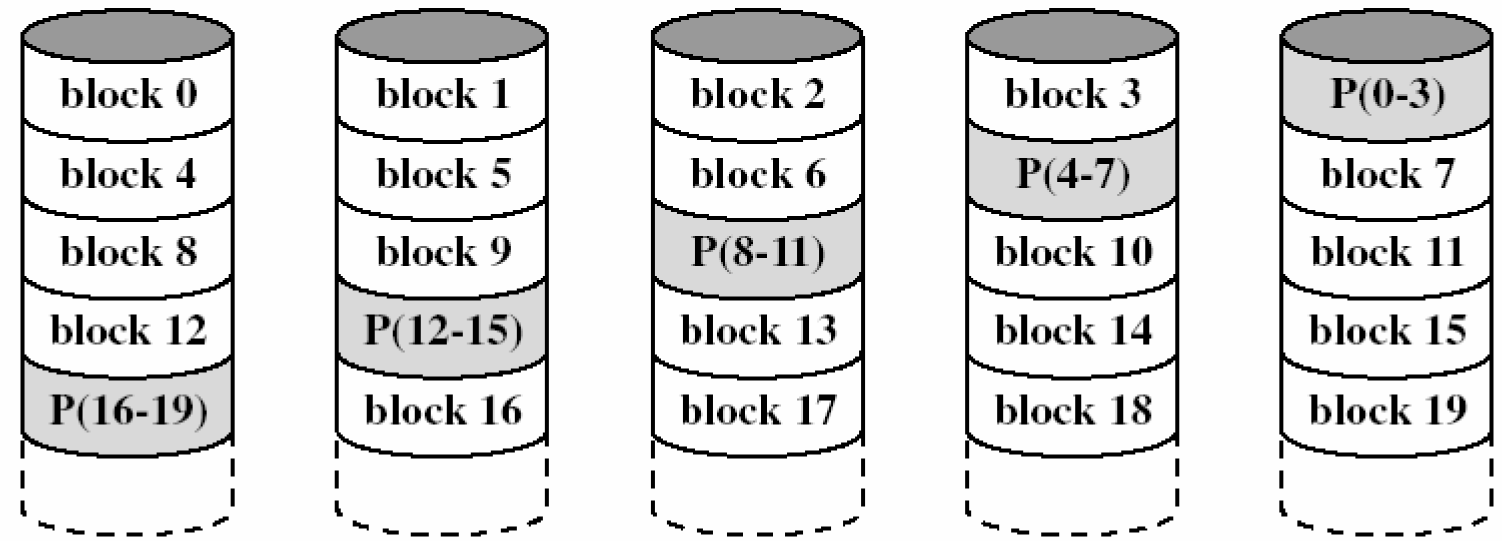
RAID 4

- Does not require synchronised spindles
- Can parallelised many independent request
- Small updates are a problem
  - Requires two reads (old block + parity) and two writes (new block + parity) to update a disk block
  - Parity disk may become a bottleneck



RAID 5

- Like RAID 4, except we distribute the parity on all disks
- Avoids parity disk updates becoming a bottleneck
- Update performance still less than a single disk
- Reconstruction after failure is tricky



Summary

- RAID 0 provides performance improvements, but no availability improvement
- RAID 1 (01,10) provides performance and availability improvements but expensive to implement (double the number of disks)
- RAID 5 is cheap (single extra disk), but has poor write update performance
- Others (2 & 3) are not used

HP AutoRAID

- Active data used RAID 1
  - Good read and write performance
- Inactive data uses RAID 5
  - Rarely accessed, RAID 5 provides low storage overheads
- Adaptive Storage
  - Empty disk uses entirely RAID 1, as disk fills, data incrementally converted to RAID 5 to increase available capacity
  - Data updates convert data back to RAID 1
- On-line array expansion
  - New disks can be added and system rebalances
  - New Disks can be an arbitrary size
- Active Hot Spare
  - The hot spare is used for mirroring until needed.

HP AutoRAID

- If you interested in the details see

John Wilkes, Richard Golding, Carl Staelin and Tim Sullivan. “The HP AutoRAID hierarchical storage system”, *ACM Trans. Comput. Syst.*, Vol 14(1), 1996