Distributed Systems

15-440 / 15-640

13 – Errors, Fault Tolerance and RAID

Tuesday, Oct 9th, 2018

Fault Tolerance Techniques So Far?

- Redundancy: information / time / physical redundancy
  - E.g., used in airplanes
- Recovery: checkpointing and logging (ARIES)
  - E.g., used in commercial databases
- Distributed Replication: Paxos
  - E.g., Survive the failure of up to f replicas
- How about *data errors in communication and storage*?

  → Main topic for this lecture

Outline

- Errors/error recovery
- Using multiple disks
  - Why have multiple disks?
  - problem and approaches
- RAID levels and performance
- Estimating availability

Type of Errors

- **Hard errors**: The component is dead.
- **Soft errors**: A signal or bit is wrong, but it doesn't mean the component must be faulty
- Note: You can have recurring soft errors due to faulty, but not dead, hardware

Examples

- DRAM errors
  - Hard errors: Often caused by motherboard - faulty traces, bad solder, etc.
  - Soft errors: Often caused by cosmic radiation or alpha particles (from the chip material itself) hitting memory cell, changing value. (Remember that DRAM is just little capacitors to store charge... if you hit it with radiation, you can add charge to it.)

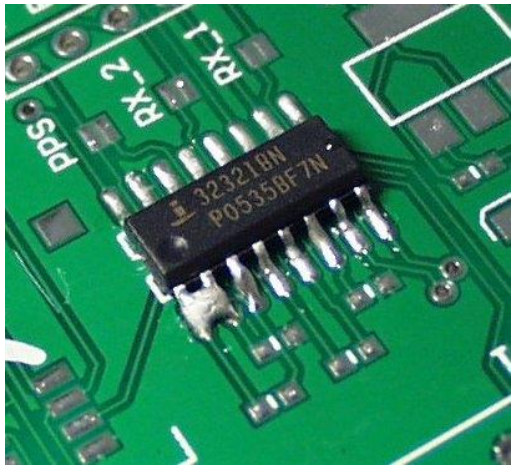


Image from
https://www.sparkfun.com/tutorials/96

Some fun #s

- Both Microsoft and Google have recently started to identify DRAM errors as an increasing contributor to failures... Google in their datacenters, Microsoft on your desktops.

- We've known hard drives fail
  - Especially when students need to hand in HW/projects :)

E.g., See "DRAM Errors in the Wild: A Large-Scale Field Study"

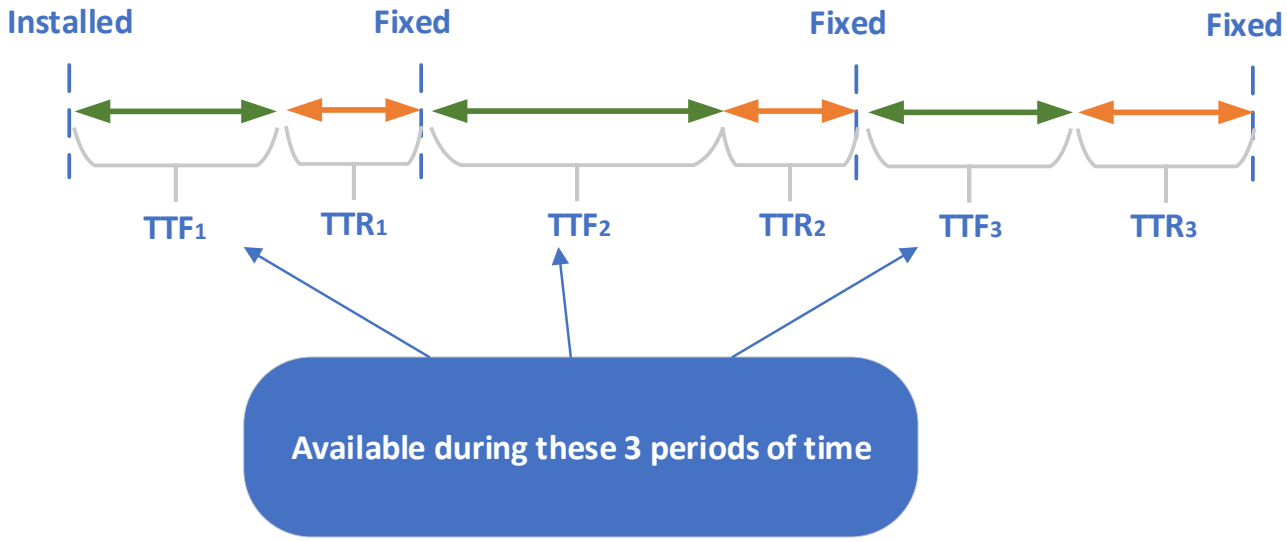
Replacement Rates

HPC1		COM1		COM2	
Component	**%**	**Component**	**%**	**Component**	**%**
Hard drive	30.6	Power supply	34.8	Hard drive	49.1
Memory	28.5	Memory	20.1	Motherboard	23.4
Misc/Unk	14.4	Hard drive	18.1	Power supply	10.1
CPU	12.4	Case	11.4	RAID card	4.1
motherboard	4.9	Fan	8	Memory	3.4
Controller	2.9	CPU	2	SCSI cable	2.2
QSW	1.7	SCSI Board	0.6	Fan	2.2
Power supply	1.6	NIC Card	1.2	CPU	2.2
MLB	1	LV Pwr Board	0.6	CD-ROM	0.6
SCSI BP	0.3	CPU heatsink	0.6	Raid Controller	0.6

From “Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you?”

Measuring Availability

- Fraction of time that server is able to handle requests
  - Computed from MTTF and MTTR (Mean Time To Repair)



$$\text{Availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

Measuring Availability

- Mean time to failure (MTTF) - “uptime”
- Mean time to repair (MTTR)
- Mean time between failures (MTBF)
- MTBF = MTTF + MTTR

- Availability = MTTF / (MTTF + MTTR)
  - Suppose OS crashes once per month,
    takes 10min to reboot.
  - MTTF ≈ 24 hours X 30 = 720 hours = 43,200 minutes
    MTTR = 10 minutes
  - Availability = 43200 / 43210 = 0.997 (~“3 nines”)

Availability

Availability %	Downtime per year	Downtime per month*	Downtime per week
90% ("one nine")	36.5 days	72 hours	16.8 hours
95%	18.25 days	36 hours	8.4 hours
97%	10.96 days	21.6 hours	5.04 hours
98%	7.30 days	14.4 hours	3.36 hours
99% ("two nines")	3.65 days	7.20 hours	1.68 hours
99.50%	1.83 days	3.60 hours	50.4 minutes
99.80%	17.52 hours	86.23 minutes	20.16 minutes
99.9% ("three nines")	8.76 hours	43.8 minutes	10.1 minutes
99.95%	4.38 hours	21.56 minutes	5.04 minutes
99.99% ("four nines")	52.56 minutes	4.32 minutes	1.01 minutes
99.999% ("five nines")	5.26 minutes	25.9 seconds	6.05 seconds
99.9999% ("six nines")	31.5 seconds	2.59 seconds	0.605 seconds
99.99999% ("seven nines")	3.15 seconds	0.259 seconds	0.0605 seconds

Availability in practice

- Carrier airlines (2002 FAA fact book)
  - 41 accidents, 6.7M departures
  - 99.9993% availability
- 911 Phone service (1993 NRIC report)
  - 29 minutes per line per year
  - 99.994%
- Standard phone service (various sources)
  - 53+ minutes per line per year
  - 99.99+%
- End-to-end Internet Availability
  - 95% - 99.6%

Real Devices

Seagate
We turn on ideas

PRODUCT OVERVIEW

Cheetah 15K.4

Mainstream enterprise disc drive

Simply the best price/ performance, lowest cost of ownership disc drive ever

KEY FEATURES AND BENEFITS

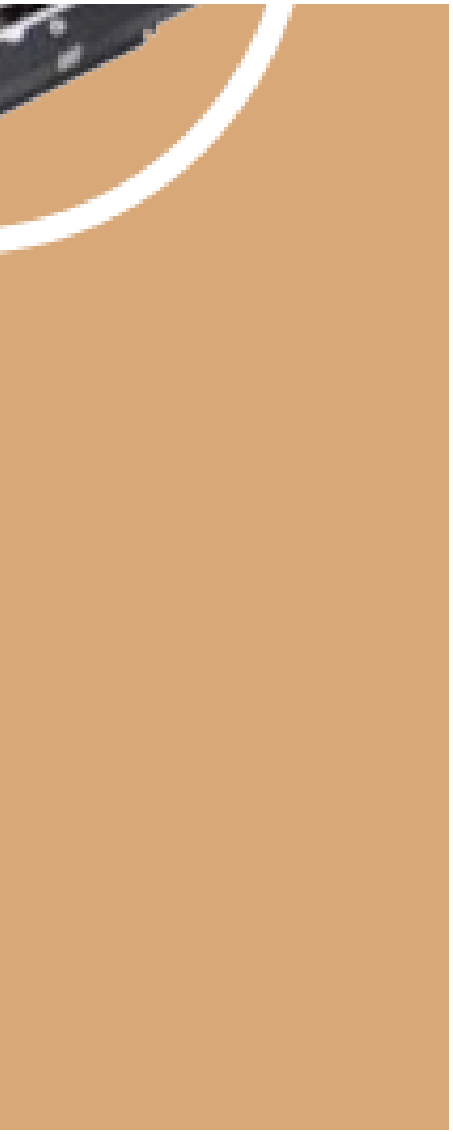
- The Cheetah® 15K.4 is the highest-performance drive ever offered by Seagate®, delivering maximum IOPS with fewer drives to yield lower TCO.
- The Cheetah 15K.4 price-per-performance value united with the breakthrough benefits of serial attached SCSI (SAS) make it the optimal 3.5-inch drive for rock solid enterprise storage.
- Proactive, self-initiated background management functions improve media integrity, increase drive efficiency, reduce incidence of integration failures and improve field reliability.
- The Cheetah 15K.4 shares its electronics architecture and firmware base with Cheetah 10K.7 and Savvio™ to ensure greater factory consistency and reduced time to market.

KEY SPECIFICATIONS

- 146-, 73- and 36-Gbyte capacities
- 3.3-msec average read and 3.8-msec average write seek times
- Up to 96-Mbytes/sec sustained transfer rate
- 1.4 million hours full duty cycle MTBF
- Serial Attached SCSI (SAS), Ultra320 SCSI and 2 Gbits/sec Fibre Channel interfaces
- 5-year warranty

For more information on why 15K is the industry's best price/performance disc drive for use in mainstream storage applications, visit ***http://specials.seagate.com/15k***

Real Devices – the small print



- The Cheetah 15K.4 is the highest-performance drive ever offered by Seagate®, delivering maximum IOPS with fewer drives to yield lower TCO.
- The Cheetah 15K.4 price-per-performance value united with the breakthrough benefits of serial attached SCSI (SAS) make it the optimal 3.5-inch drive for rock solid enterprise storage.
- Proactive, self-initiated background management functions improve media integrity, increase drive efficiency, reduce incidence of integration failures and improve field reliability.
- The Cheetah 15K.4 shares its electronics architecture and firmware base with Cheetah 10K.7 and Savvio™ to ensure greater factory consistency and reduced time to market.

KEY SPECIFICATIONS

- 146-, 73- and 36-Gbyte capacities
- 3.3-msec average read and 3.8-msec average write seek times
- Up to 96-Mbytes/sec sustained transfer rate
- 1.4 million hours full duty cycle MTBF
- Serial Attached SCSI (SAS), Ultra320 SCSI and 2 Gbits/sec Fibre Channel interfaces
- 5-year warranty

For more information on why 15K is the industry's best price/performance disc drive for use in mainstream storage applications, visit **http://specials.seagate.com/15k**

Real Devices – the small print

170 years….??!

