Fault Tolerance & RAID

Announcements

- Project 2 Released
  - Recitation this Wednesday
- Midterm-1 Recap. (Come and talk to the instructors, as per Piazza)
- P3 – Group Project, DO NOT have to have the same partner

Outline

- Errors (focus on errors in DRAM and disk drives)
  - Presence, impact, availability metric
- Tools/options for dealing with errors
  - Retries, checksums, CRC, error detection/correction codes, …
- RAID levels and performance
  - For disk systems
- Estimating availability

Types of Errors

Hard errors: Dead/damaged component which experiences a fail-stop/crash.

Soft errors: A flipped signal or bit, caused by an external source or a faulty component.

A faulty component can cause recurring soft errors.

Example: DRAM

Hard error

- Faulty traces/bad solder on the motherboard
- Defective DRAM bank

Soft error

- Bit flips caused by cosmic radiation or alpha particles (from the chip itself) hitting memory cell, changing values
- DRAM is just little capacitors to store charge…
  if you hit it with radiation, you can add charge to it.

DRAM errors in real life

Both Microsoft® and Google have recently started to identify DRAM errors as an increasing contributor to system failures

Google – Datacenter servers

Microsoft® – Consumer desktop/laptop

DRAM Errors in the Wild: A Large-Scale Field Study [Schroeder '09]

Disk failures in the real world [Schroeder '07]

HPC 1

Component	%
Hard drive	**30.6**
Memory	28.5
Misc/Unk	14.4
CPU	12.4
PCI motherboard	4.9
Controller	2.9
QSW	1.7
Power supply	1.6
MLB	1.0
SCSI BP	0.3

Company 1

Component	%
Power supply	34.8
Memory	20.1
Hard drive	**18.1**
Case	11.4
Fan	8.0
CPU	2.0
SCSI Board	0.6
NIC Card	1.2
LV Power Board	0.6
CPU heatsink	0.6

Company 2

Component	%
Hard drive	**49.1**
Motherboard	23.4
Power supply	10.1
RAID card	4.1
Memory	3.4
SCSI cable	2.2
Fan	2.2
CPU	2.2
CD-ROM	0.6
Raid Controller	0.6

Hard drives are a major cause of server failures.

Impact of failures

One small error:

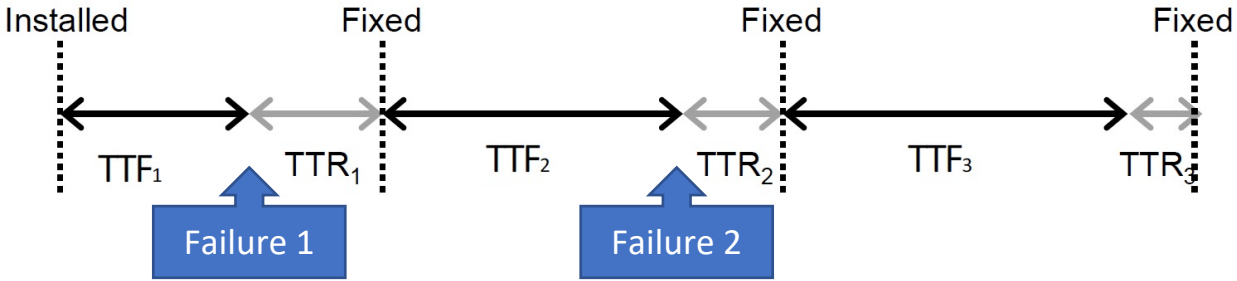
A single bit in DRAM flips due to a cosmic ray

Propagates:

- This bit happens to be mapped in kernel address space
- You soon get a kernel panic
- This node was part of a dozen storage servers for your DFS
- A client hangs trying to read from the DFS
- The Gradescope frontend can't fetch your 15-440 assignment
- You get an F in the course

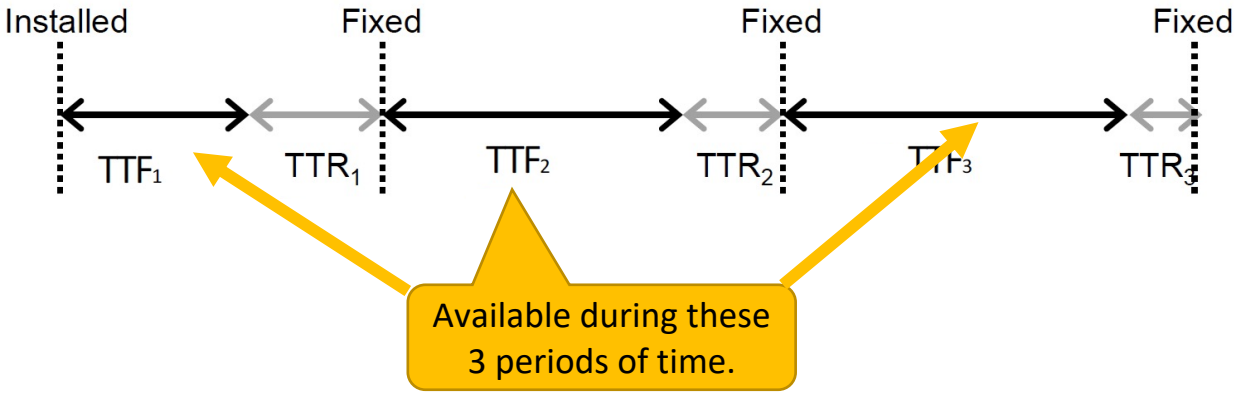
Measuring Availability

- Mean time to **failure** (MTTF)
- Mean time to **repair** (MTTR)
- Mean time between failures (MTBF) = MTTF + MTTR



Measuring Availability

- Mean time to **failure** (MTTF)
- Mean time to **repair** (MTTR)
- Mean time between failures (MTBF) = MTTF + MTTR



Measuring Availability

- Mean time to **failure** (MTTF)
- Mean time to **repair** (MTTR)
- Mean time between failures (MTBF) = MTTF + MTTR

Availability = MTTF / (MTTF + MTTR)

- Example: Your OS crashes once a month, takes 10 minutes to reboot
- MTTF = 720 hours (43,200 minutes)
- MTTR = 10 minutes
- Availability = 43,200 / 43,210 = 0.997 ("2 nines")

Availability --- the “nines”

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%

Availability SLOs and SLAs

- Availability Service Level Objective (**SLO**)
  - An availability threshold which your system targets
- Availability Service Level Agreement (**SLA**)
  - An availability threshold that you guarantee for customers

Let’s look at an example

Example: AWS EC2 Availability SLA

Monthly Uptime Percentage	Service Credit Percentage
Less than 99.99% but equal to or greater than 99.0%	10%
Less than 99.0% but equal to or greater than 95.0%	30%
Less than 95.0%	100%

Real Devices

Specifications

	12TB
Model Number[1]	WD120EFAX
Interface[2]	SATA 6 Gb/s
Formatted capacity[2]	12TB
Form factor	3.5-inch
Native command queuing	Yes
Advanced Format (AF)	Yes
RoHS compliant[3]	Yes
Performance	
Interface Transfer Rate[2] up to	196 MB/s
Cache (MB)[2]	256
Performance Class	5400 RPM Class
Reliability/Data Integrity	
Load/unload cycles[4]	600,000
Non-recoverable errors per bits read	<1 in 10^{14}
MTBF (hours)[5]	1,000,000
Workload Rate (TB/year)[6]	180
Limited warranty (years)[7]	3
Power Management[8]	
12VDC ±5% (A, peak)	1.84
5VDC ±5% (A, peak)	
Average power requirements (W)	
Read/Write	6.3
Idle	2.9
Standby and Sleep	0.6
Environmental Specifications[9]	
Temperature (°C)	
Operating	0 to 65
Non-operating	-40 to 70
Shock (Gs)	
Operating, (2 ms, read/write)	30
Operating, (2 ms, read)	65
Non-operating (2 ms)	300

