Handling Disk Failures.

- Types of failures of a disk drive

(1) Intermittent failure
- Disk ~~open~~ is not defective.
- Due to fluctuations in electrical current, sector was NOT written successfully.
- A repeated write (many n times) will be successful.

(2) Media decay
- a sector is worn
- can not be written correctly.

(3) Write failure
- A power failure occurred during a write operation.
- Data written to sector is corrupt
- ☆ (old data is lost also) Try to save to old !!! to old data !...

(4) Disk Crash
- Entire disk is unusable
- All data on Disk is lost

Handling Intermittent Failures

⛤ We need to ability to detect whether the data written is correct.

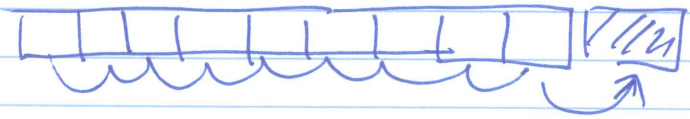
Naive Approach:

1. Write data to sector
2. Read the data back from sector.
3. Compare data read with orig. data

Write again if failure.

More efficient Approach: checksum.

1. Write data to sector



2. Add a checksum field at end.
3. Read the data back from sector
4. Re-compute checksum.
   - Verify checksum

Write again if checksum mismatch

(2) Handling Medra Decay:

- A bad/worn sector is marked a "bad sector"
  (Disk maintains a "bad sector table" and will not use these sectors)

- The block containing one or more "bad sectors" are listed in a "bad block table"
  - The rest of the disk can be used normally.

Handling Write failure

Fact: during a power failure, the content of main memory is lost

The NEW data will be lost from main memory + from the Disk (because it was incomplete).

There is no hope of getting the new data back.

The only hope is:

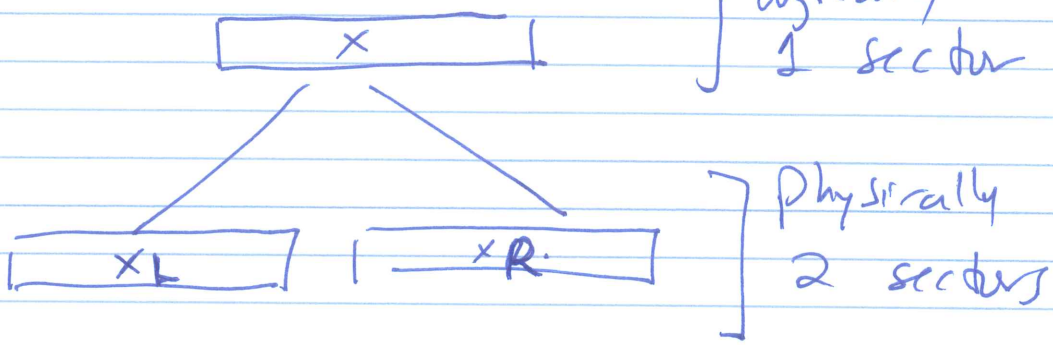
get the old data back.

⇓

Stable Storage update policy.

(Very expensive).

Stable Storage update policy:



Write policy to a sector x

(1) Write data to sector x_L first.

(2) Read data back + verify.

(Repeat a number of times)

(3) If sector x_L is bad

- Mark x_L as bad
- REplace x_L with another sector
- Repeat from (1)

and steps

Do the with x_R

(4) (x_L write successful).

Write data to ~~sto~~ sector x_L

Read back + verify

if (x_R bad) - replace

Read policy of the Stable Storage technique

```
While(tries < n)
{
    read (X_L)
    if ( check pass)
        return X_L
    read (X_R)
    if((check pass)
        return X_R
}
```

Recovery from Disk Crashes

Fact: recover from a disk crash always use:

redundancy.

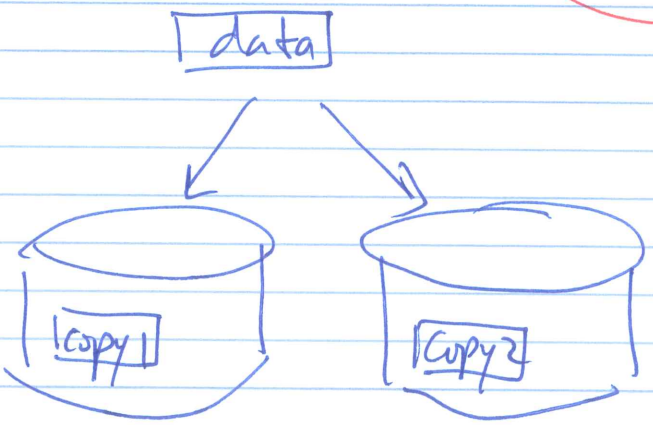
~~Fact~~:

Naive way to obtain redundancy:

disk mirroring

expensive

2x storage cost.



