Disk Arrays

Dave Eckhardt
de0u@andrew.cmu.edu

Synchronization

- Today: Disk Arrays
  - Text: 14.5 (a good start)
    - Please read remainder of chapter
  - www.acnc.com 's “RAID.edu” pages
    - Pittsburgh's own RAID vendor!
  - www.uni-mainz.de/~neuffer/scsi/what_is_raid.html
  - Papers (@ end)

Overview

- Historical practices
  - Striping, mirroring
- The reliability problem
- Parity, ECC, why parity is enough
- RAID “levels” (really: flavors)
- Applications
- Papers

Striping

- Goal
  - High-performance I/O for databases, supercomputers
  - “People with more money than time”
- Problems with disks
  - Seek time
  - Rotational delay
  - Transfer time

Seek Time

- Technology issues evolve slowly
  - Weight of disk head
  - Stiffness of disk arm
  - Positioning technology
- Hard to dramatically improve for some customers
- Sorry!

Rotational Delay

- How fast ***can*** we spin a disk?
  - Fancy motors, lots of power – spend more money
- Probably limited by data rate
  - Spin faster $\Rightarrow$ must process analog waveforms faster
  - Analog $\Rightarrow$ digital via ***serious*** signal processing
- Special-purpose disks generally spin ***a little*** faster
  - 1.5X, 2X – not 100X

Transfer Time

- Transfer time =
  - Assume seek & rotation complete
  - How fast to transfer ____ kilobytes?
- How to transfer faster?

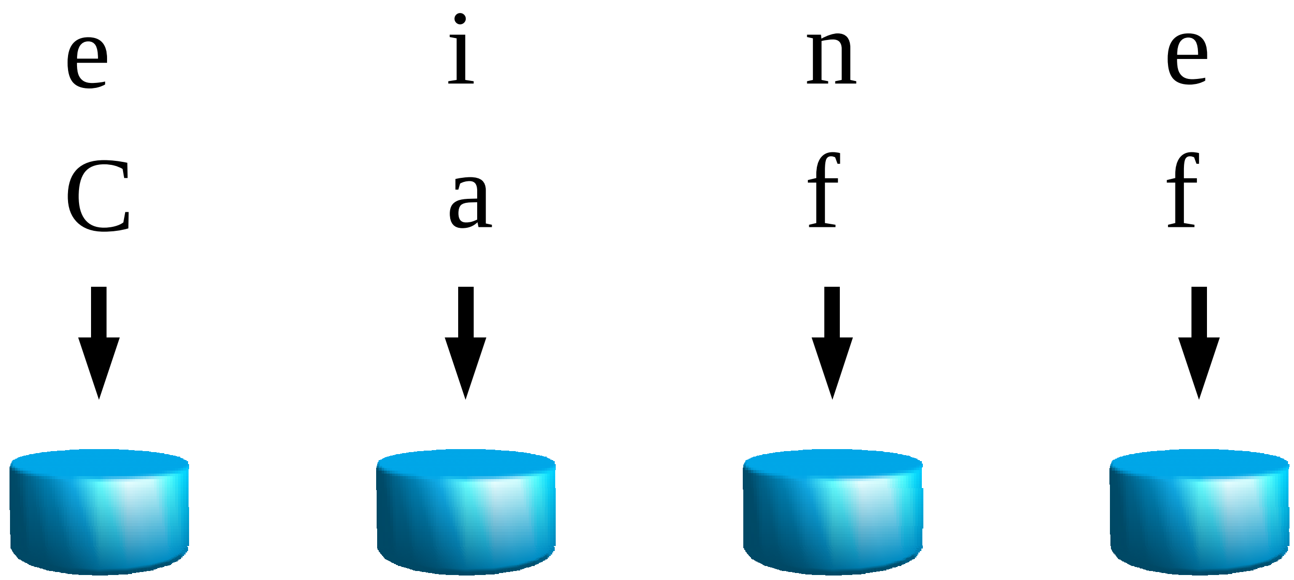
Parallel Transfer?

- Reduce transfer time (without spinning faster)
- Read from multiple heads at same time?
- Practical problem
  - Disk needs N copies of analog $\Rightarrow$ digital hardware
  - Expensive, but we have ***some*** money to burn
- Marketing problem
  - Do we have ***enough*** money to buy a new factory?
  - Can't we use our existing product somehow?