Real Devices

Specifications

	12TB
Model Number[1]	WD120EFAX
Interface[2]	SATA 6 Gb/s
Formatted capacity[2]	12TB
Form factor	3.5-inch
Native command queuing	Yes
Cache (MB)[2]	256
Performance Class	5400 RPM Class
Reliability/Data Integrity	
Load/unload cycles[4]	600,000
Non-recoverable errors per bits read	<1 in 10^{14}
MTBF (hours)[5]	1,000,000
Workload Rate (TB/year)[6]	180
Limited warranty (years)[7]	3
Power Management[8]	
12VDC ±5% (A, peak)	1.84
5VDC ±5% (A, peak)	
Average power requirements (W)	
Read/Write	6.3
Idle	2.9
Standby and Sleep	0.6
Environmental Specifications[9]	
Temperature (°C)	
Operating	0 to 65
Non-operating	-40 to 70
Shock (Gs)	
Operating, (2 ms, read/write)	30
Operating, (2 ms, read)	65
Non-operating (2 ms)	300

MTBF (hours)[5] 1,000,000

Real Devices

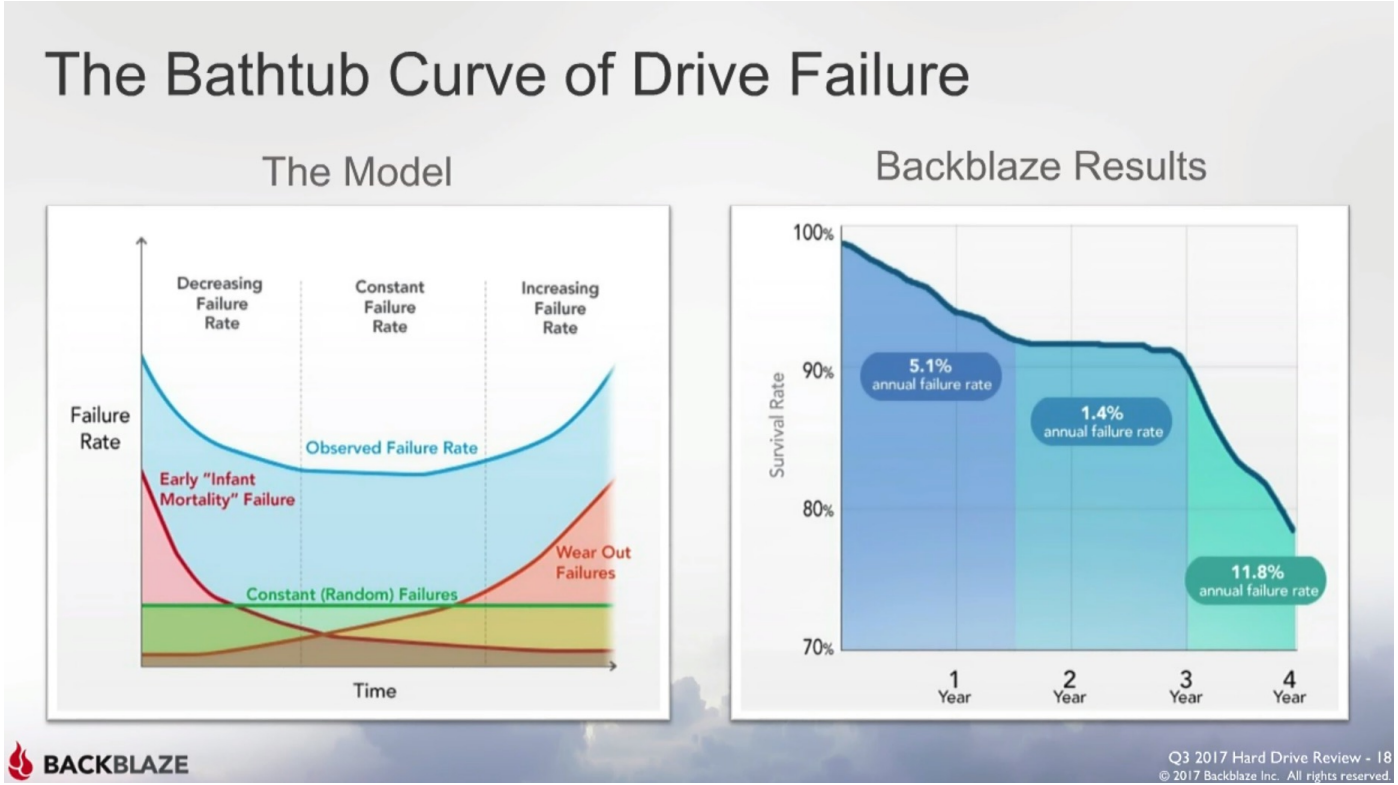
Specifications

	12TB
Model Number[1]	WD120EFAX
Interface[2]	SATA 6 Gb/s
Formatted capacity[2]	12TB
Form factor	3.5-inch
Native command queuing	Yes
Cache (MB)[2]	256
Performance Class	5400 RPM Class
Reliability/Data Integrity	
Load/unload cycles[4]	600,000
Non-recoverable errors per bits read	<1 in 10^{14}
MTBF (hours)[5]	1,000,000
Workload Rate (TB/year)[6]	180
Limited warranty (years)[7]	3
Power Management[8]	
12VDC ±5% (A, peak)	1.84
5VDC ±5% (A, peak)	
Average power requirements (W)	
Read/Write	6.3
Idle	2.9
Standby and Sleep	0.6
Environmental Specifications[9]	
Temperature (°C)	
Operating	0 to 65
Non-operating	-40 to 70
Shock (Gs)	
Operating, (2 ms, read/write)	30
Operating, (2 ms, read)	65
Non-operating (2 ms)	300

MTBF (hours)[5] 1,000,000

MTBF of 114 years!

MBTF of 1,000,000 hours: the fine print



Outline

- Errors (focus on errors in DRAM and disk drives)
  - Presence, impact, availability metric
- Tools/options for dealing with errors
  - Retries, checksums, CRC, error detection/correction codes, …
- RAID levels and performance
  - For disk systems
- Estimating availability

What are our options?

1. Silently return the wrong data.
2. Detect Failure.
3. Correct / mask the failure.

What are our options?

~~1. Silently return the wrong data.~~

2. Detect Failure.

3. Correct / mask the failure.

What are our options?