- The Cheetah 15K.4 is the highest-performance drive ever offered by Seagate®, delivering maximum IOPS with fewer drives to yield lower TCO.
- The Cheetah 15K.4 price-per-performance value united with the breakthrough benefits of serial attached SCSI (SAS) make it the optimal 3.5-inch drive for rock solid

ed background management functions improve media integrity,
ncy, reduce incidence of integration failures and improve

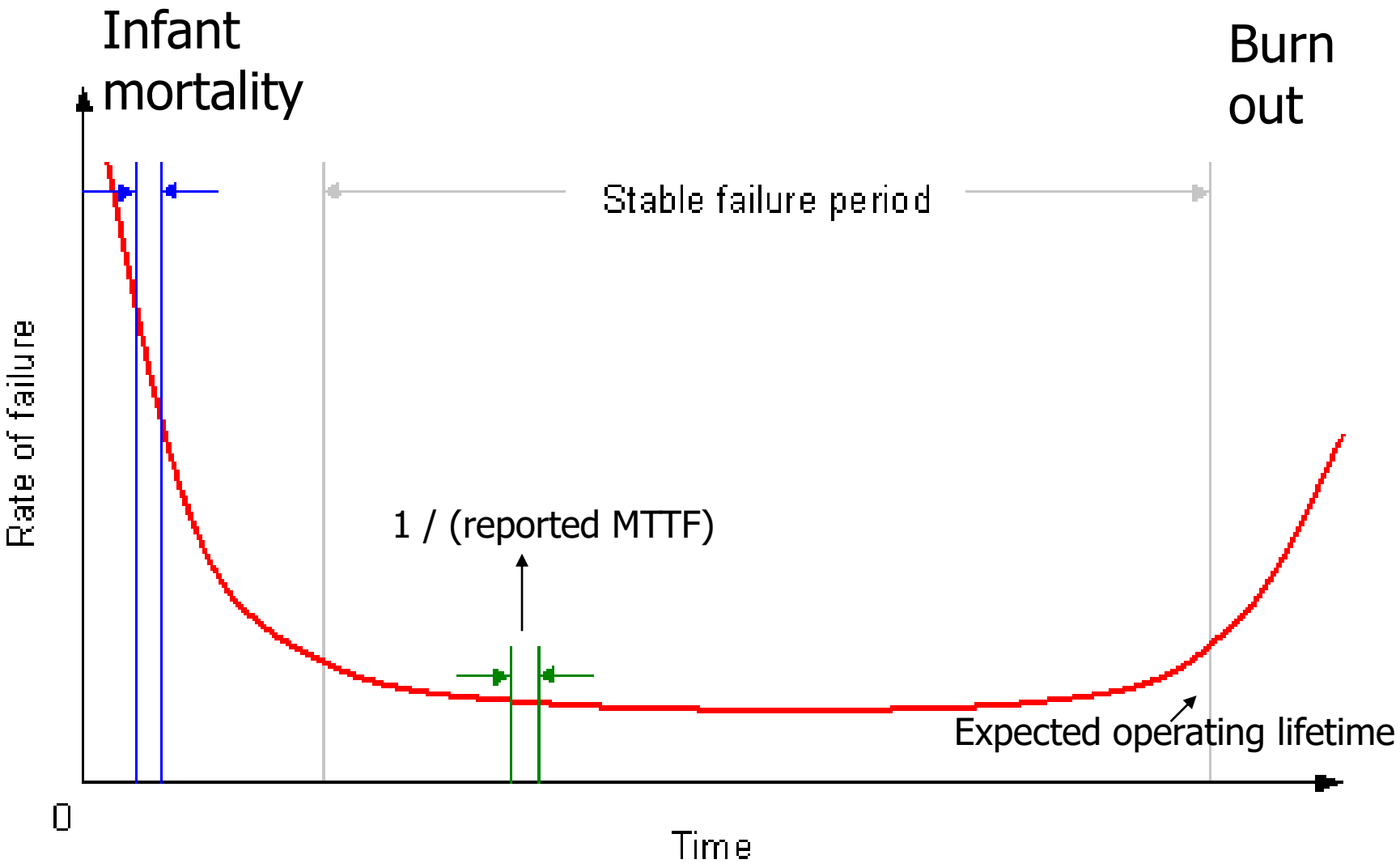
lecture and firmware base with
actory consistency and reduced

write seek times

2 Gbits/sec Fibre Channel interfaces

's best price/performance disc drive for
use in mainstream storage applications, visit **http://specials.seagate.com/15k**

Disk failure conditional probability distribution - Bathtub curve



Coping with failures...

- A failure
  - Let's say one bit in your DRAM fails.
- Propagates
  - Assume it flips a bit in a memory address the kernel is writing to. That causes a big memory error elsewhere, or a kernel panic.
  - This program is running one of a dozen storage servers for your distributed filesystem.
  - A client can't read from the DFS, so it hangs.
  - A professor can't check out a copy of your 15-440 assignment, so he gives you an F.

What are our options?

1. Silently return the wrong answer.

2. Detect failure.

3. Correct / mask the failure

Parity Checking

1 bit to detect error

101110 _

6 data bits

Single Bit Parity:

Detect single bit errors

Even parity bit

If odd # of 1 in D:

parity bit = 1

else:

parity bit = 0

101110 0

101110 0

0

Calculated parity bit: 1
Recorded parity bit: 0
Error detected!!

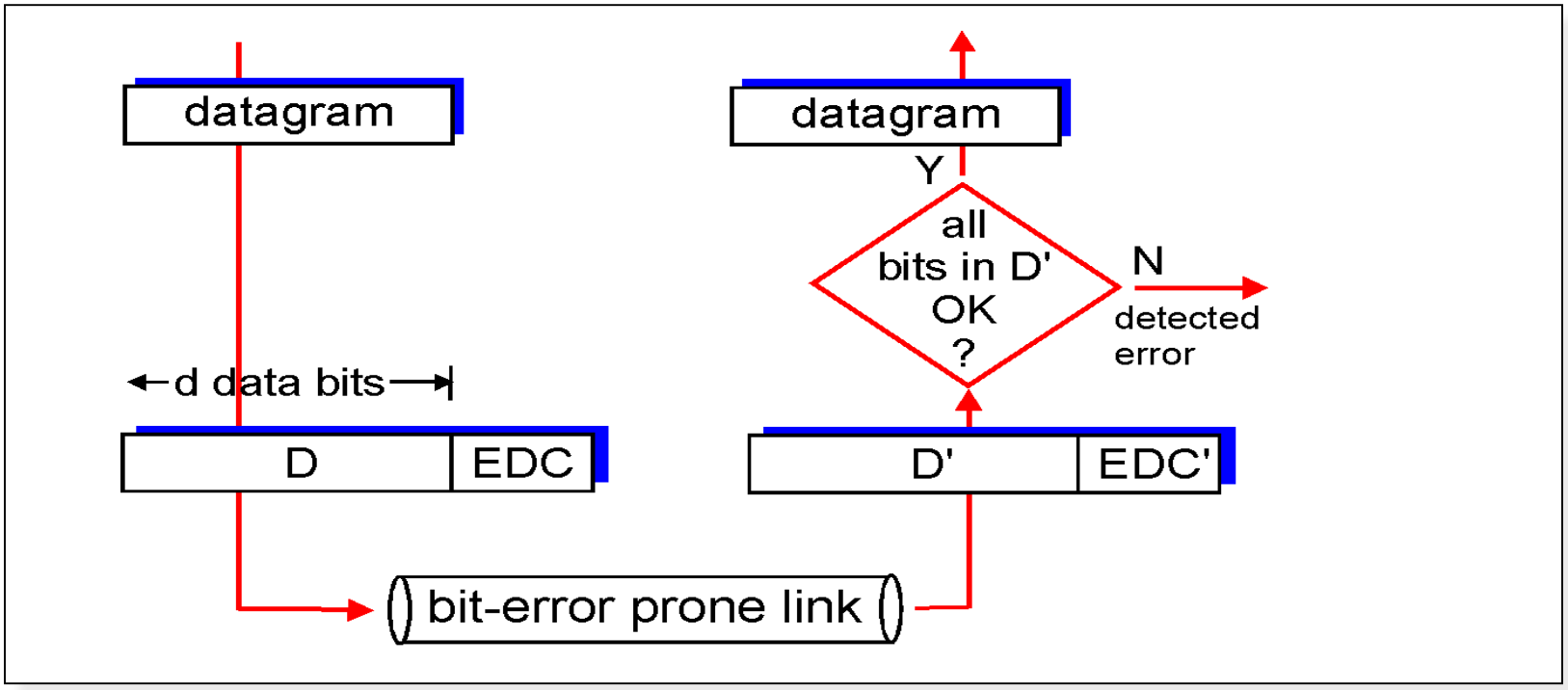
101110 0

10

Calculated parity bit: 1
Recorded parity bit: 1
No error detected!!

Block Error Detection

- EDC= Error Detection and Correction bits (redundancy)
- D = Data protected by error checking, may include header fields
- Error detection not 100% reliable!
  - Protocol may miss some errors, but rarely
  - Larger EDC field yields better detection and correction



Error Detection - Checksum

- Used by TCP, UDP, IP, etc..
- Ones complement sum of all words/shorts/bytes in packet
- Simple to implement
- Relatively weak detection
  - Easily tricked by typical error patterns – e.g. bit flips

Error Detection - Checksum

- Goal: detect “errors” (e.g., flipped bits) in transmitted segment

Sender

- Treat segment contents as sequence of 16-bit integers
- Checksum: addition (1’s complement sum) of segment contents
- Sender puts checksum value into checksum field in header

Receiver

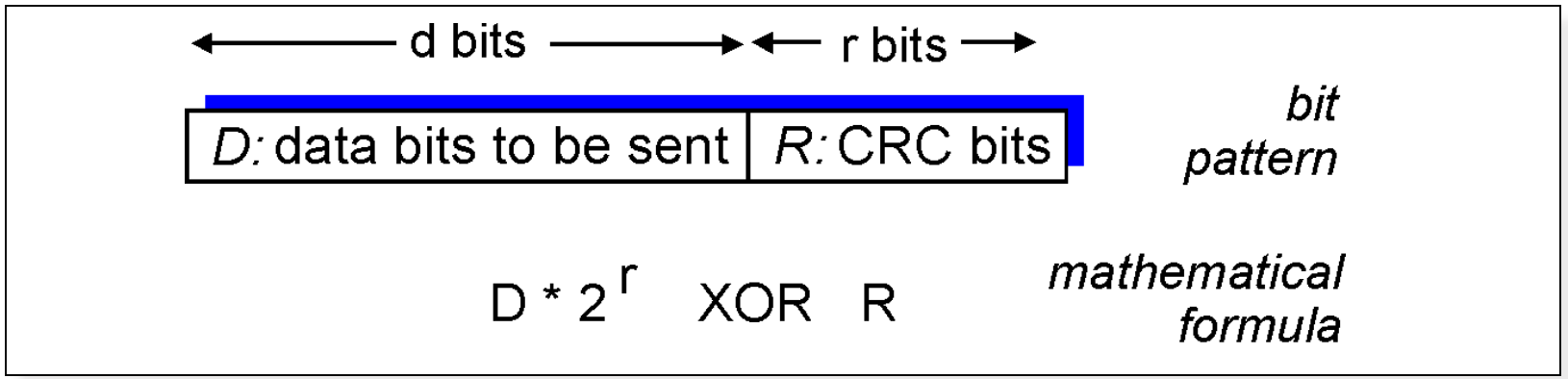
- Compute checksum of received segment
- Check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. But maybe errors nonethless?