Striping

- Goal
  - High-performance I/O for databases, supercomputers
- Solution: parallelism
  - Gang ***multiple disks*** together

Striping



Striping

- Stripe ***unit*** (what each disk gets) can vary
  - Byte
  - Bit
  - Sector (typical)
- Stripe ***size*** = stripe unit X #disks
- Operation: “fat sectors”
  - File system maps bulk data request ⇒ N disk ops
  - Each disk reads/writes 1 sector

Striping Example

- 4 disks, stripe unit = 512 bytes
- Stripe size = 2K
- Seek time: 1X base case (ok)
- Transfer rate (2K stripe): 4X base case (great!)
- Rotational delay ***gets worse***
  - Must wait for ***fourth*** disk to rotate to right place
  - Single disk pays ***average*** rotational cost (50%)
  - N disks tend to pay ***worst-case*** rotational cost (100%)

Fixing Striping

- Rotational delay ***gets worse***
  - Cannot wait for Nth disk to rotate
- Spindle synchronization!
  - Make sure N platters are always aligned
  - Sector 0 passes under each head at “same” time
- Result
  - Commodity disks with extra synchronization hardware

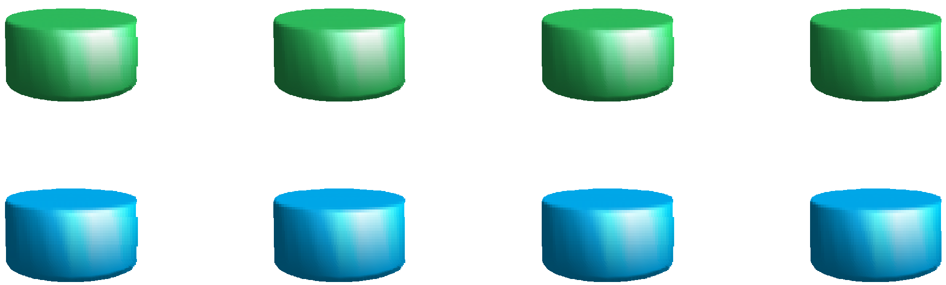
Less Esoteric: Capacity

- Users always want more disk space
- Easy answer
  - Build a larger disk!
  - IBM 3380: size of refrigerator
- “Marketing on line 1”...
  - These monster disks sure are expensive to build!
  - Can't we hook small disks together like last time?

The Reliability Problem

- MTTF = Mean time to failure
- MTTF(array) = MTTF(disk) / #disks
- Example from original 1988 RAID paper
  - Connors CP3100 (100 megabytes!)
  - MTTF = 30,000 hours = 3.4 years
- Array of 100 CP3100's
  - MTTF = 300 hours = ***12.5 days***
  - Reload array from tape every 2 weeks???

Mirroring



Copy A

Copy B

Mirroring

- Operation
  - Write: write to ***both*** mirrors
  - Read: read from ***either*** mirror
- Cost per byte ***doubles***
- Performance
  - Writes: a little slower
  - Reads: maybe 2X faster
- Reliability ***vastly*** increased

Mirroring

- When a disk breaks
  - Identify it to system administrator
    - Beep, blink a light
  - System administrator provides blank disk
  - Copy contents from surviving mirror
- Result
  - Expensive but safe
  - Banks, hospitals, etc.
  - Home PC users???

Error Coding

- If you are good at math
  - Lin, Shu, & Costello
  - Error Control Coding: Fundamentals & Applications
- If you are like me
  - Arazi
  - Commonsense Approach to the Theory of Error Correcting Codes

Error Coding In One Easy Lesson

- Data vs. message
  - Data = what you want to convey
  - Message = data plus extra bits ("code word")
- Error detection
  - Message indicates: something got corrupted
- Error ***correction***
  - Message indicates: bit 37 should be 0, not 1
  - Very useful!

Lesson 1, Part B

- Error codes can be overwhelmed
- “Too many” errors: ***wrong answers***
- Can typically detect more errors than can correct
  - Code Q
    - Can detect 1..4 errors, can fix any single error
    - Five errors will report “fix” - to a ***different*** user data word!