~~1. Silently return the wrong data.~~

2. Detect Failure.

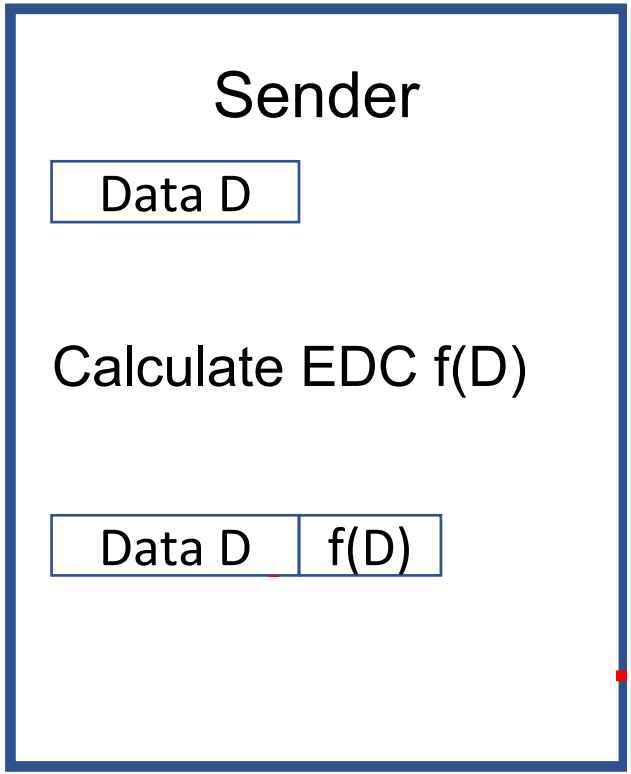
3. Correct / mask the failure.