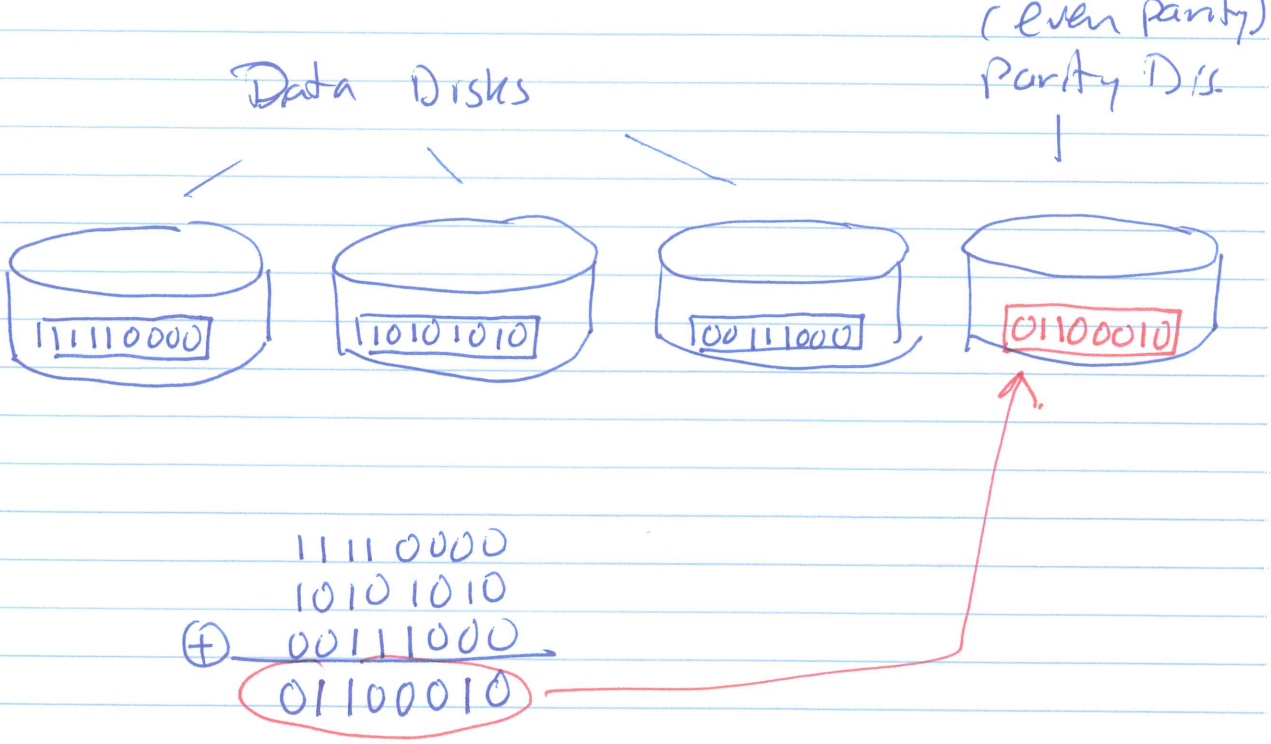
Technique is known as: RAID level 1

Redundant Array of Inexpensive Disks

original name → Independant → new name.