Parity

- Parity = XOR "sum" of bits
  - $0 \oplus 1 \oplus 1 = 0$
- Parity provides ***single error detection***
  - Sender provides ***code word*** and ***parity bit***
  - Correct: 011,0
  - Incorrect: 011,1
    - Something is wrong with this picture – ***but what?***
- ***Cannot*** detect (all) multiple-bit errors

ECC

- ECC = error correcting code
- “Super parity”
  - Code word, ***multiple*** “parity” bits
  - Mysterious math computes parity from data
    - Hamming code, Reed-Solomon code
  - Can detect N *multiple-bit* errors
  - Can ***correct*** M (< N) bit errors!
  - Often M ~ N/2

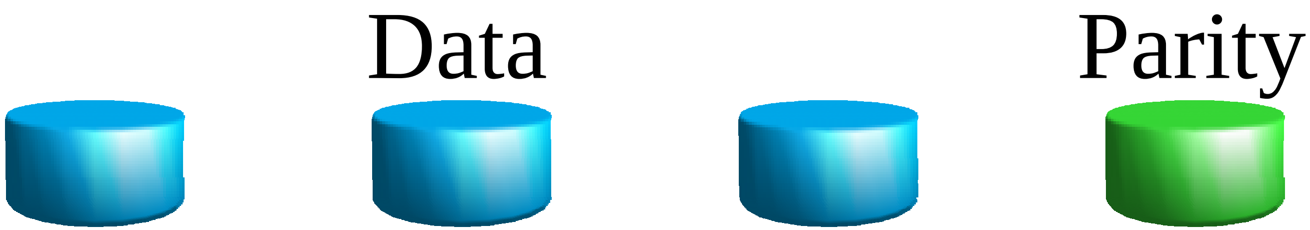
Parity revisited

- Parity provides single ***erasure*** correction!
- Erasure channel
  - Knows when it doesn't know something
  - Each bit is 0 or 1 or “don't know”
- Sender provides code word, parity bit: (0 1 1 , 0)
- Channel provides corrupted message: (0 ? 1 , 0)
- $? = 0 \oplus 1 \oplus 0 = 1$

Erasure channel???

- Are erasure channels real?
- Radio
  - signal strength during reception of bit
- Disk drives!
  - Each sector is stored with CRC
    - Read sector 42 from 4 disks
    - Receive 0..4 good sectors, 4..0 errors
  - “Drive not ready” = “erasure” of all sectors

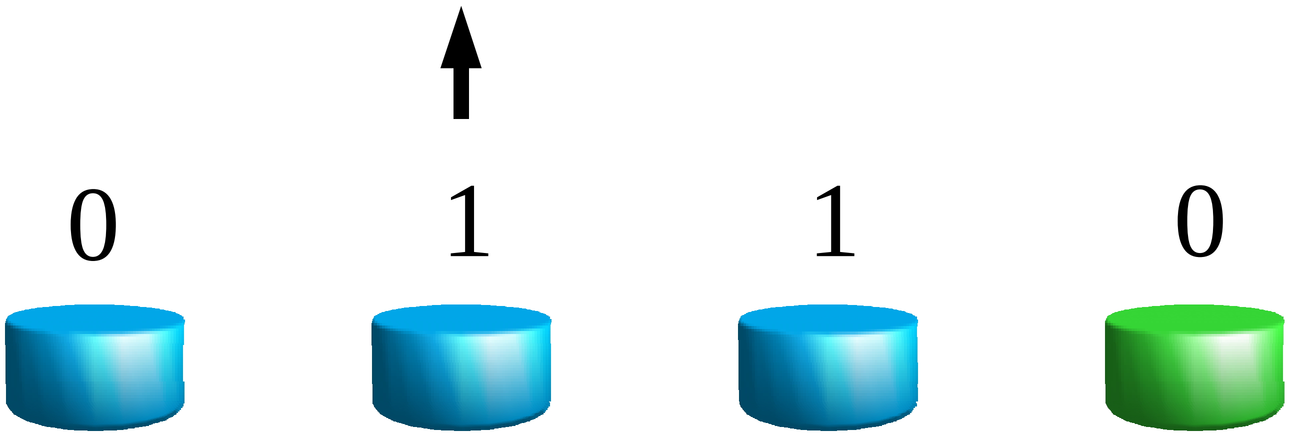
“Fractional mirroring”