Detecting Errors

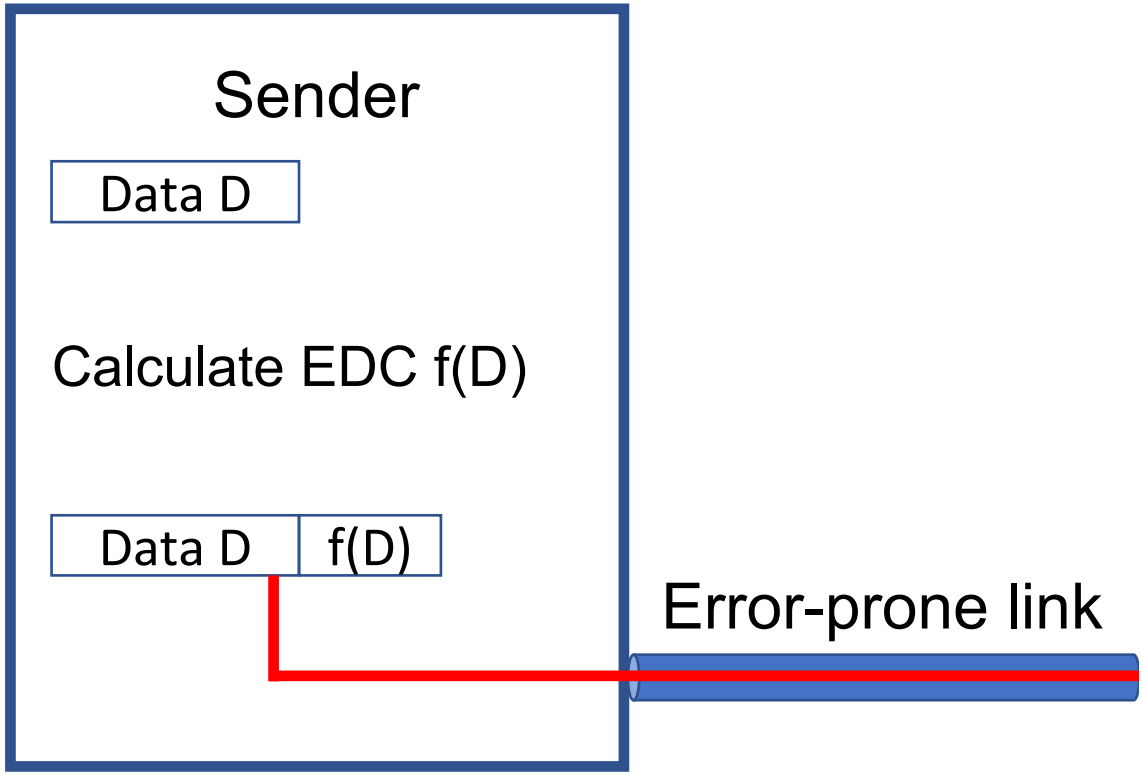
Key idea: information redundancy

- Add additional information (bits) that can be used to verify the correctness

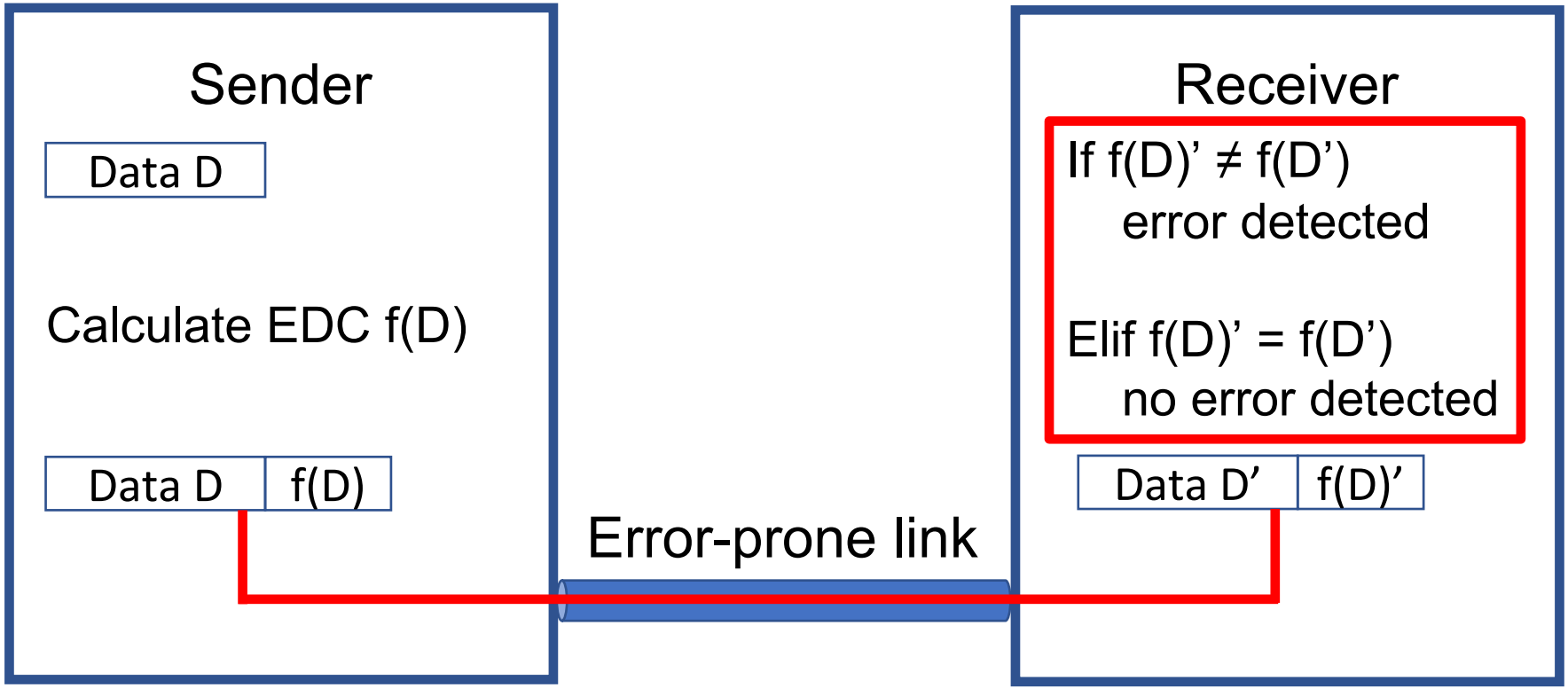
Error Detecting Codes



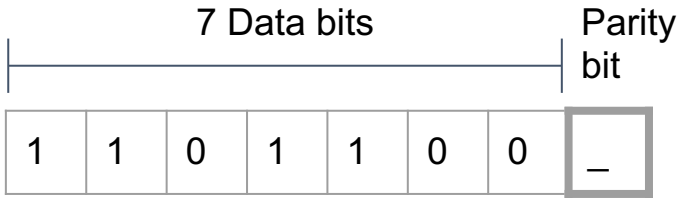
Error Detecting Codes



Error Detecting Codes

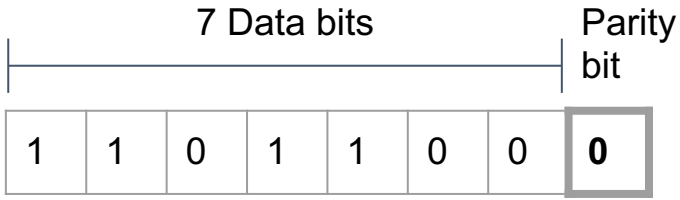


Error Detection: Single Bit Parity



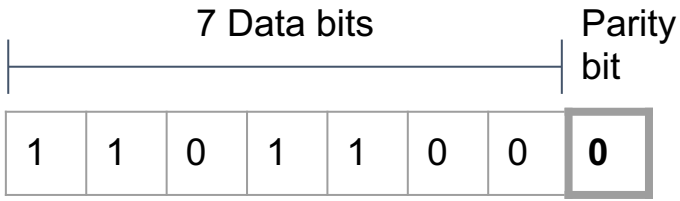
f(D) = sum of bits in D (mod 2)

Error Detection: Single Bit Parity

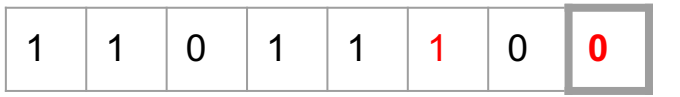


f(D) = sum of bits in D (mod 2)

Error Detection: Single Bit Parity

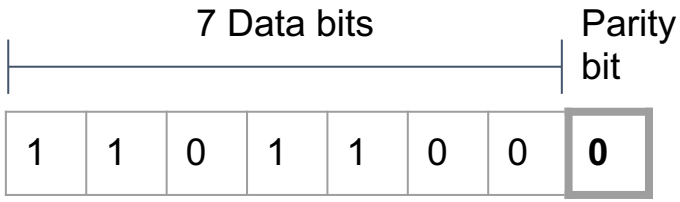


f(D) = sum of bits in D (mod 2)

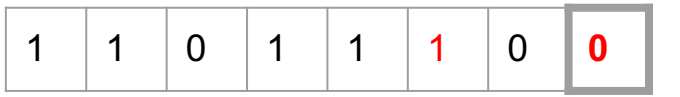


Single Bit Parity can detect a single-bit error.

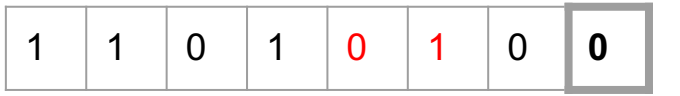
Error Detection: Single Bit Parity



f(D) = sum of bits in D (mod 2)



Single Bit Parity can detect a single-bit error.



Single Bit Parity cannot reliably detect multiple bit errors.

Error Detection - Checksum

- Same principle as single bit parity but at coarser granularity
- Example: IP packets

- f(D) = Ones' complement sum of all bytes in a packet
  - Simple to implement
  - Relatively weak detection
  - Tricked by typical error patterns - e.g. burst errors

Error Detection - Cyclic Redundancy Check (CRC)

- Strong detection capability
- Good at detecting burst errors
- Wide adoption
  - Efficient streaming implementation in hardware
  - x86 instruction to calculate CRC
- Used by ethernet and hard drives

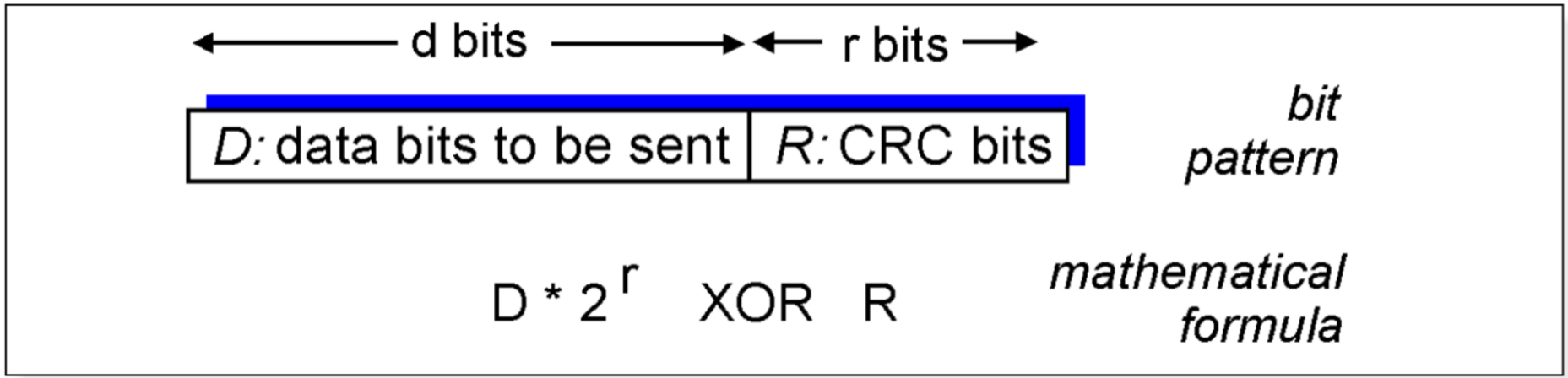
Error Detection - CRC

- Based on “polynomial codes”
  - Treat data bits, D, as polynomial coefficients
  - Choose r+1 bit “generator polynomial”, G
    - Send/receiver share in advance
  - Add r bits to packet as CRC bits, R
  - Packet received <D, R> should be divisible by G
  - Can detect all burst errors less than r+1 bits

  - See the next few slides if you are curious

Error Detection - Cyclic Redundancy Check (CRC)

- Treat the bits of transmitted data as polynomial coefficients
- Sender and receiver agree on a Generator polynomial *G*
- Append *check bits* to the data so it is divisible by *G* (mod 2)