Error Detection – Cyclic Redundancy Check (CRC)

- Better loss detection properties than checksums
  - Cyclic codes have favorable properties in that they are well suited for detecting burst errors
  - Therefore, used on networks/hard drives
- Polynomial code
  - Treat packet bits a coefficients of n-bit polynomial
  - Choose r+1 bit generator polynomial (well known – chosen in advance)
  - Add r bits to packet such that message is divisible by generator polynomial

Error Detection – CRC

- View data bits, D, as a binary number
- Choose r+1 bit pattern (generator), G
- Goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - Receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
  - Can detect all burst errors less than r+1 bits
- Widely used in practice: Ethernet, disks



CRC Example

Want:

$D \cdot 2^r$ XOR R = nG

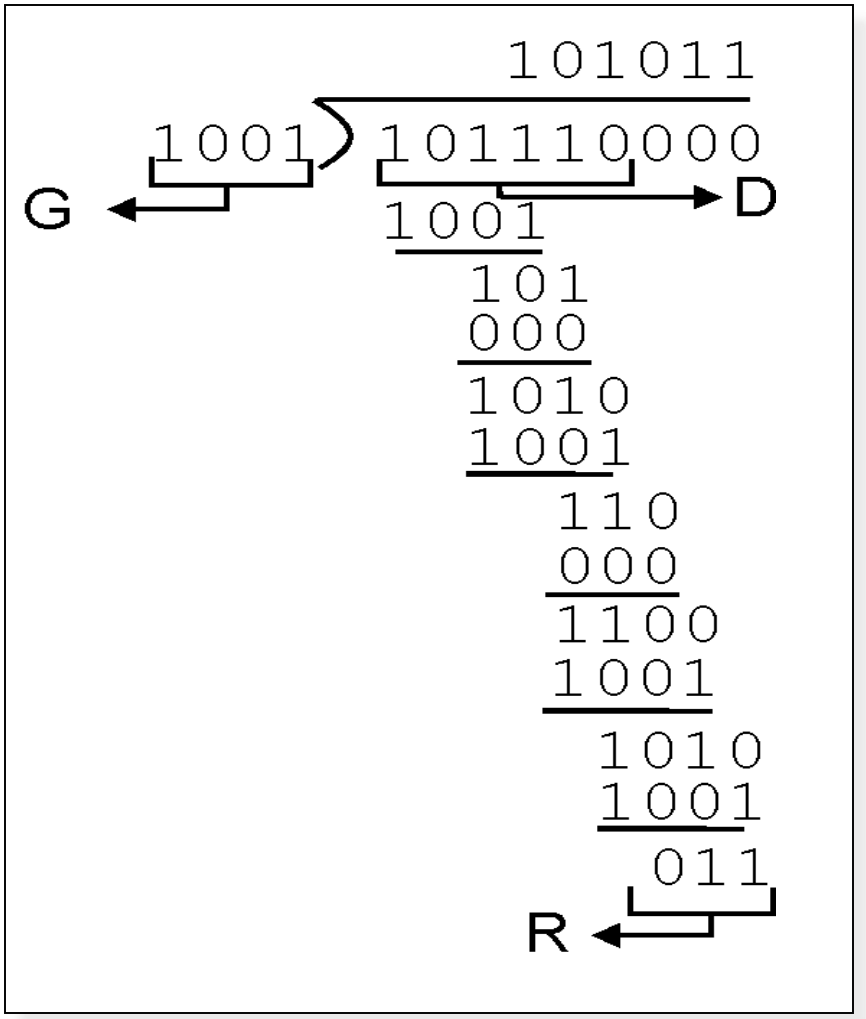
equivalently:

$D \cdot 2^r$ = nG XOR R

equivalently:

if we divide $D \cdot 2^r$ by G, want reminder R

$$R = \text{remainder}\left[\frac{D \cdot 2^r}{G}\right]$$



101110011 divisible by 1001

<D,R> G

What are our options?

1. Silently return the wrong answer.

2. Detect failure.

3. Correct / mask the failure

Error Recovery

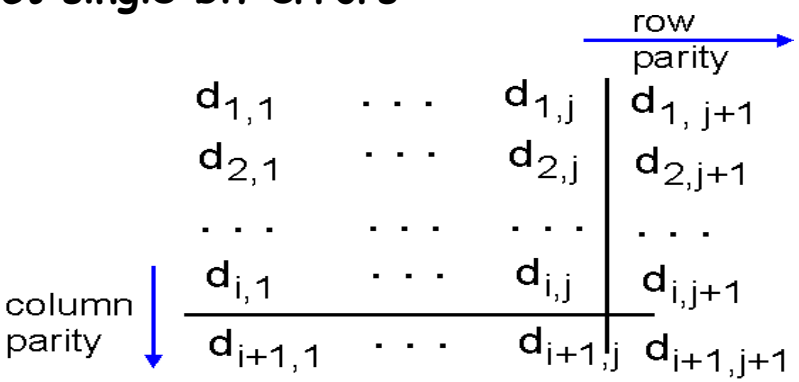
- Two forms of error recovery
  - Redundancy
    - Error Correcting Codes (ECC)
    - Replication/Voting
  - Retry

- ECC
  - Keep encoded redundant data to help repair losses
  - Forward Error Correction (FEC) – send bits in advance
    - Reduces latency of recovery at the cost of bandwidth

Error Recovery – Error Correcting Codes (ECC)

Two Dimensional Bit Parity:

Detect *and correct* single bit errors



10101	1
11110	0
01110	1
00101	0

No errors

10101	1
10110	0
01110	1
00101	0

Single bit error

10101	1
10110	1
01110	1
01101	0

Computed parity bits

Replication/Voting

- If you take this to the extreme

  [r1] [r2] [r3]

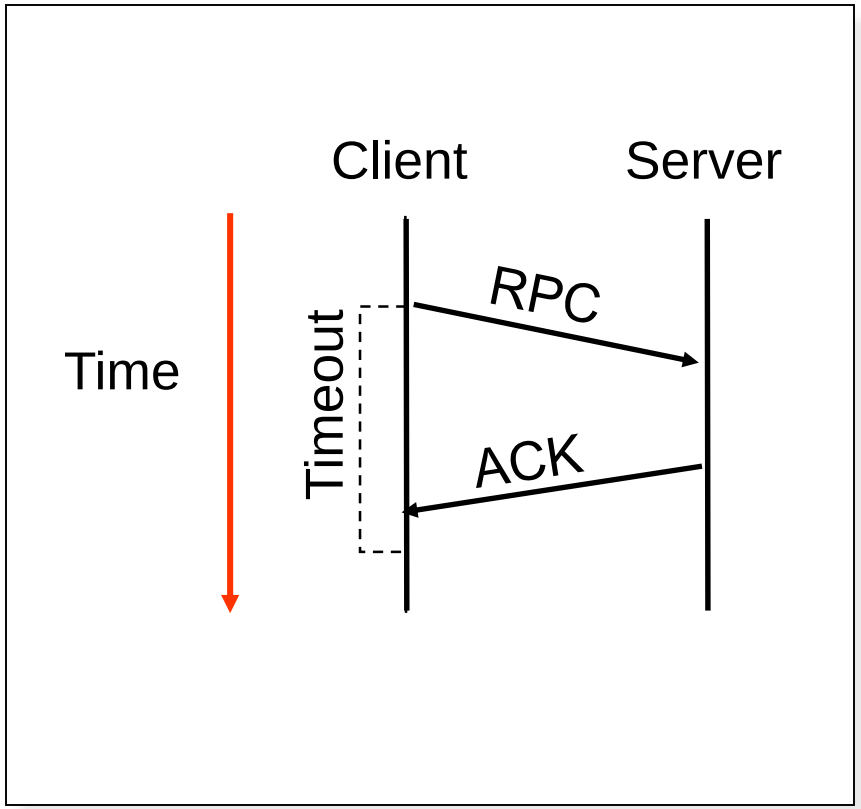
- Send requests to all three versions of the software: Triple modular redundancy
  - Compare the answers, take the majority
  - Assumes no error detection

- In practice - used mostly in space applications; some extreme high availability apps (stocks & banking? maybe. But usually there are cheaper alternatives if you don't need real-time)
  - Stuff we cover later: surviving malicious failures through voting (byzantine fault tolerance)

28

Retry – Network Example

- Sometimes errors are transient
- Need to have error detection mechanism
  - E.g., timeout, parity, chksum
  - No need for majority vote



One key question

- How correlated are failures?
- Can you assume independence?
  - If the failure probability of a computer in a rack is p,
  - What is p(computer 2 failing) | computer 1 failed?
    - Maybe it's p... or maybe they're both plugged into the same UPS...
- Why is this important?
  - Correlation reduces value of redundancy

Fault Tolerant Design

- Quantify probability of failure of each component
- Quantify the costs of the failure
- Quantify the costs of implementing fault tolerance

- This is all probabilities...

Outline

- Errors/error recovery
- **RAID levels and performance**
- Estimating availability

Back to Disks…

- Real HDDs
  - A sequence of sectors (blocks)
    - Normally 512 B or 4KB

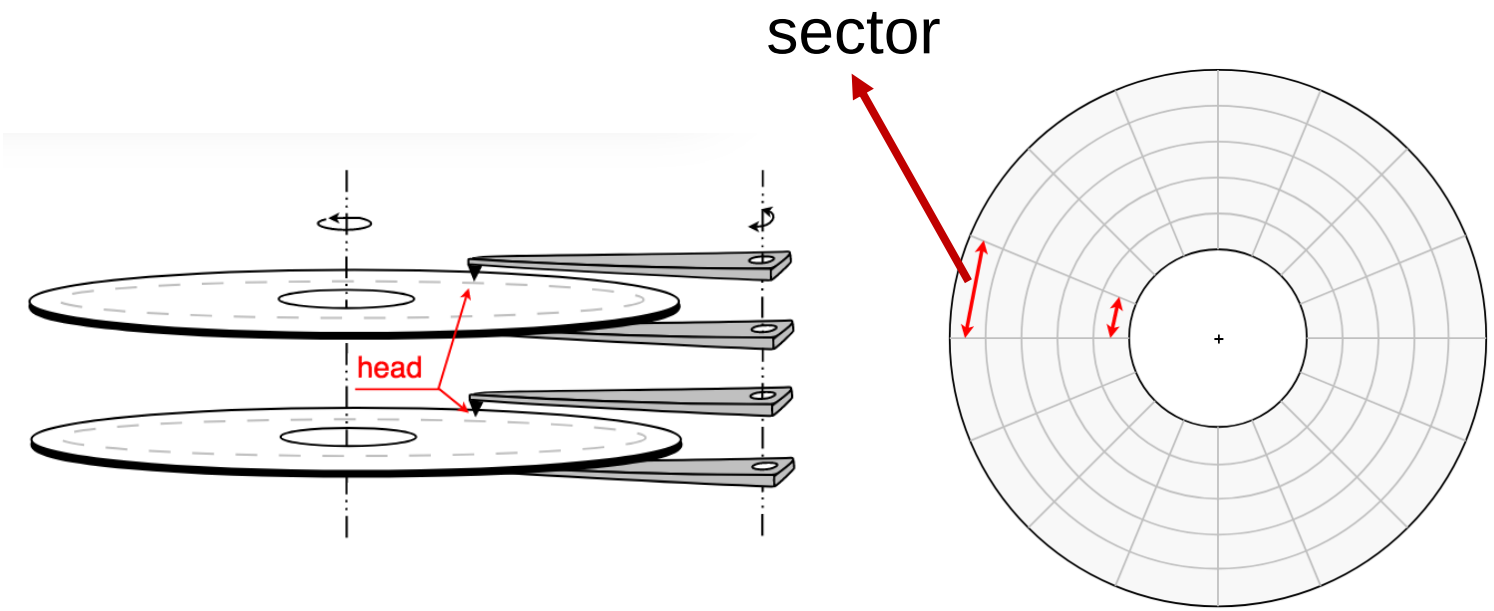


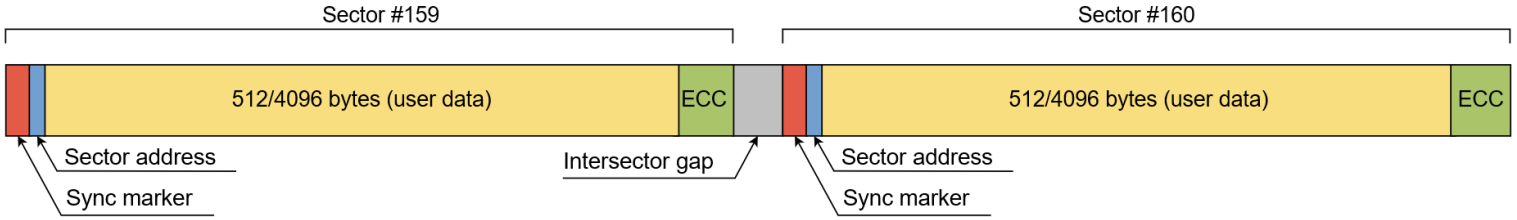
*Image source: Storage subsystem performance: analysis and recipes http://gudok.xyz/sspar/

Back to Disks…
What are our options?

1. Silently return the wrong answer.
2. Detect Failure
   - Put CRC in header/trailer of each physical block
   - If CRC mismatches, return error
3. Correct / Mask Failure
   - Re-read if the firmware signals error (may help if transient error, may not)
   - Use an error correcting code (what kinds of errors do they help with?)
     - Can handle bit flips? Damaged blocks?

Back to Disks…

- Correcting/Masking Ex: Real HDDs
  - Every sector has an ECC after data section. Every read fetches both, computes the ECC on the data, and compares it to the version in the section. If mismatch, returns recoverable soft error.



Sector layout

*Image source: Storage subsystem performance: analysis and recipes http://gudok.xyz/sspar/

Back to Disks…
What are our options?