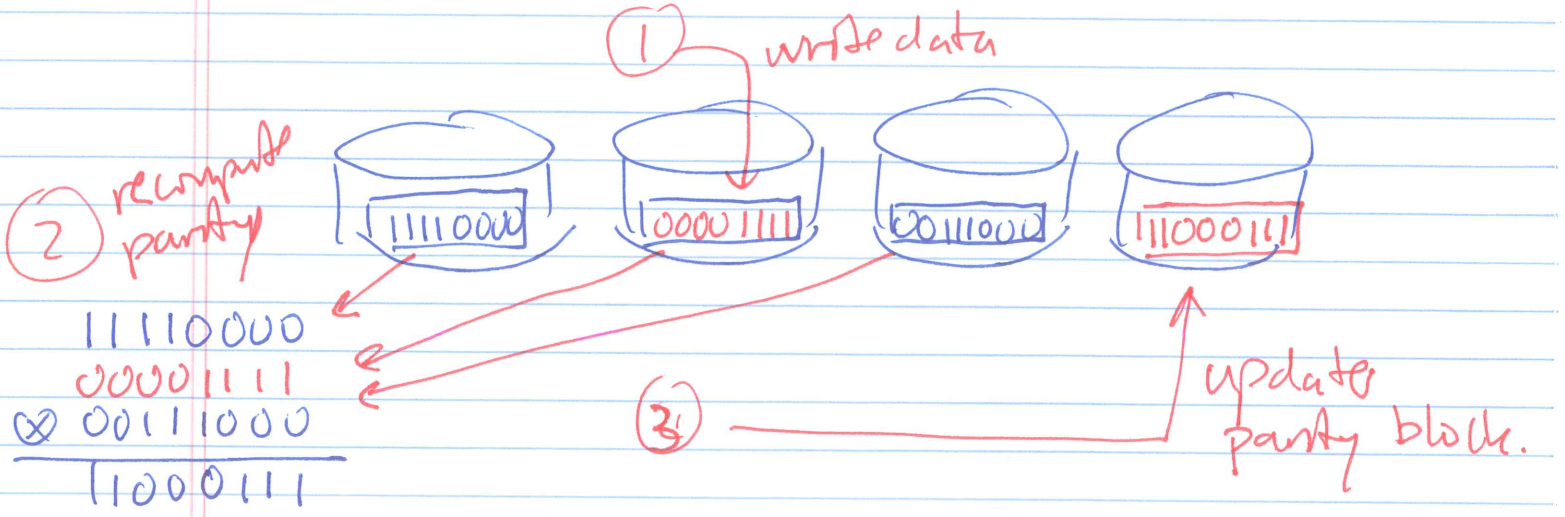
RAID - level 4 (Parity DISK).

(Modulo-2 sum = Exclusive OR)



Reading blocks from a data disk: just read.

Write block to data disk:



Better way to recompute parity:

Old data = 10101010

New data = 00001111 ⊛ XOR

10100101

Old parity data = 01100010

⊗ 10100101

11000111

New parity data.

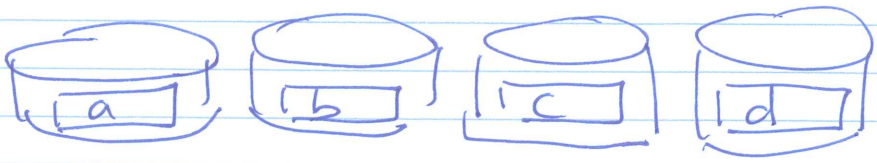
Due to these properties of XOR:

$$X \oplus Y = Y \oplus X$$

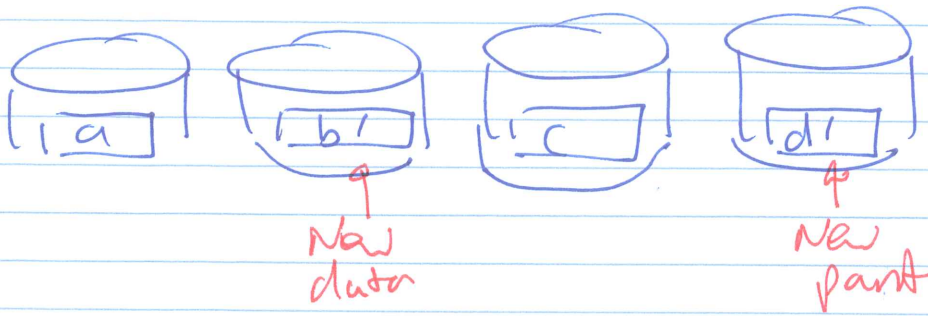
$$(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z)$$

$$X \oplus 0 = 0 \oplus X = X.$$

$$X \oplus X = 0 \quad !!!$$



$a \oplus b \oplus c = d$ ..... (1)



$a \oplus b' + c = d'$ ..... (2)

$a \oplus b \oplus c = d$ ..... (1).