CRC Example

<D, R> is evenly divisible by G modulo 2

$D \cdot 2^r \oplus R \equiv kG \pmod 2$

$D \cdot 2^r \equiv kG \oplus R \pmod 2$ *XOR both sides by R*

$D \cdot 2^r \bmod G \equiv 0 \oplus R \pmod 2$ *mod both sides by G*

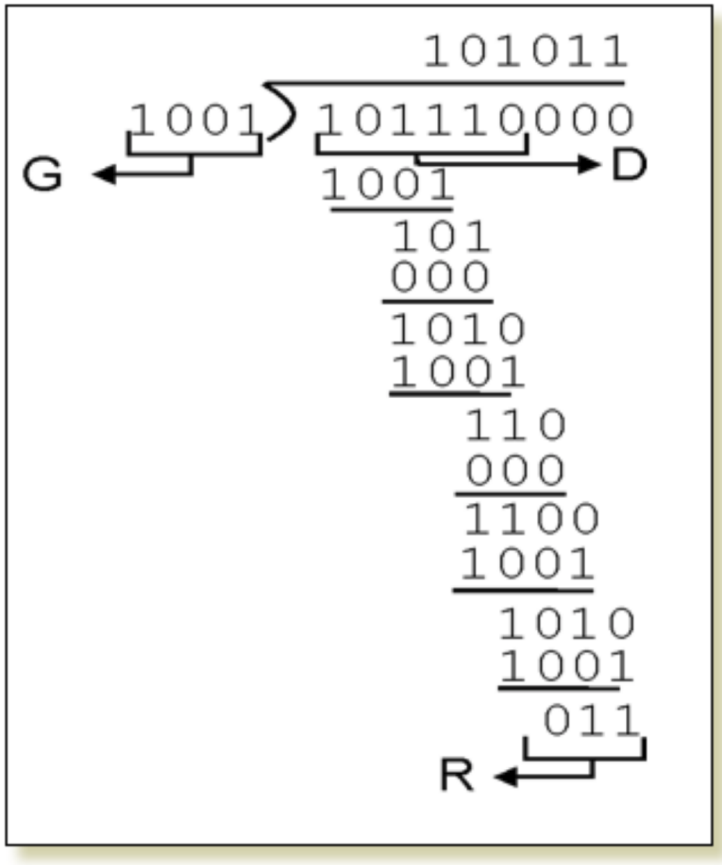
Or equivalently

$(D \cdot 2^r) / G \pmod 2 = R$

CRC Example

D = 101110
G = 1001
r = 3

Find R



What are our options?

~~1. Silently return the wrong answer.~~

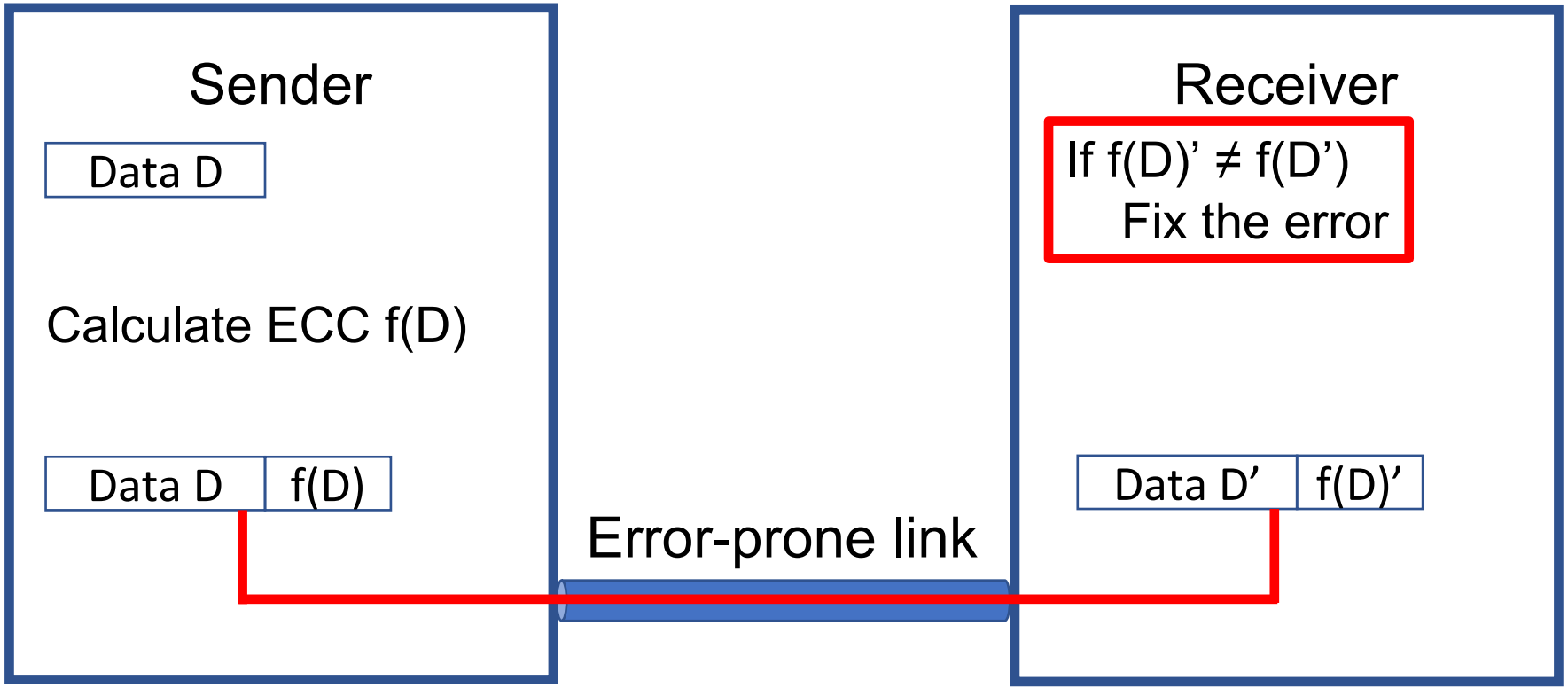
2. Detect Failure.

3. Correct / mask the failure.

Error Recovery

- Two forms of error recovery
  - Redundancy (forward recovery)
    - Error Correcting Codes (ECC)
    - Replication/Voting
  - Retry (backward recovery)
- ECC
  - Use encoded redundant data to help repair data loss
  - Forward Error Correction – send bits in advance
    - Tradeoff?
    - Reduces latency of recovery at the cost of BW

Error Correcting Codes



We will not go into ECC in this course

Courses that cover ECC, if you are interested:

- 15-750 Graduate Algorithms
- 15-853 Algorithms in the real world
- 15-848 Practical information and coding theory for computer systems

- Course material and information available on Rashmi's webpage

Error Recovery - Replication/Voting

Triple modular redundancy

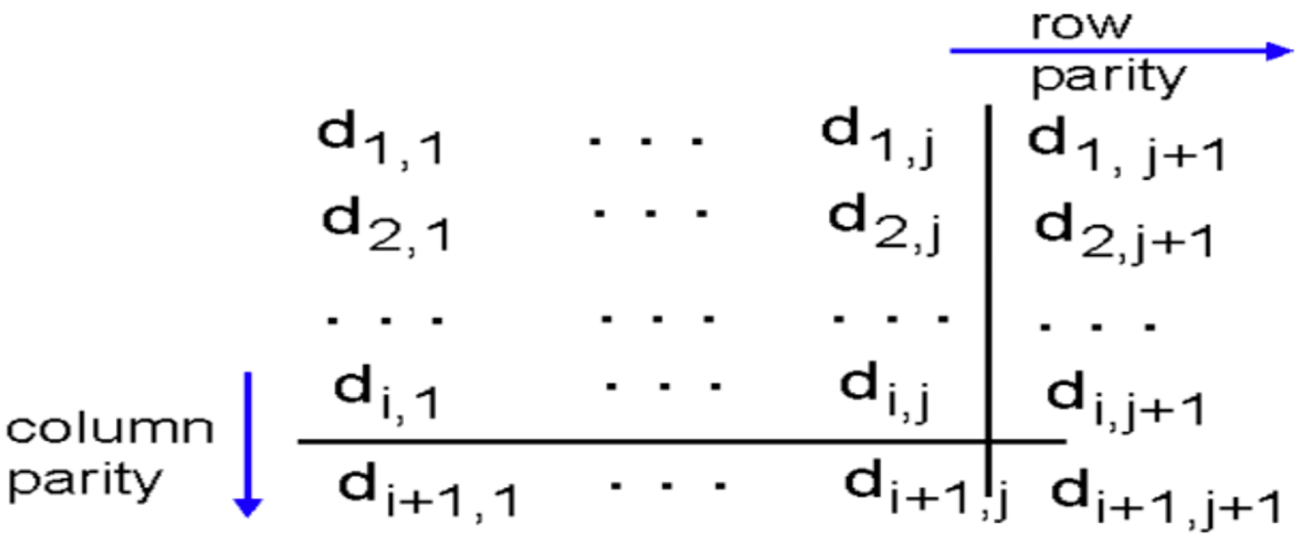
- Send the same request to 3 different instances of the system
- Compare the answers, take the majority

Only commonly used in space applications. Why?