1. Silently return the wrong answer.
2. Detect Failure
   - Put CRC in header/trailer of each physical block
   - If CRC mismatches, return error
3. Correct / Mask Failure
   - Re-read if the firmware signals error (may help if transient error, may not)
   - Use an error correcting code (what kinds of errors do they help with?)
     - Can handle bit flips? Damaged blocks?
   - Have the data stored in multiple places (RAID)

RAID Taxonomy

- Redundant Array of Inexpensive Independent Disks
  - Constructed by UC-Berkeley researchers in late 80s
  - Exposed to users as a single logical disk
  - Actually an array of multiple physical disks

- Standard Levels
  - **RAID 0** – Coarse-grained Striping with no redundancy
  - **RAID 1** – Mirroring of independent disks
  - RAID 2 – Fine-grained data striping plus Hamming code disks
  - RAID 3 – Fine-grained data striping plus parity disk
  - **RAID 4** – Coarse-grained data striping plus parity disk
  - **RAID 5** – Coarse-grained data striping plus striped parity
  - RAID 6 – Extends RAID 5 by adding another parity block

RAID

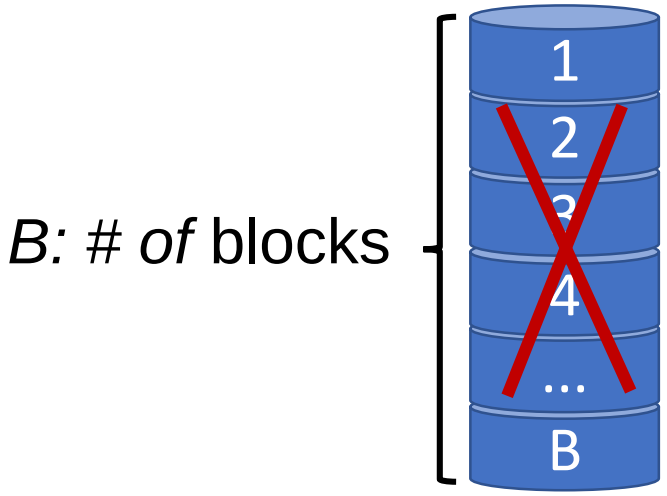
- Definitions
  - Reliability : # of disk failures we can tolerate
  - Latency : Time to process Read/Write in RAID
- Will only consider Random R/W in this class
  - We are reading any block from disks (does not have to be sequential)

RAID Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
Single Disk						
RAID-0						
RAID-1						
RAID-4						
RAID-5						

Let's get started!

Single Disk

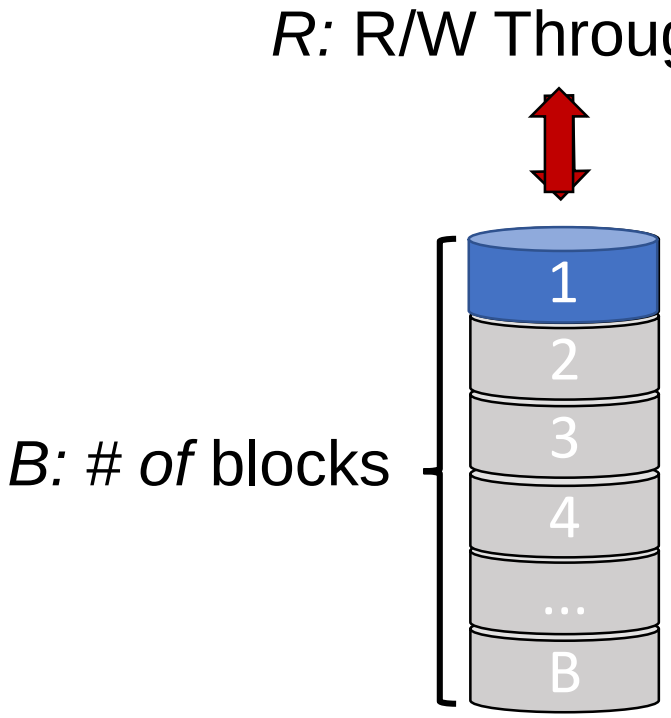
B: # of blocks per disk
R: R/W throughput of a disk
N: # of disks ***D***: time to R/W block



Level	Capacity	Reliability	Write Throughput	Read Throughput	Write Latency	Read Latency
Single Disk	*B*	*0*				

Single Disk

B: # of blocks per disk
R: R/W throughput of a disk
N: # of disks ***D***: time to R/W block



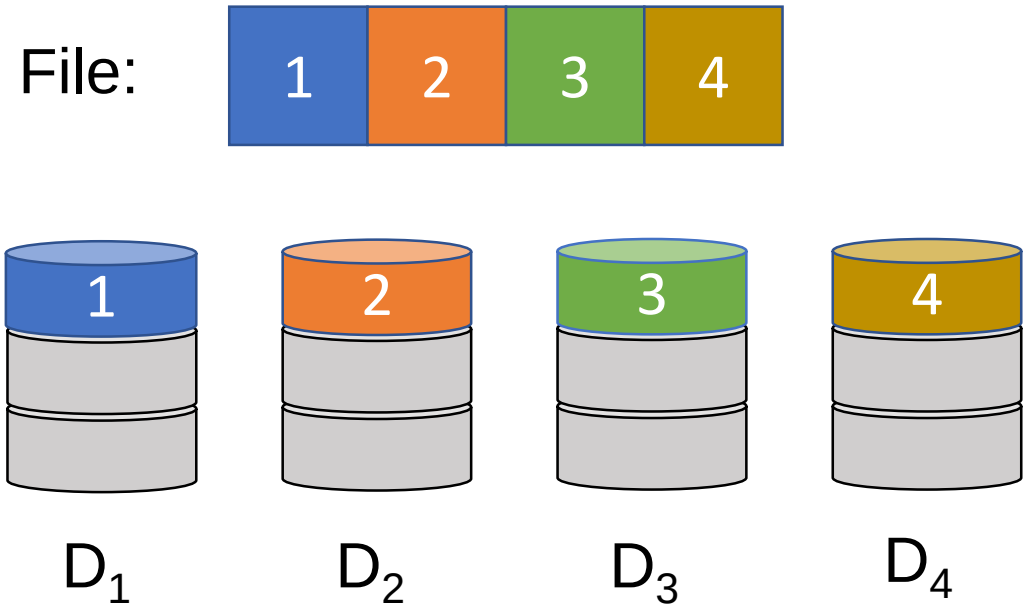
Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
Single Disk	*B*	*0*	*R*	*D*	*R*	*D*

Use multiple disks?

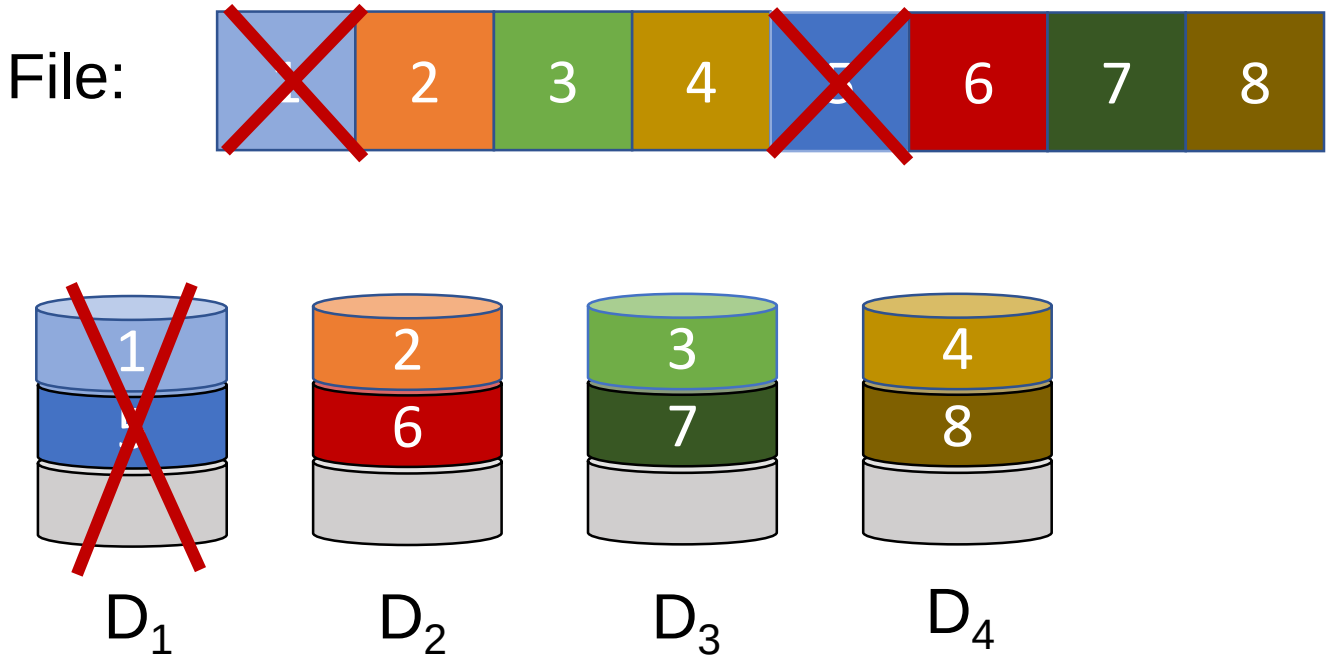
- Capacity
  - More disks allows us to store more data
- Performance
  - Access multiple disks in parallel
  - Each disk can be working on independent read or write
  - Overlap seek and rotational positioning time for all
- Reliability
  - Recover from disk (or single sector) failures
  - Will need to store multiple copies of data to recover

RAID-0: Striping

- To optimize Performance
- Interleave data across multiple disks
  - Large file streaming can enjoy parallel transfers
  - Small requests benefit from load balancing
    - If blocks of hot files equally likely on all disks (really?)



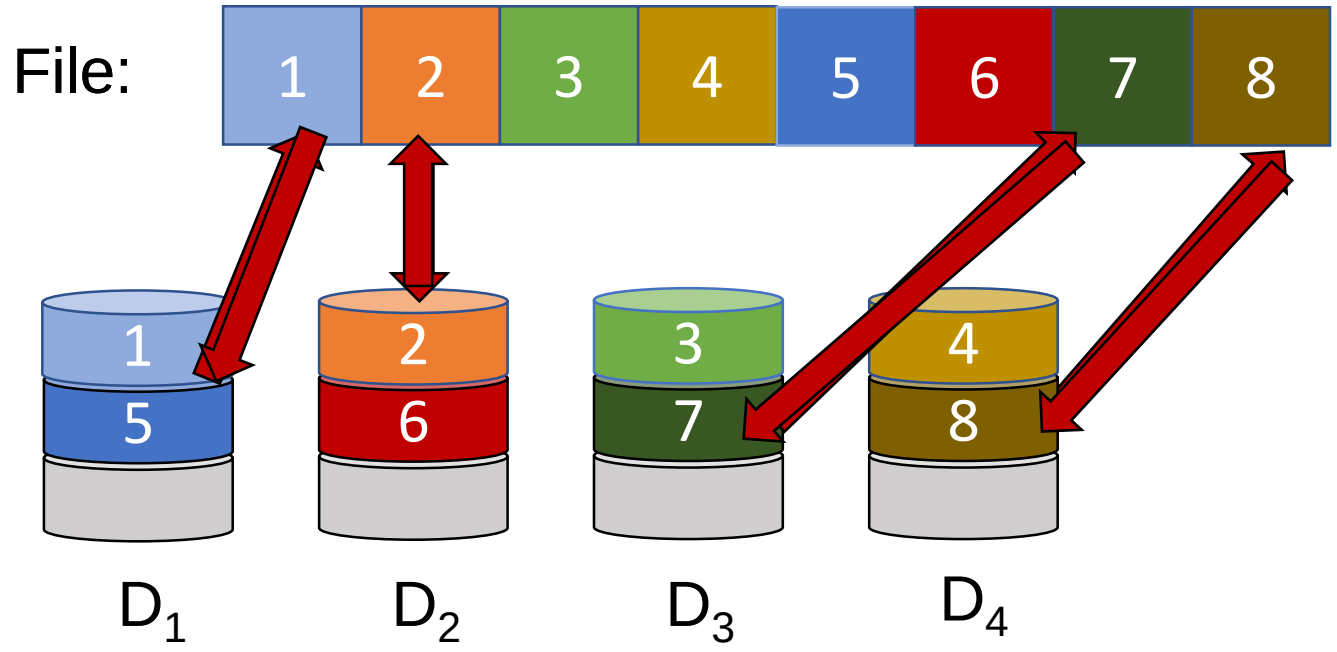
RAID-0: Striping



Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID-0	$N \cdot B$	0				

RAID-0: Striping

B: # of blocks per disk
R: R/W throughput of a disk
N: # of disks ***D***: time to R/W block



Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID-0	$N \cdot B$	0	$N \cdot R$	D	$N \cdot R$	D

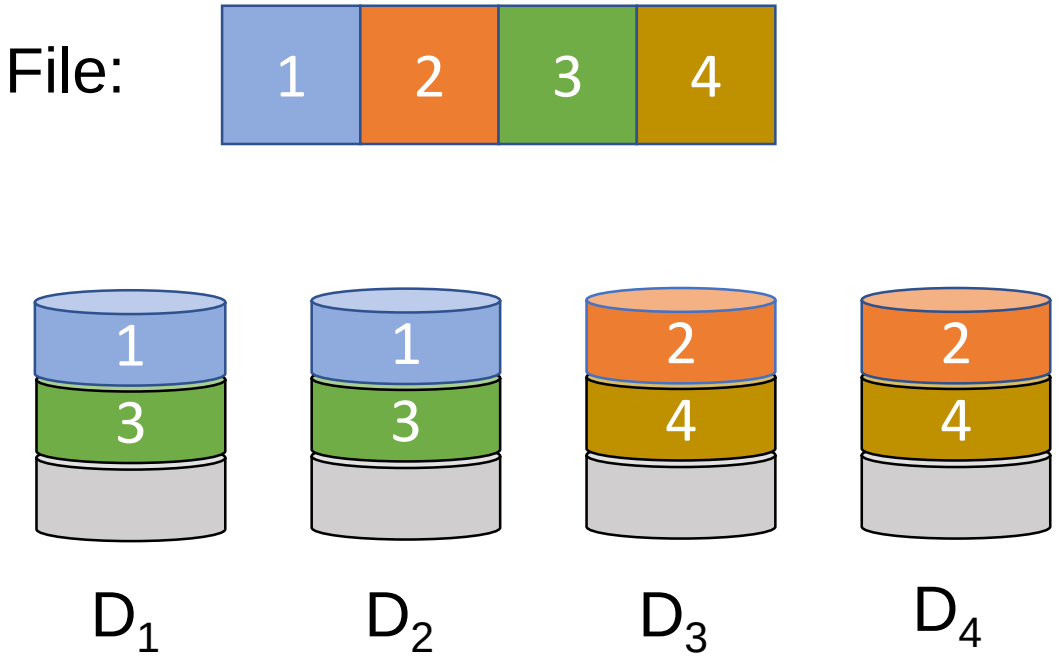
Now, What If A Disk Fails?

- In a striped system
  - a part of each file system lost

- Periodic Backups?
  - backing up takes time and effort
  - backup doesn't help recover data lost during that day
    - Any data loss is a big deal to a bank or stock exchange

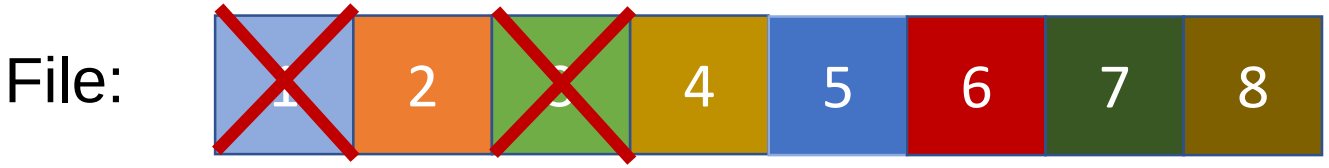
RAID-1: Mirroring

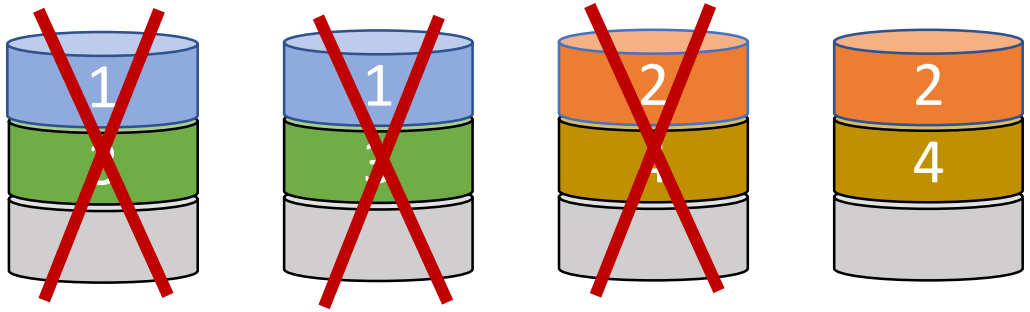
- To achieve better reliability
- Two (or more) copies
  - mirroring, shadowing, duplexing, etc.
- Write both, read either



RAID-1: Mirroring

B: # of blocks per disk
R: R/W throughput of a disk
N: # of disks ***D***: time to R/W block

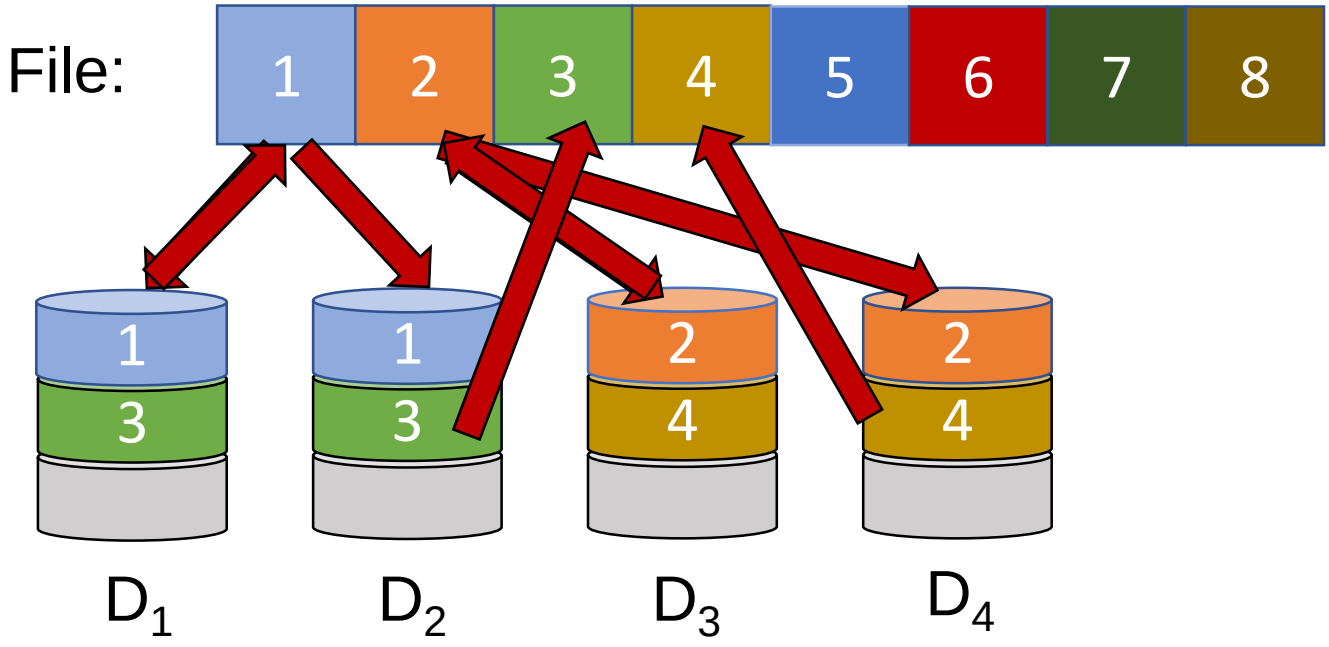




Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID-1	$\frac{N}{2} \cdot B$	1	$N/2$ *(if lucky)*			

RAID-1: Mirroring

B: # of blocks per disk
R: R/W throughput of a disk
N: # of disks ***D***: time to R/W block



Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID-1	$\frac{N}{2} \cdot B$	1 $N/2$ *(if lucky)*	$\frac{N}{2} \cdot R$	D	$N \cdot R$	D

RAID: So-far

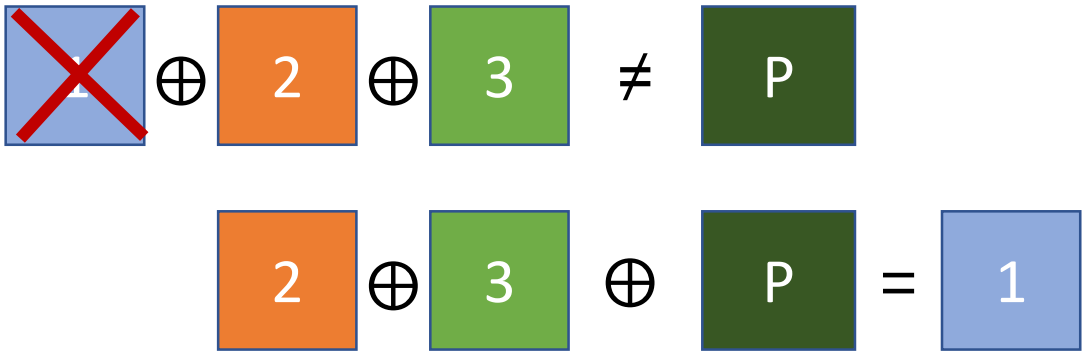
**** Latency Omitted***

Level	Scheme	Capacity	Reliability	Read Throughput	Write Throughput
Single Disk		B	0	R	R
RAID-0	Striping	$N \cdot B$	0	$N \cdot R$	$N \cdot R$
RAID-1	Mirroring	$\frac{N}{2} \cdot B$	1 (for sure) $\frac{N}{2}$ (if lucky)	$N \cdot R$	$\frac{N}{2} \cdot R$
RAID-4					
RAID-5					

- Is Mirroring the best approach for reliability?
  - Parity: RAID-4/5

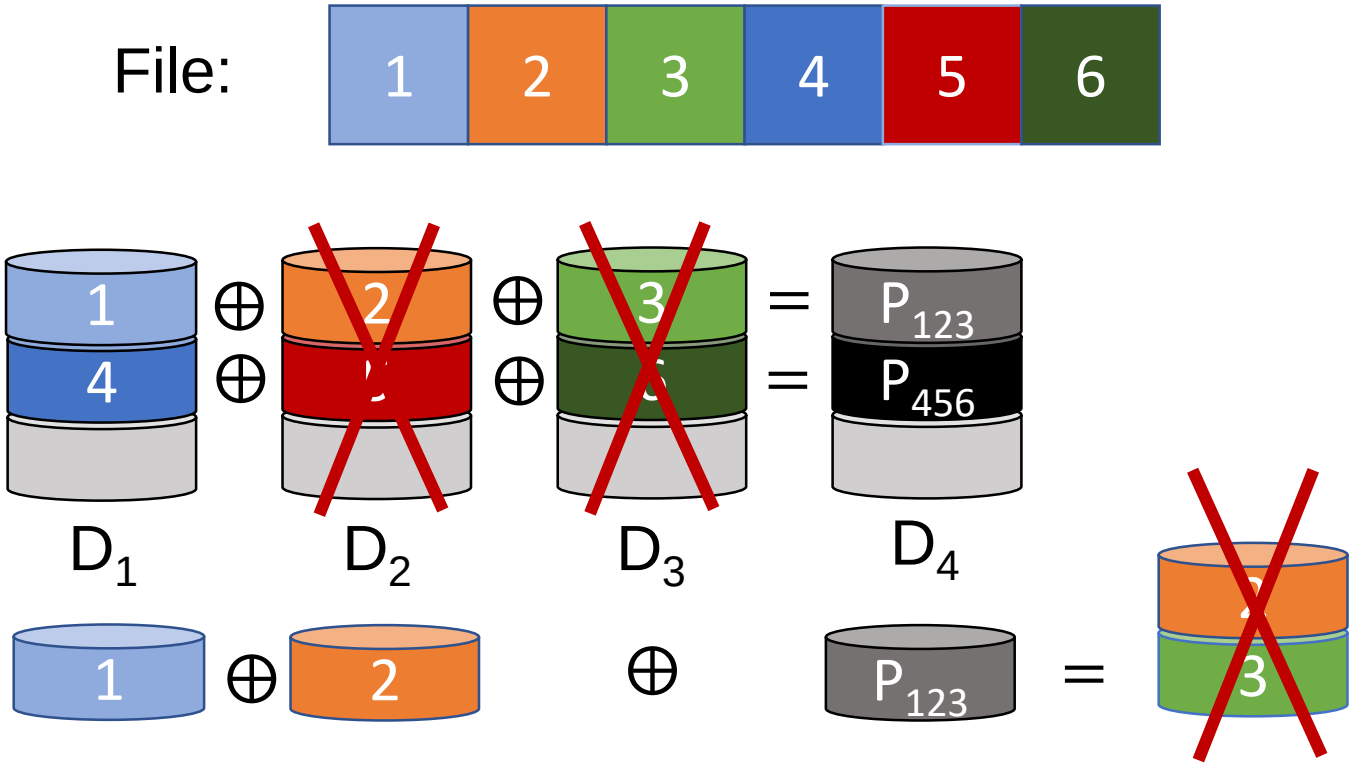
RAID-4: Parity Disk

- Disk failures are self-identifying (a.k.a. erasures)
  - Don't have to find the error
- Erasure code: ECCs under the assumption of bit erasures
  - XOR is one common example
- N-Error detecting code is also N-Erasure Correcting
  - Error-detecting codes can't find an error, just know its there
  - But if you independently know where error is, allows repair