$\Rightarrow a \oplus c \oplus b = d$

$\Rightarrow a \oplus c \oplus \underbrace{b \oplus b}_{0} = b \oplus d.$

$\Rightarrow a \oplus c = b \oplus d.$

Similarly: $a \oplus c = b' \oplus d'.$

$b' \oplus d' = (b \oplus d)$. old data, old parity

$d' = b' \oplus (b \oplus d)$.

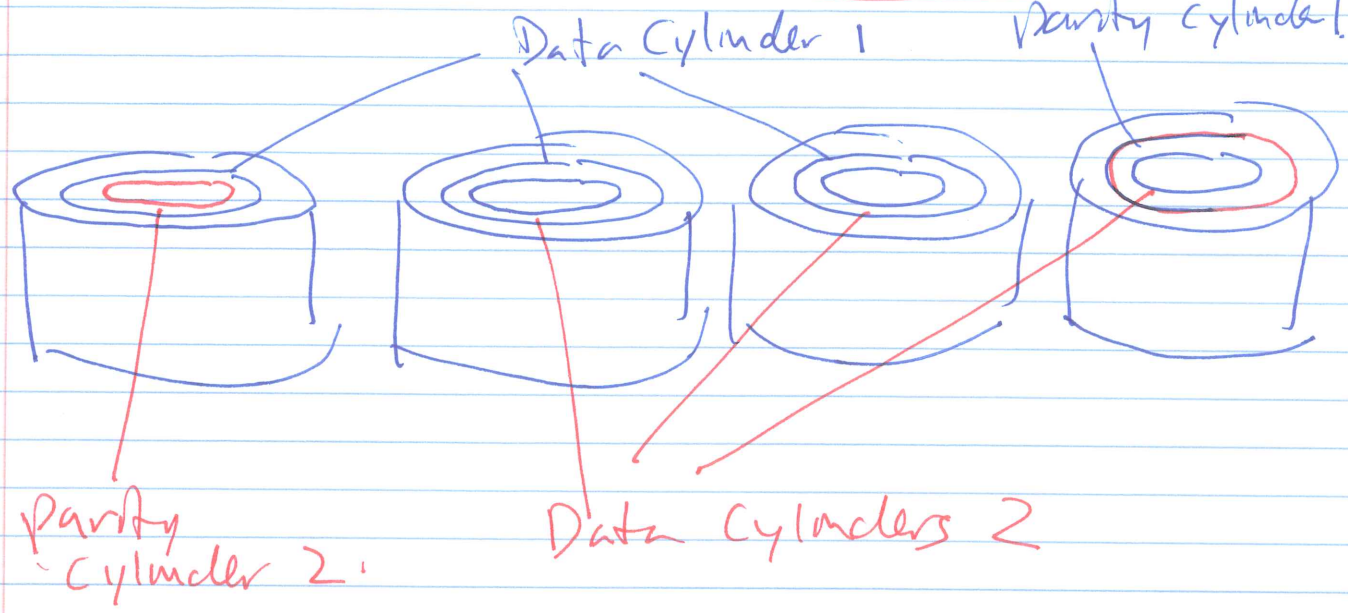
new data.

Problem with RAID 4

The parity DISK is a "Hot" spot for write operations !

RAID 5

RAID Level 5 : Parity CYLINDERS.