- $$$
- Atmosphere blocks cosmic rays

Error Recovery - Error Correcting Codes (ECC)

Two Dimensional Bit Parity: Detect *and correct* single bit errors



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

Error Recovery - Retry (Network)

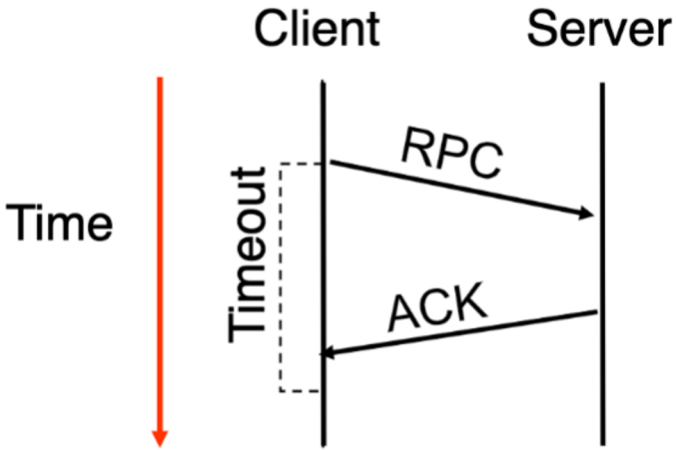
How can we deal with soft faults in the network?

Detect the error:

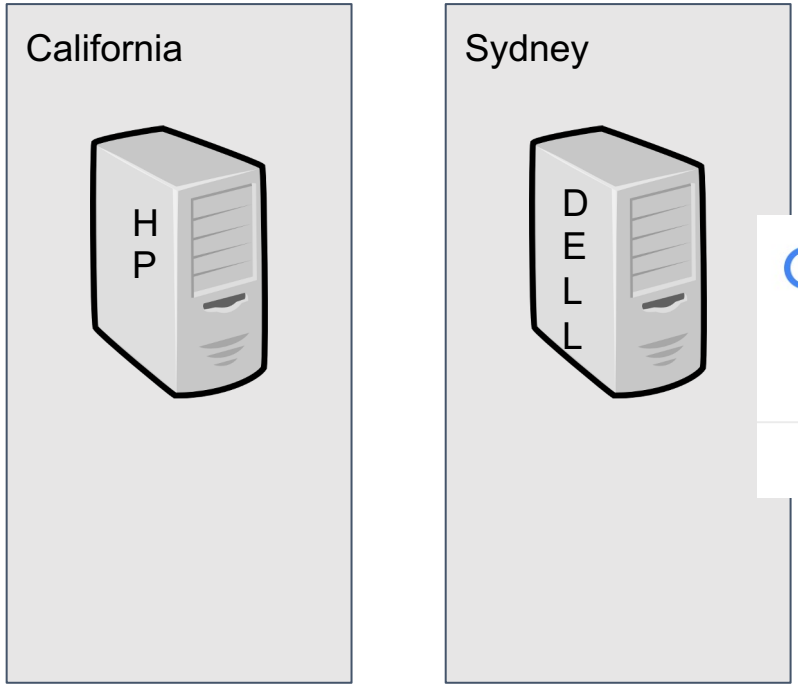
- Timeout
- Parity
- Checksum

Correct the error:

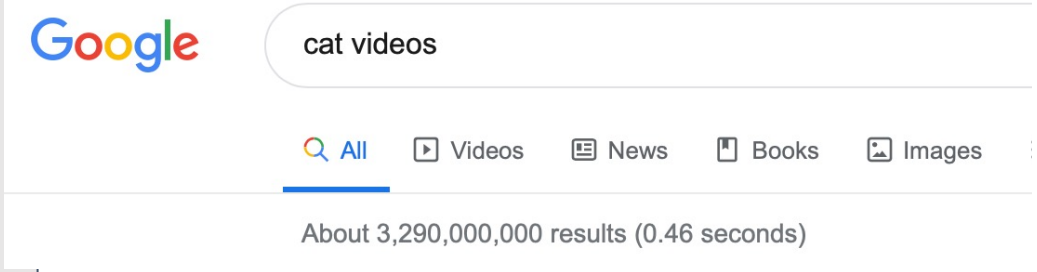
- Retry/Retransmit



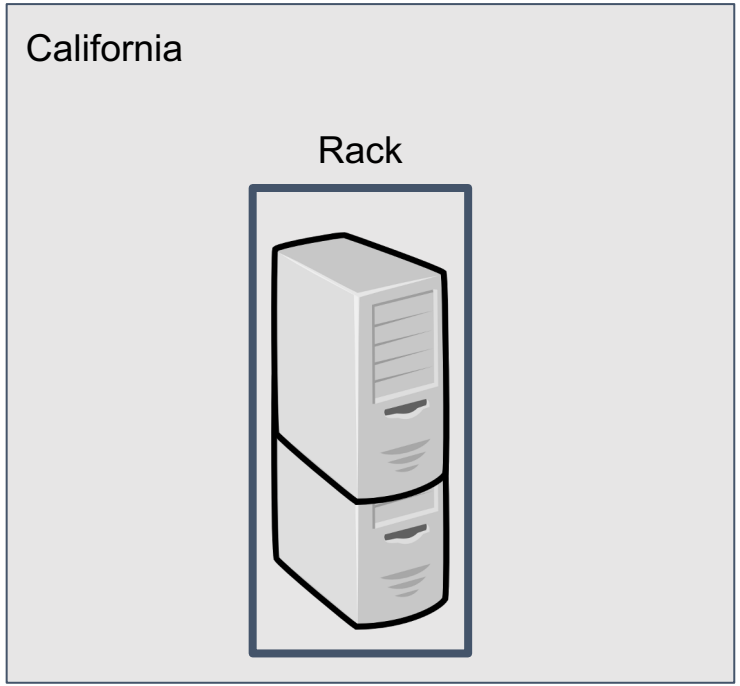
Key question: How Correlated are Failures?