RAID-4: Parity Disk

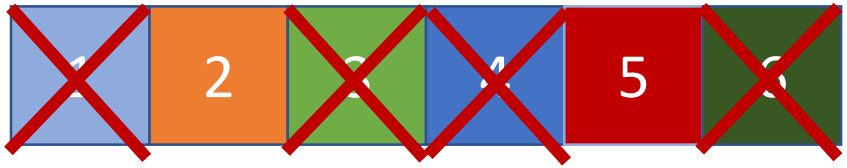
- Capacity: one extra disk needed per stripe

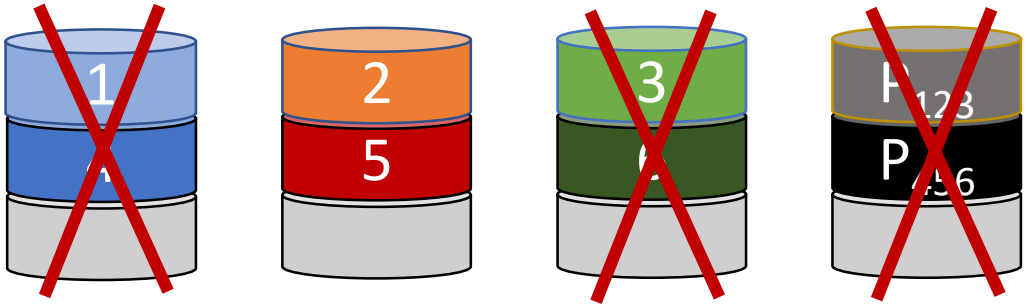


RAID-4: Parity Disk

B: # of blocks per disk
R: R/W throughput of a disk
N: # of disks ***D***: time to R/W block

File:



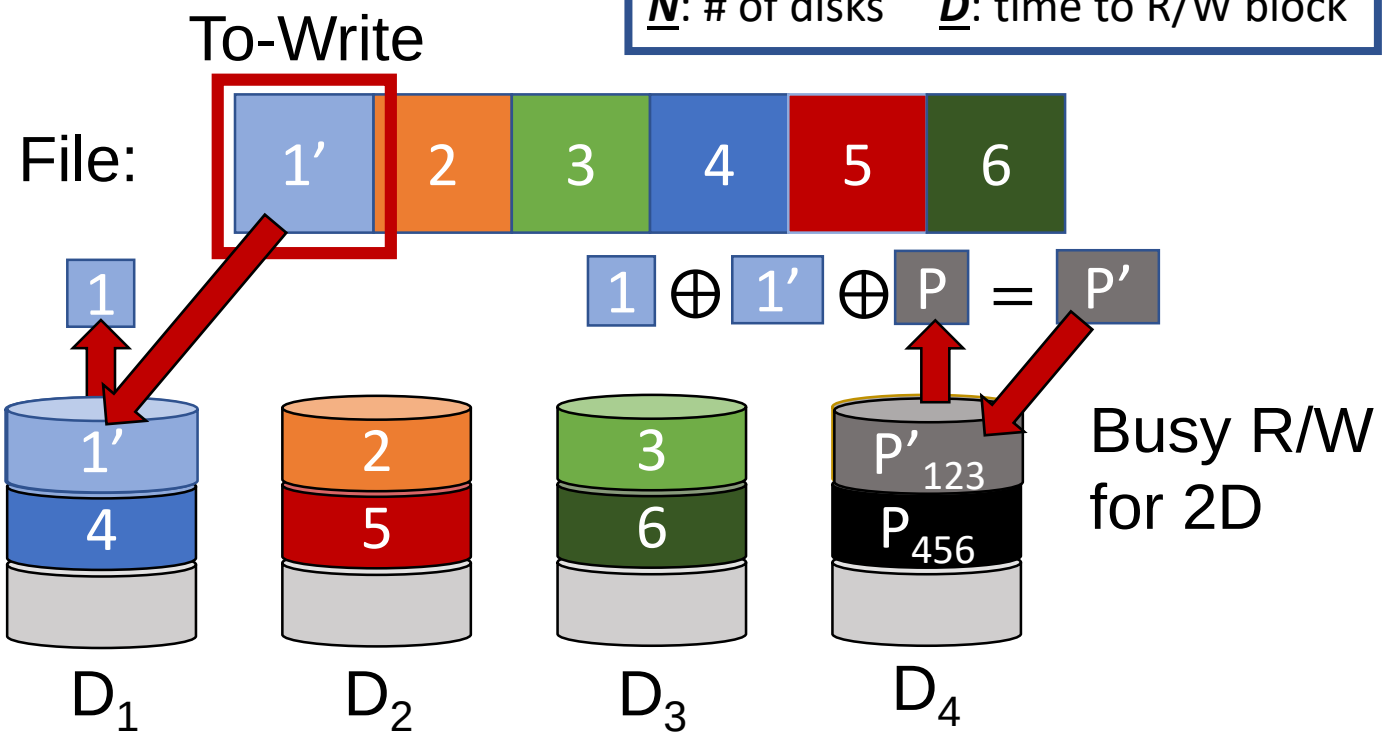


D_1 D_2 D_3 D_4

Level	Capacity	Reliability	Write Throughput	Read Throughput	Write Latency	Read Latency
RAID-4	$(N-1)B$	1				

RAID-4: Parity Disk

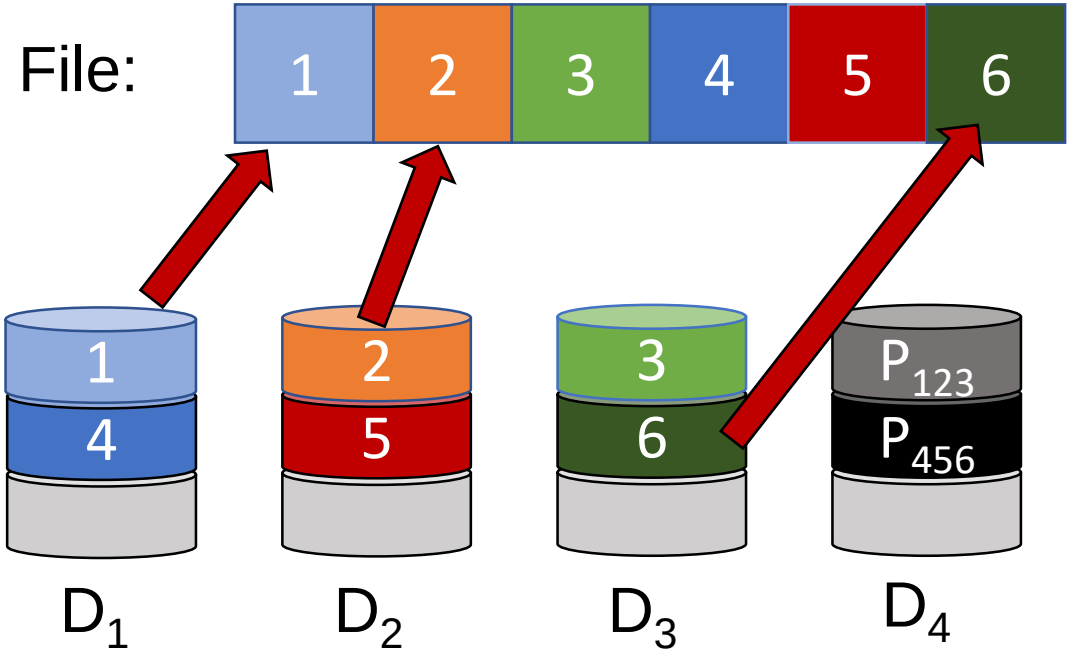
B: # of blocks per disk
R: R/W throughput of a disk
N: # of disks ***D***: time to R/W block



Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID-4	$(N-1)B$	1	$\frac{R}{2}$	$2D$		

RAID-4: Parity Disk

B: # of blocks per disk
R: R/W throughput of a disk
N: # of disks ***D***: time to R/W block



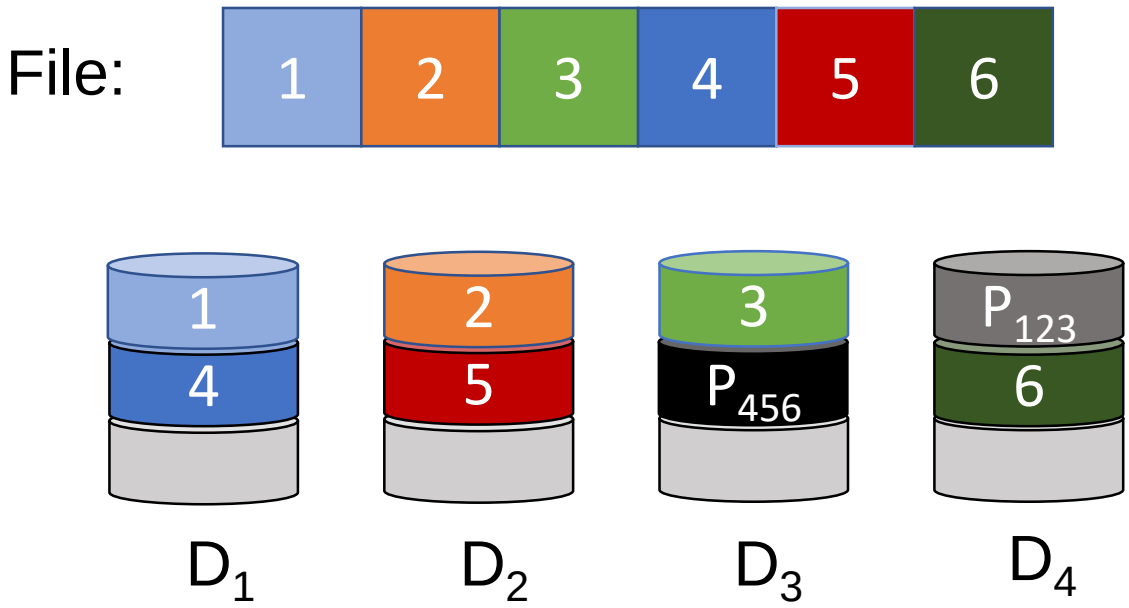
Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID-4	$(N-1)B$	1	$\frac{R}{2}$	$2D$	$(N-1)R$	D

The parity disk bottleneck

- Reads go only to the data disks
  - But, hopefully load balanced across the disks
- All writes go to the parity disk
  - And, worse, usually result in Read-Modify-Write sequence
  - So, parity disk can easily be a bottleneck
  - Parity disk can wear out very fast!
- Adding disk does not provide any performance gain