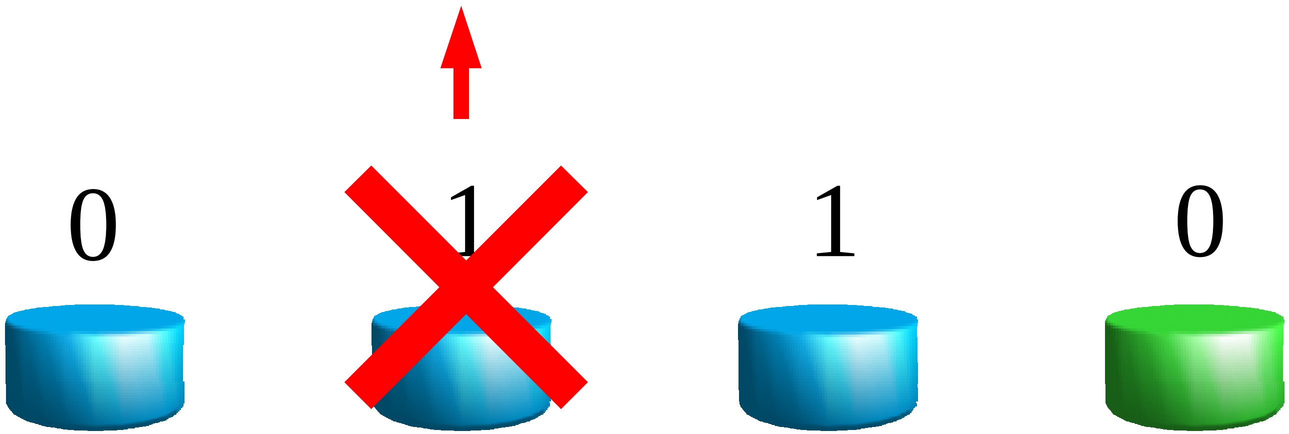
“Fractional mirroring”

- Operation
  - Read: read data disks
    - Error? Read parity disk, compute lost value
  - Write: write data disks ***and parity disk***

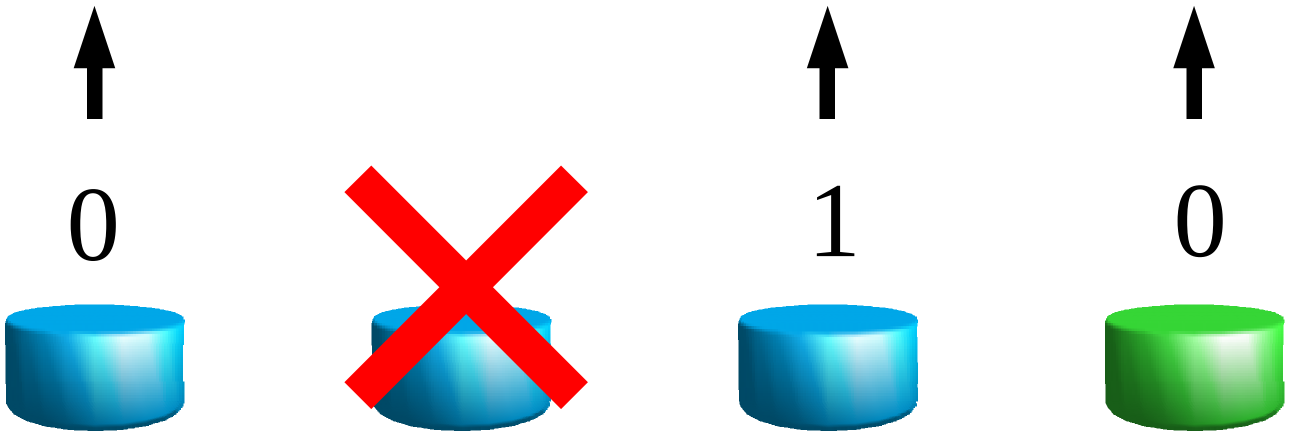
Read



Read Error



Read Reconstruction



Missing = 0 ⊕ 1 ⊕ 0 = 1

“Fractional mirroring”

- Performance
  - Writes: slower (see “RAID 4” below)
  - Reads: unaffected
- Reliability ***vastly*** increased
  - Not quite as good as mirroring
    - Why not?

“Fractional mirroring”

- Cost
  - ***Fractional*** increase (50%, 33%, ...)
  - Cheaper than mirroring's 100%

RAID

- RAID
  - Redundant Arrays of Inexpensive Disks
- SLED
  - Single Large Expensive Disk
- Terms from original RAID paper (@end)
- Different ways to aggregate disks
  - Paper presented a number-based taxonomy
  - Metaphor tenuous then, stretched ridiculously now

RAID “levels”