Failures not likely to be correlated

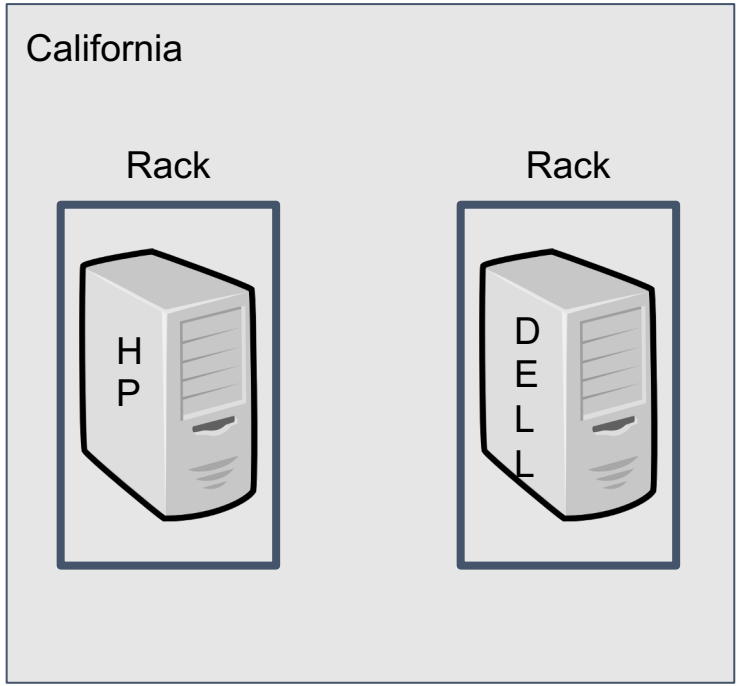


Key question: How Correlated are Failures?



- Top-of-Rack (ToR) switch malfunctions
- Shared UPS breaks

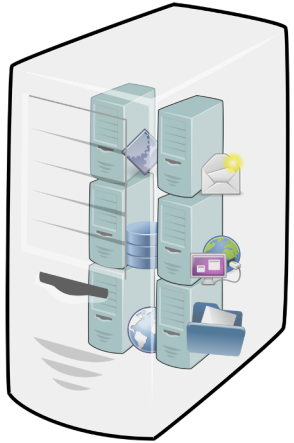
Key question: How Correlated are Failures?



- Fire in the datacenter
- AC breakdown*
- Natural disaster

*** Facebook's first data center drenched by actual cloud, 2013**

Key question: How Correlated are Failures?



Correlation reduces value of redundancy

All of your replicas are virtual machines running on the same physical host?

Everything fails

Fault Tolerant Design

When designing a fault tolerant system, consider

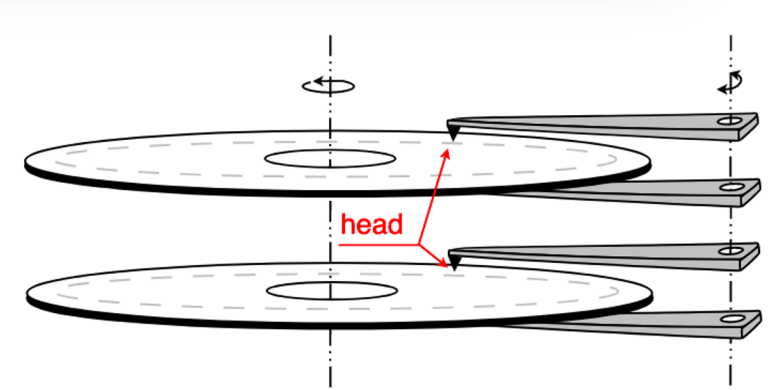
1. The probability of failure of each component
2. The cost of failure
3. The cost of implementing fault tolerance

Outline

- Errors (focus on errors in DRAM and disk drives)
  - Presence, impact, availability metric
- Tools/options for dealing with errors
  - Retries, checksums, CRC, error detection/correction codes, …
- RAID levels and performance
  - For disk systems
- Estimating availability

Back to Hard Drives...

- Stack of spinning disks
- Sequence of small data sectors
  - Usually 4KB



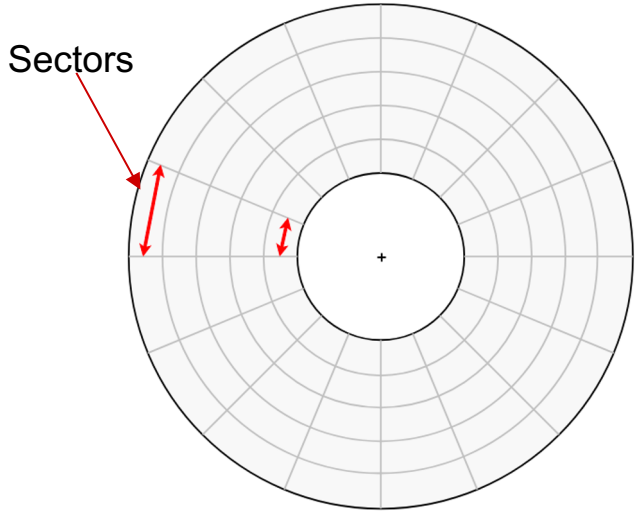


*Image source:Storage subsystem performance: analysis and recipes http://gudok.xyz/sspar/

Fault Tolerant Options

1. ~~Silently return the wrong answer.~~
2. Detect Failure.
   - Add CRC to block header/trailer
   - CRC mismatch → report error
3. Correct / mask the failure.
   - Re-read if received firmware error signal (may help with transient errors, may not)
   - Use error correcting code
   - What errors does ECC help?
     - Bit flip: Yes Block damaged: No
   - Have data stored in multiple places (RAID)

RAID Taxonomy

- Redundant Array of Inexpensive Disks (RAID)
  - Constructed by UC-Berkeley researchers in late 80s
  - Original problem: increase IO capacity with multiple disks

- Solution
  - Use multiple disks to form single logical disk
  - Enhance reliability and fault tolerance

RAID Levels

RAID 0 - Data striping without redundancy

RAID 1 - Mirroring of independent disks

RAID 4 - Data striping plus parity disk

RAID 5 - Data striping plus stripped (rotating) parity

Others (RAID 2, 3, 6)

Preliminaries

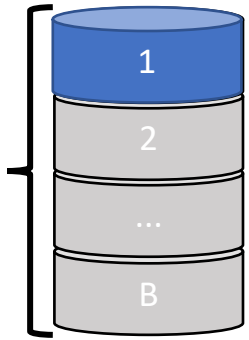
- Definitions
  - Reliability: # of disk failures we can tolerate
  - Latency: time to process Read/Write requests
  - Throughput: bandwidth for R/W requests

- Assumptions:
  - Only consider random R/W in this class
  - Same throughput, latency for R/W access

Single Disk

B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block

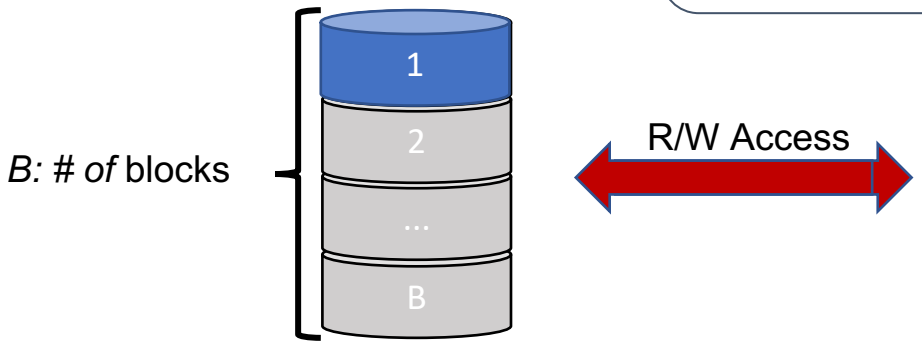
B: # of blocks



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

Single Disk

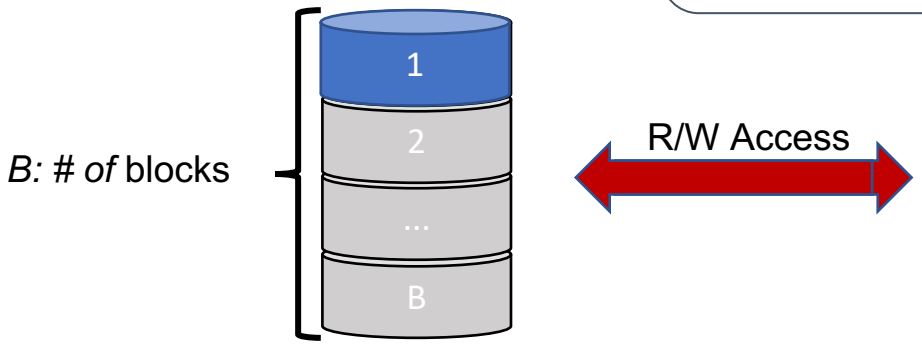
B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block