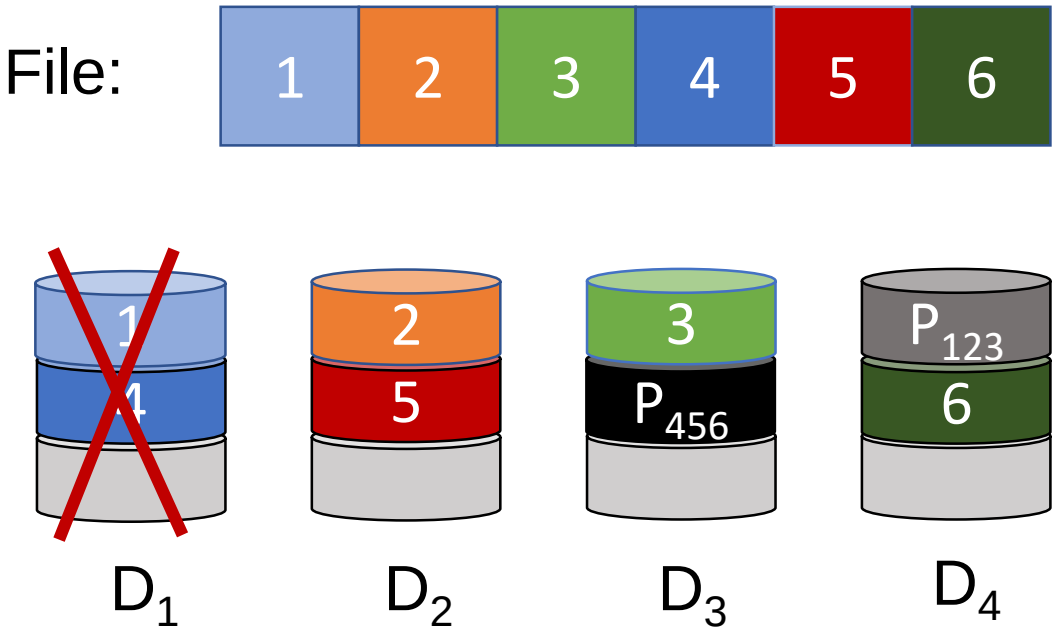
RAID-5: Rotating Parity

- To distribute parity writes, place parity in round-robin manner



RAID-5: Rotating Parity

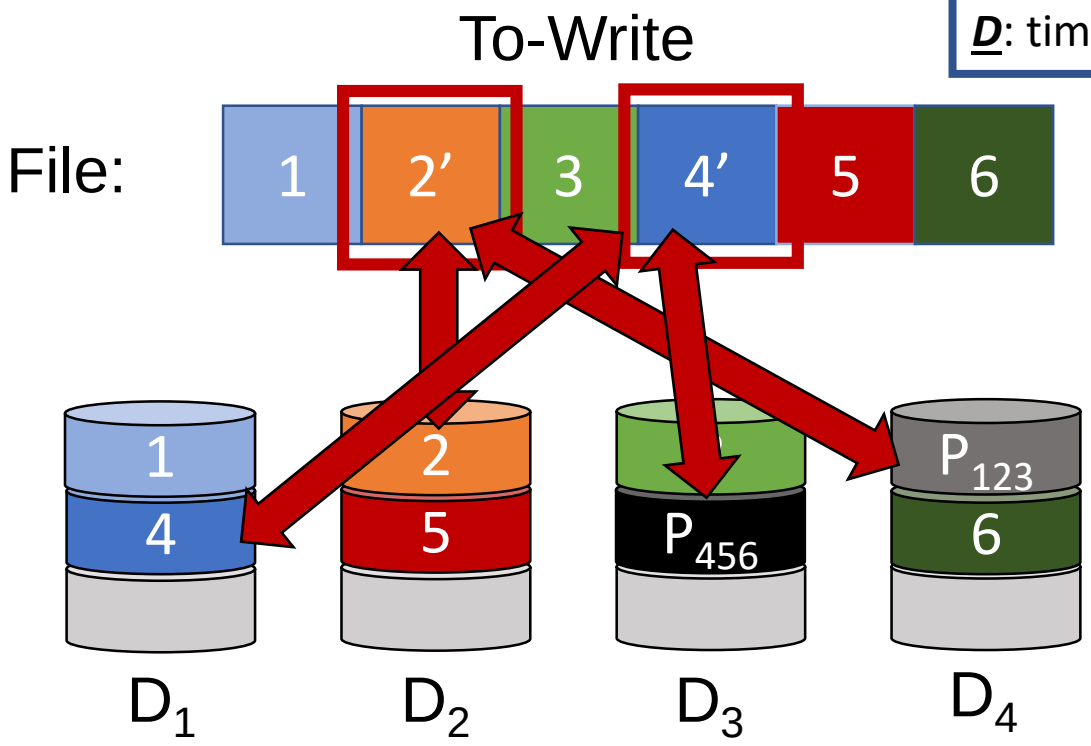
B: # of blocks per disk
R: R/W throughput of a disk
N: # of disks
D: time to R/W block



Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID-5	$(N-1)B$	1				

RAID-5: Rotating Parity

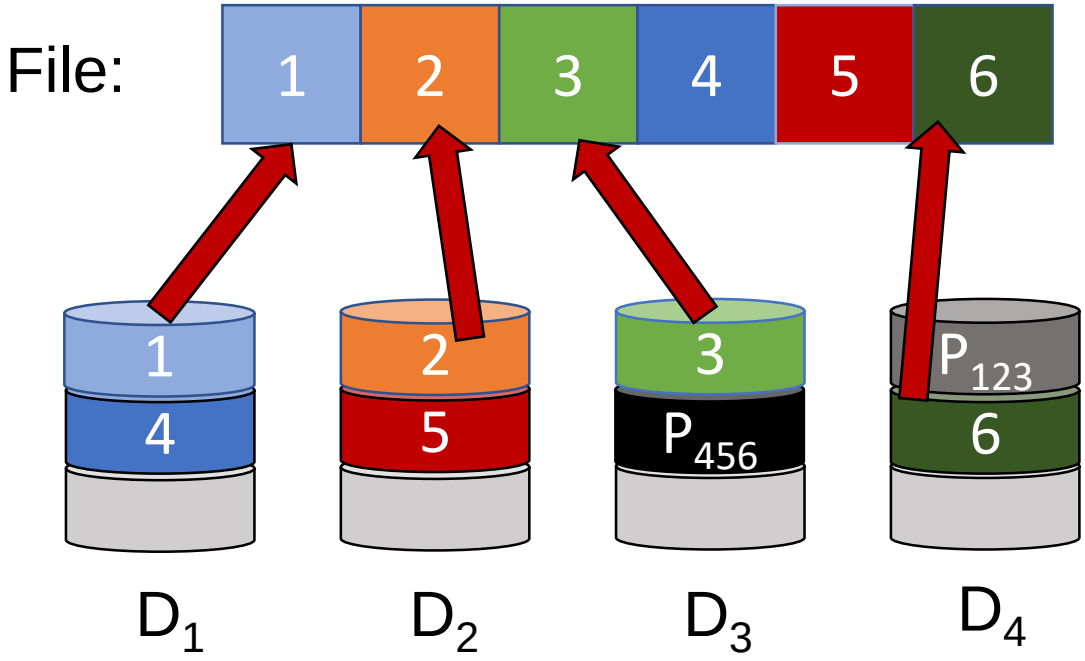
B: # of blocks per disk
R: R/W throughput of a disk
N: # of disks
D: time to R/W block



Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID-5	$(N-1)B$	1	$\frac{N}{4} \cdot R$	$2D$		

RAID-5: Rotating Parity

B: # of blocks per disk
R: R/W throughput of a disk
N: # of disks
D: time to R/W block



Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID-5	$(N-1)B$	1	$\frac{N}{4} \cdot R$	$2D$	$N \cdot R$	D

Recap: RAID

**** Latency Omitted***

Level	Scheme	Capacity	Reliability	Read Throughput	Write Throughput
Single Disk		B	0	R	R
RAID-0	Striping	$N \cdot B$	0	$N \cdot R$	$N \cdot R$
RAID-1	Mirroring	$\frac{N}{2} \cdot B$	1 (for sure) $\frac{N}{2}$ (if lucky)	$N \cdot R$	$\frac{N}{2} \cdot R$
RAID-4	Parity Disk	$(N-1)B$	1	$(N-1)R$	$\frac{R}{2}$
RAID-5	Rotating Parity	$(N-1)B$	1	$N \cdot R$	$\frac{N}{4} \cdot R$

We only considered Random Read/Write Throughput.
For Sequential Read/Write (reading 1, 2, 3, … order), refer to the reading material

Outline

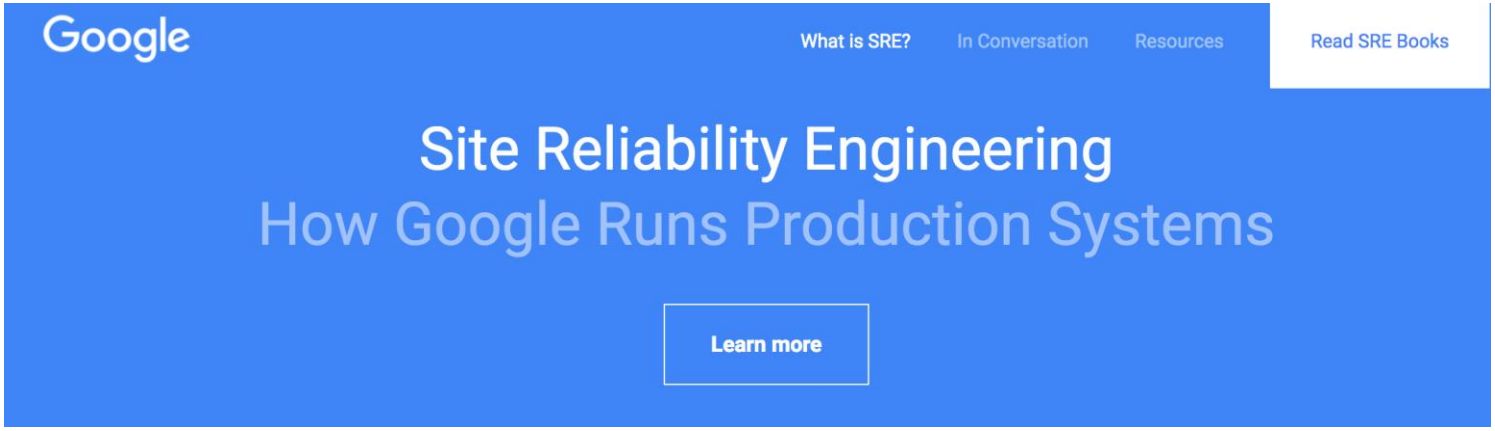
- Errors/error recovery
- RAID levels and performance
- **Estimating availability**

Availability / Reliability

- We will try to calculate availability/reliability using example of disks & RAID

- Why matters?
  - All major companies have reliability team
    - Ex. Google (https://landing.google.com/sre/)
  - If you are interested, there is a career path for reliability engineer!

Availability / Reliability Metrics



What is SRE?

SRE is what you get when you treat operations as if it's a software problem. Our mission is to protect, provide for, and progress the software and systems behind all of Google's public services — Google Search, Ads, Gmail, Android, YouTube, and App Engine, to name just a few — with an ever-watchful eye on their availability, latency, performance, and capacity.

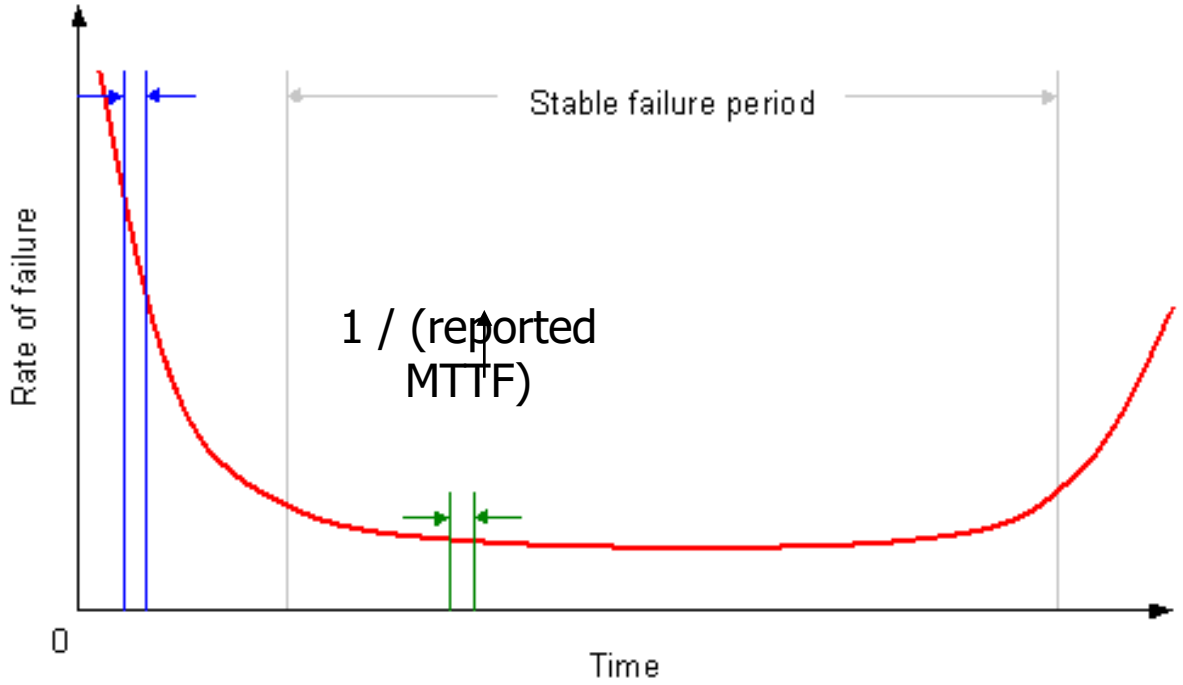
Availability / Reliability Metrics

- Using Disk/RAID…

- We want to calculate MTTDL
  - Mean Time To First Data Loss (MTTDL)
  - Given that we know the MTTF
  - Under simplifying assumptions