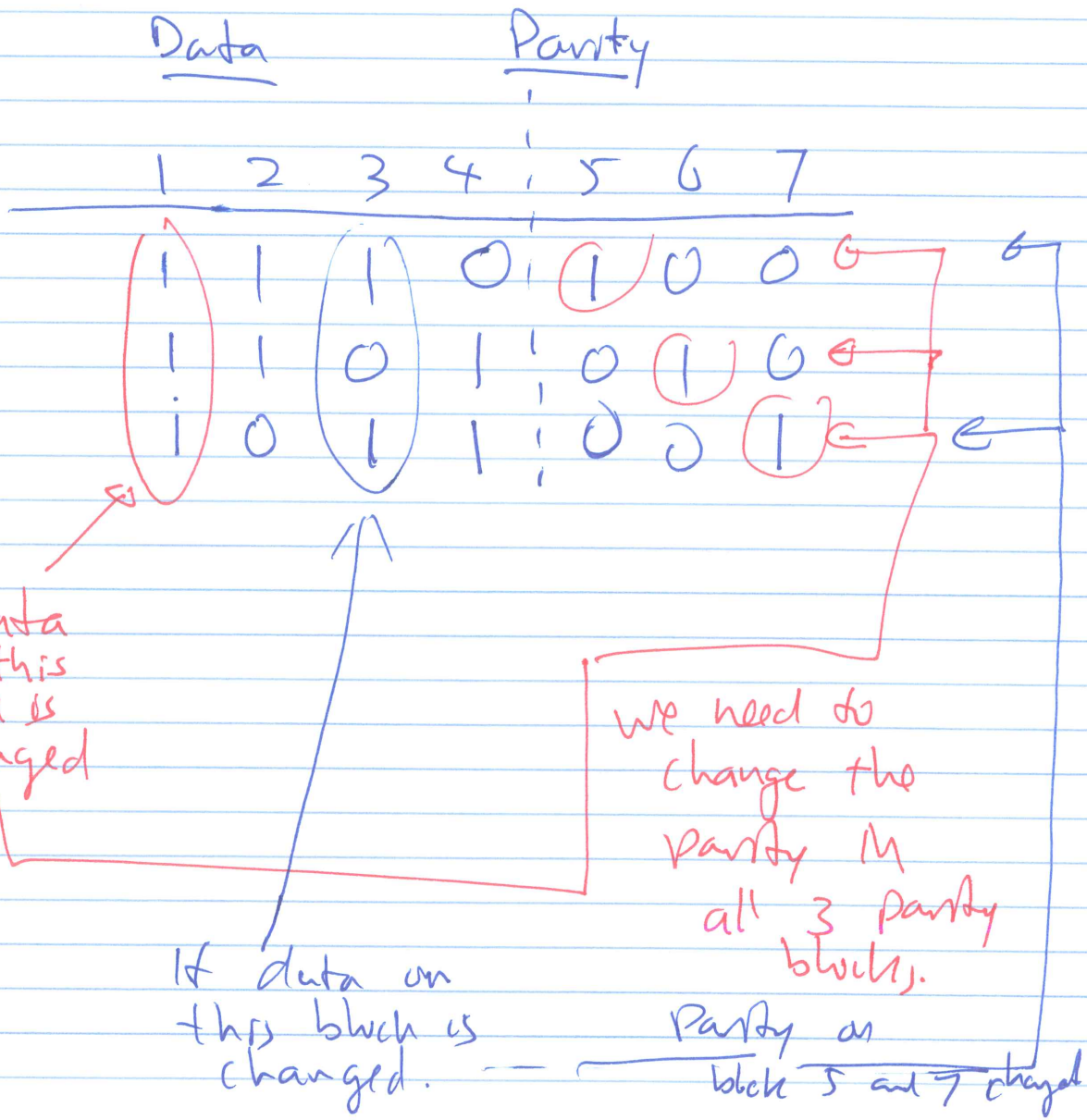
Now the write load to "parity cylinders" is distributed across all disk drives.

Protecting against 2 failures:

Need more disks.

Need more combinations

One example: protect against (2) simultaneous disk crashes.



Example:

Data Disks				Parity Disks		
1	2	3	4	5	6	7
1	1	1	0	1	0	0
1	1	0	1	0	1	0
1	0	1	1	0	0	1

Data Disks:	Content
1	11110000
2	10101010
3	00111000
4	01000001

Calc. the parity disk content:

(5): $1 \oplus 2 \oplus 3$.

$$\begin{array}{r} 11110000 \\ \Rightarrow\ 10101010 \\ 00111000 \\ \hline 01100010 \end{array}$$

(6): $1 \oplus 2 \oplus 4$

$$= 00011011$$

(7): $1 \oplus 3 \oplus 4$

$$= 10001001.$$

Su:

1	11110000	
2	10101010	X
3	00111000	
4	01000001	
5	01100010	X.
6	00011011	
7	10001001.	

Suppose Disk 2 + 5 fail simultaneously.

Look for this:

1	2	3	4	5	6	7
1	1	1	0	1	0	0
→ 1	1	0	1	0	1	0
1	0	1	1	0	0	1

Different value (one 0, one 1).

failed

Su: D1 ⊕ D2 ⊕ D4 ⊕ D6.

operational !!!

→ we can reconstruct D2 !!!

Suppose Disk 1 + 2 failed simultaneously

1	2	3	4	5	6	7
1	1	1	0	1	0	0
1	0	0	0	0	1	0
?	0	1	1	1	0	1

different

So use: $D1 \oplus D3 \oplus D4 \oplus D7 = 0$

failed. operational

We can use $D3, D4 \oplus D7$

$$D1 = D3 \oplus D4 \oplus D7.$$