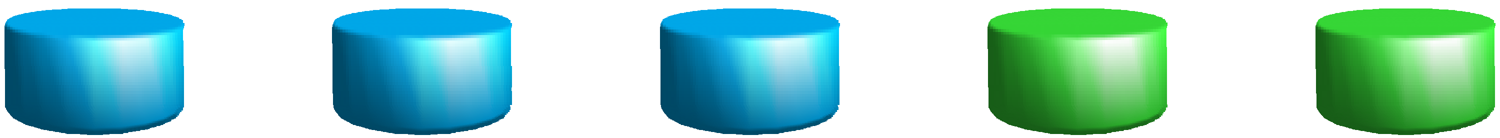
- They're not really levels
  - RAID 2 isn't “more advanced than” RAID 1
    - People really do RAID 1
    - People basically never do RAID 2
- People invent new ones randomly
  - RAID 0+1 ???
  - JBOD ???

Easy cases

- JBOD = "just a bunch of disks"
  - N disks in a box pretending to be 1 large disk
  - Box controller maps "sector" ⇒ disk, sector
- RAID 0 = striping
- RAID 1 = mirroring

RAID 2

- Stripe size = ***byte*** (unit = 1 bit per disk)
- N data disks, M parity disks
- Use ECC to get multiple-error correction
- Very rarely used



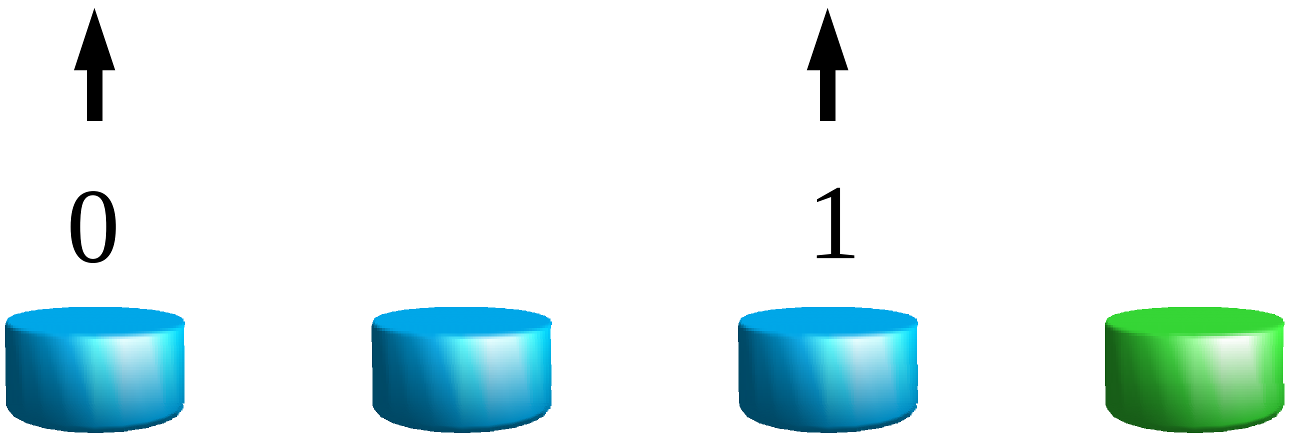
RAID 3

- Stripe size = ***byte*** (unit = 1 bit per disk)
- Use parity instead of ECC (disks report erasures)
- N data disks, 1 parity disk
- Used in some high-performance applications

RAID 4

- RAID 3, unit = ***sector*** instead of ***bit***

Single-sector reads: Parallel!



Single-sector ***writes***

- Modifying a single sector is harder
- Must fetch old version of sector
- Must maintain parity invariant for stripe

Sector write

RAID 4

- RAID 3, unit = ***sector*** instead of ***bit***
- Single-sector reads involve only 1 disk: parallel!
- Single-sector writes: read, read, write, write!
- Rarely used: parity disk is a ***hot spot***

RAID 5

- RAID 4, distribute parity among disks
- No more “parity disk hot spot”
- Frequently used

Other fun flavors

- RAID 6, 7, 10, 53
  - Esoteric, single-vendor, non-standard terminology
- RAID 0+1
  - Stripe data across half of your disks
  - Use the other half to mirror the first half
  - Sensible if you like mirroring but need lots of space