Estimating Availability / Reliability

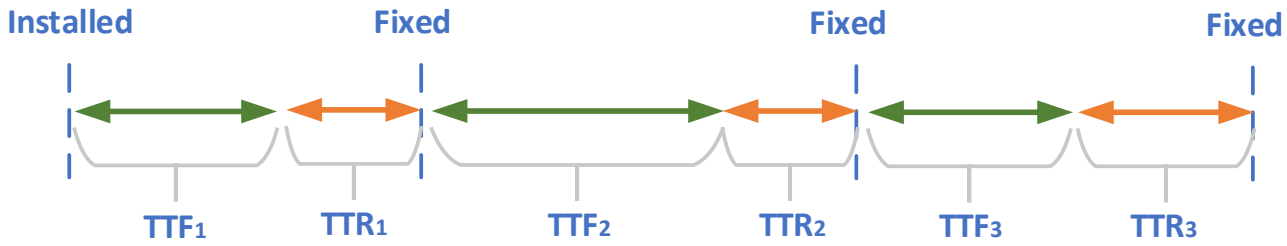
- Back to Disks….
- Back-of-the-envelope calculation
  - Assuming failures across devices are independent
  - Assuming failures over time are independent
  - Assuming failure rate from stable failure period



Recap: MTTF, MTTR

- MTTF: Mean Time to Failure
  - Inverse of prob of failure

- MTTR: Mean Time to Repair



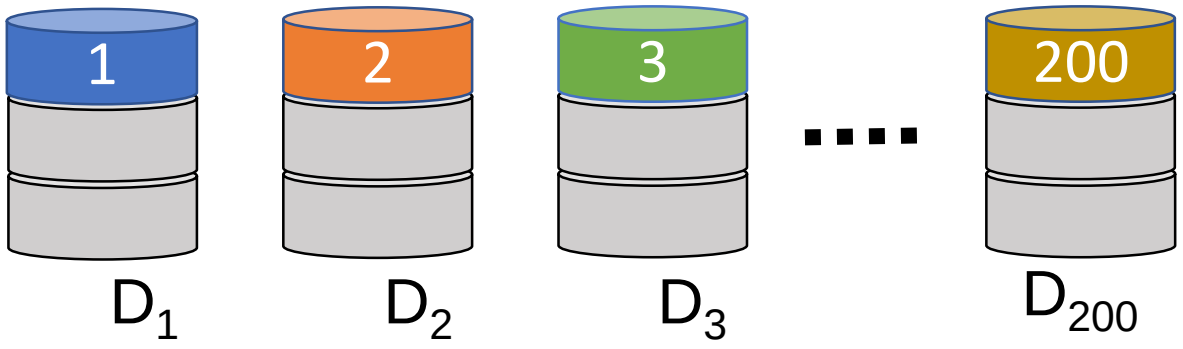
How often are failures

- MTTF of a disk(Mean Time to Failure)
  - $MTTF_{disk}$ ~ 1,200,000 hours (~136 years, <1% per year)

- Recap: MTTF is inverse of failure rate
- With 2 disks,
  - Twice more likely to fail
  - Mean time to first disk failure ~ $MTTF_{disk}$ / 2
- So, we approximate
  - With n disks, n times more likely to fail, and
  - Mean time to first disk failure ~ $MTTF_{disk}$ / (# of disks)

Reliability without Rebuild

- If disk fails, just leave it failed
  - We do not try to rebuild it
- Let's say we want to keep data whose size equals to the capacity of 200 drives
- $MTTDL_{RAID\text{-}0}$ = mean time to first disk failure
  - Can tolerate only 0 disk failure
  - Ex. For a striped array of 200 drives
    - $MTTDL_{RAID\text{-}0}$ = 136 years / 200 drives = 0.65 years



Reliability without Rebuild

- To keep data whose size equals to the capacity of 200 drives
- Add 200 more disks to build RAID-1 (Mirroring)
- total 400 disk needed

- $MTTDL_{RAID\text{-}1}$
  - $MTTDL_{pair} = \underline{(MTTF_{drive} / 2)} + \underline{MTTF_{drive}} = 1.5 * MTTF_{drive}$

    Mean time to first disk failure

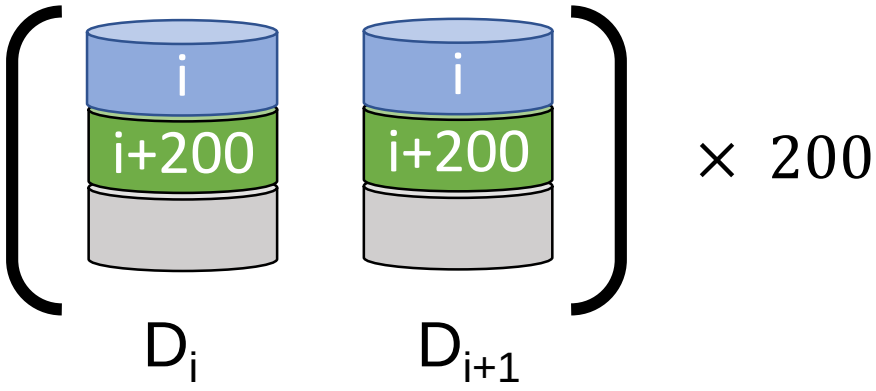
    Mean time to second disk failure

  - $MTTDL_{RAID\text{-}1} = MTTDL_{pair}$ / (# of pairs)
    - RAID-1 fails if at least one pair fail

Reliability without Rebuild

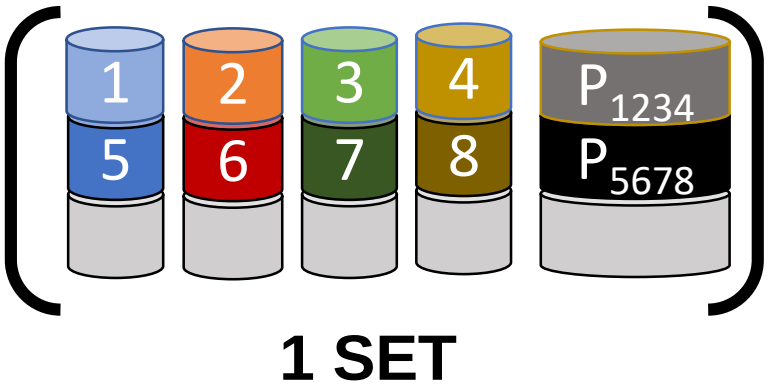
- Add 200 more disks to build RAID-1 (Mirroring)
- total 400 disk needed

- $MTTDL_{pair} = (MTTF_{drive} / 2) + MTTF_{drive} = 1.5 * MTTF_{drive}$
- $MTTDL_{RAID\text{-}1} = MTTDL_{pair}$ / (# of pairs)
- For a RAID-1 of 400 drives (200 mirrored pairs)
- $MTTDL_{RAID\text{-}1}$ = 1.5 * 136 years / 200 pairs = 1.02 years



Reliability without Rebuild

- To keep data whose size equals to the capacity of 200 drives
- Add only 50 more disks to build RAID-4
- total 250 disks
- 1 parity disk per 4 data disk



- $MTTDL_{RAID-4}$
  - 1 Set : 4 data disk + 1 parity disk
  - $MTTDL_{set} = \underline{MTTF_{drive} / 5} + \underline{MTTF_{drive} / 4}$

    Mean time to first disk failure

    Mean time to second disk failure
  - $MTTDL_{RAID-4} = MTTF_{set}$ / (# of sets)

Reliability without Rebuild

- To keep data whose size equals to the capacity of 200 drives
- Add only 50 more disks to build RAID-4
- total 250 disks
- 1 parity disk per 4 data disk

- $MTTDL_{RAID\text{-}4}$
  - $MTTDL_{set} = (MTTF_{drive} / 5) + MTTF_{drive} / 4 = \mathbf{0.45 * MTTF_{drive}}$
  - $MTTDL_{RAID\text{-}4} = MTTDL_{set}$ / (# of sets)

  - For a RAID-4 of 250 drives (50 sets)
  - $MTTDL_{RAID\text{-}4}$ = 0.45* 136 years / 50 sets = 1.22 years

Reliability without Rebuild

- Comparisons
  - To keep data whose size is equal to capacity of 200 drives
  - MTTF: the longer the better!
  - RAID 0: Striping
    - With total 200 drives, MTTDL = 0.65 years
  - RAID 1: Mirroring
    - With total 400 drives, MTTDL = 1.02 years
  - RAID 4: Parity Disk
    - With total 250 drives, MTTDL = 1.22 years

Rebuild: restoring redundancy after failure (extra)

- After a drive failure
  - data is still available for access
  - but, a second failure is BAD

- So, should reconstruct the data onto a new drive
  - online spares are common features of high-end disk arrays
    - reduce time to start rebuild
  - must balance rebuild rate with foreground performance impact
    - a performance vs. reliability trade-offs

- How data is reconstructed
  - Mirroring: just read good copy
  - Parity: read all remaining drives (including parity) and compute

Reliability consequences of adding rebuild (extra)

- No data loss, if fast enough
  - That is, if first failure fixed before second one happens
- Now MTTR is considered
- New math is...
  - $MTTDL_{array}$ = ***1/ prob of 1st failure*** * **1/ prob of 2nd failure before repair**)

1/ prob of 1st failure = ***Mean time to first disk failure***

... where ***prob of 2nd failure before repair*** is **MTTRdrive / MTTFseconddrive**

Reliability consequences of adding rebuild (extra)

- For mirroring
  - $MTTDL_{pair} = (MTTF_{drive} / 2) * (MTTF_{drive} / MTTR_{drive})$

    Mean time to first disk failure

    Inverse of prob. of second disk failure before repair
  - $MTTDL_{RAID\text{-}1} = MTTDL_{pair}$ / (# of pairs)

- For 5-disk parity-protected arrays
  - $MTTDL_{set} = (MTTF_{drive} / 5) * ((MTTF_{drive} / 4) / MTTR_{drive})$

    Mean time to first disk failure

    Inverse of prob. of second disk failure before repair
  - $MTTDL_{RAID\text{-}4} = MTTDL_{set}$ / (# of sets)

Three modes of operation

- Normal mode
  - everything working; maximum efficiency
- Degraded mode
  - some disk unavailable
  - must use degraded mode operations
- Rebuild mode
  - reconstructing lost disk's contents onto spare
  - degraded mode operations plus competition with rebuild

Mechanics of rebuild

- Background process
  - use degraded mode read to reconstruct data
  - then, write it to replacement disk

- Implementation issues
  - Interference with foreground activity and controlling rate
    - Rebuild is important for reliability
    - Foreground activity is important for performance
  - Using the rebuilt disk
    - For rebuilt part, reads can use replacement disk
    - Must balance performance benefit with rebuild interference

Summary

- Definition of MTTF/MTBF/MTTR: Understanding availability in systems.
- Failure detection and fault masking techniques
- Engineering tradeoff: Cost of failures vs. cost of failure masking.
  - At what level of system to mask failures?
  - Leading into replication as a general strategy for fault tolerance
- Thought to leave you with:
  - What if you have to survive the failure of entire computers? Of a rack? Of a datacenter?

79

Summary

- RAID turns multiple disks into a larger, faster, more reliable disk
- **RAID-0**: Striping
  Good when performance and capacity really matter, but reliability doesn't
- **RAID-1**: Mirroring
  Good when reliability and write performance matter, but capacity (cost) doesn't
- **RAID-5**: Rotating Parity
  Good when capacity and cost matter or workload is read-mostly
  - Good compromise choice