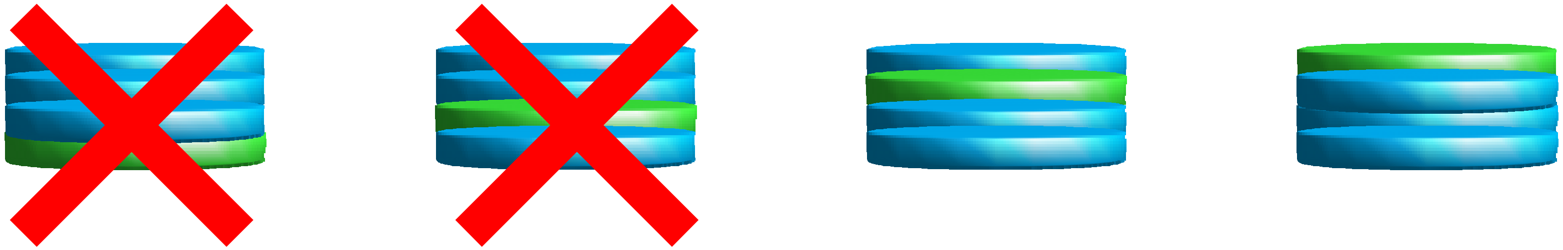
Applications

- RAID 0
  - Supercomputer temporary storage / swapping
- RAID 1
  - Simple to explain, reasonable performance, expensive
  - Traditional high-reliability applications (banking)
- RAID 5
  - Cheap reliability for large on-line storage
  - AFS servers

Are failures independent?

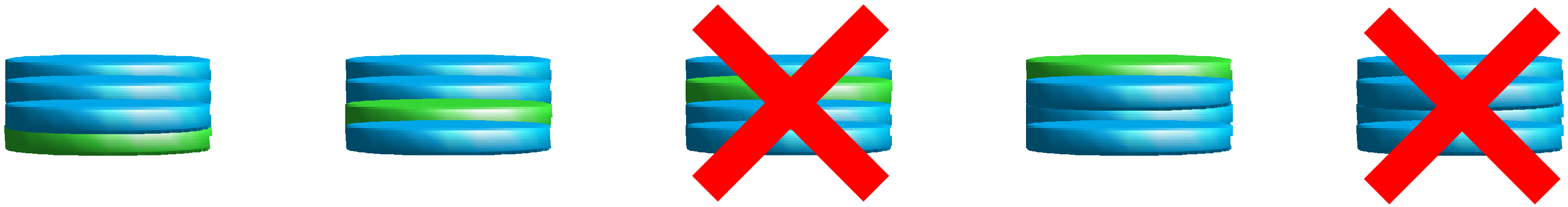
- With RAID (1-5) disk failures are “ok”
- ***Array*** failures are never ok
  - “Too many” disk failures “too soon”
  - No longer possible to recompute original data
  - Hope your backup tapes are good...
  - ...and your backup system is tape-drive-parallel!
- #insert <quad-failure.story>

Are failures independent?



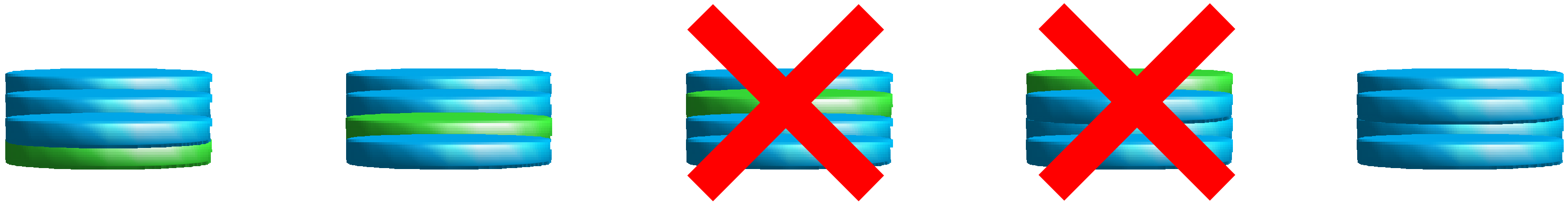
- Hint: IDE

Are failures independent?



- Hint: test before trust!

Are failures independent?



- Hint: some days are bad days

Papers

- 1988: Patterson, Gibson, Katz: A Case for Redundant Arrays of Inexpensive Disks (RAID), www.cs.cmu.edu/~garth/RAIDpaper/Patterson88.pdf
- 1990: Chervenak, Performance Measurements of the First RAID Prototype, isi.edu/~annc/papers/masters.ps
- Countless others

Summary

- Need more disks!
  - More space, lower latency, more throughput
- ***Cannot*** tolerate 1/N reliability
- Store information carefully and redundantly
- Lots of variations on a common theme
- You should understand RAID 0, 1, 5