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

Single Disk

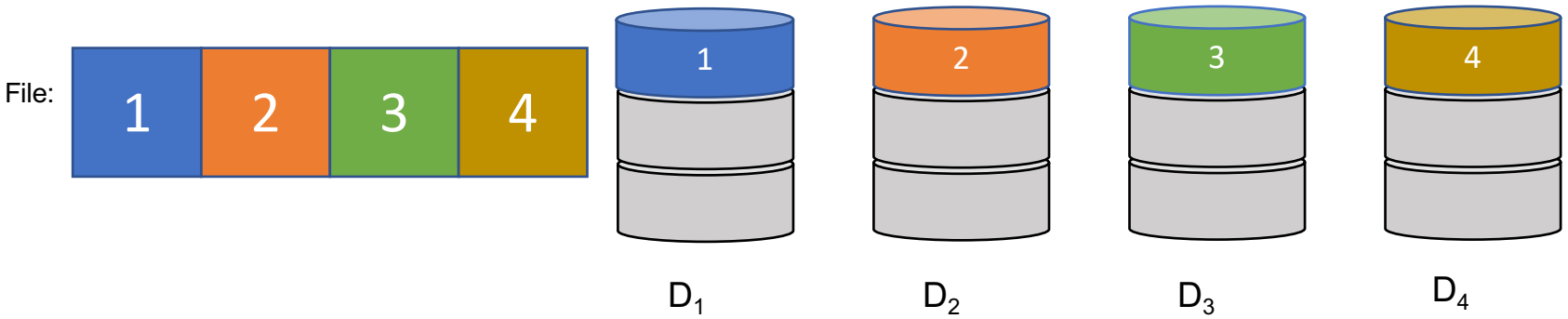
B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block



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

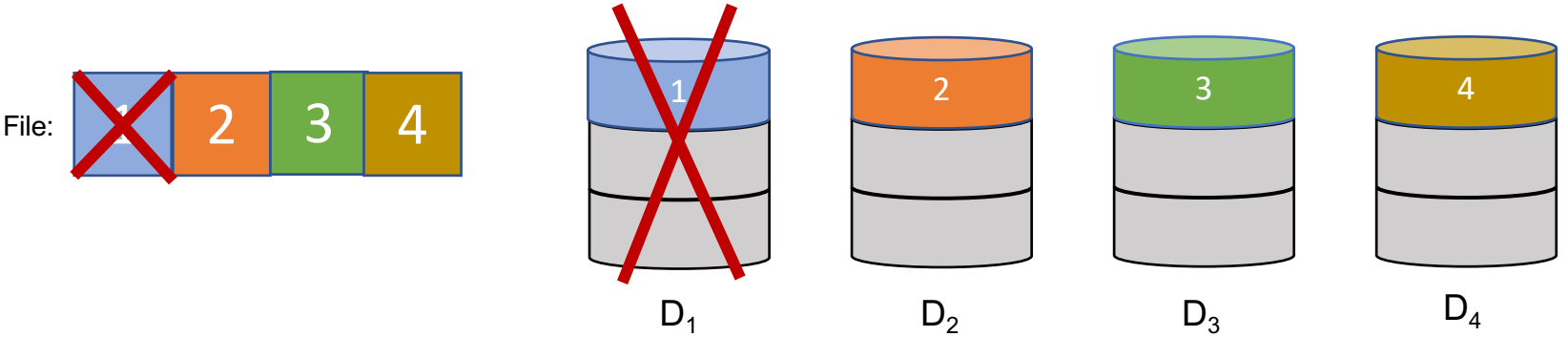
RAID-0: Striping

- To optimize performance
- Interleave data across multiple disks
  - Large file streaming → parallel transfers
  - Small requests benefit from load balancing (If hot files evenly distributed on all disks)



RAID-0: Striping

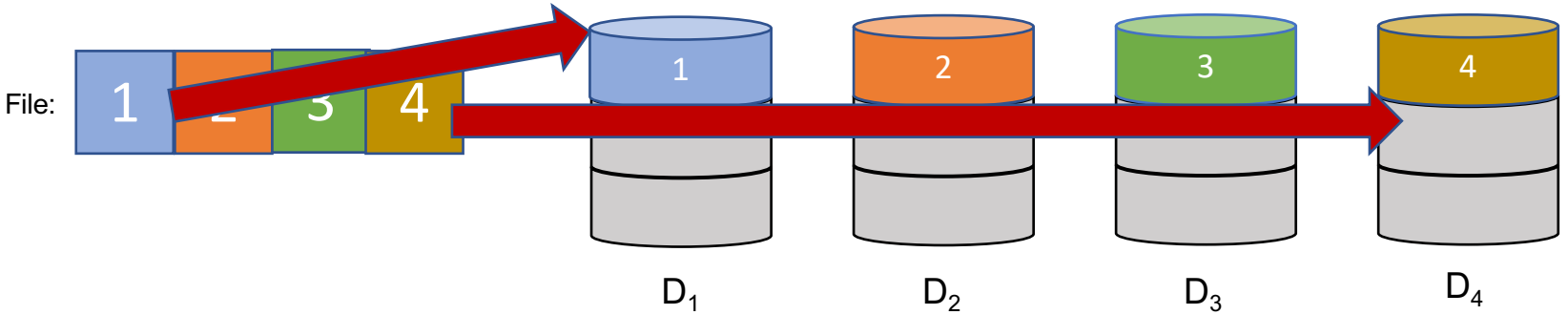
B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block



Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID 0	N*B	0				

RAID-0: Striping

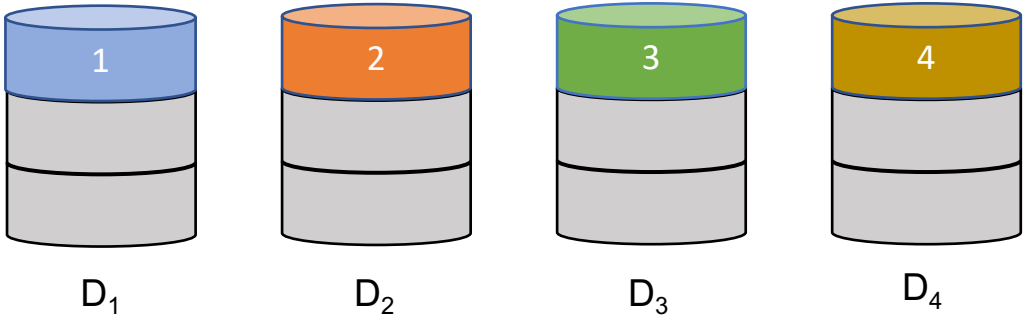
B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block



Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID 0	N*B	0	N*R		N*R	

RAID-0: Striping

B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block



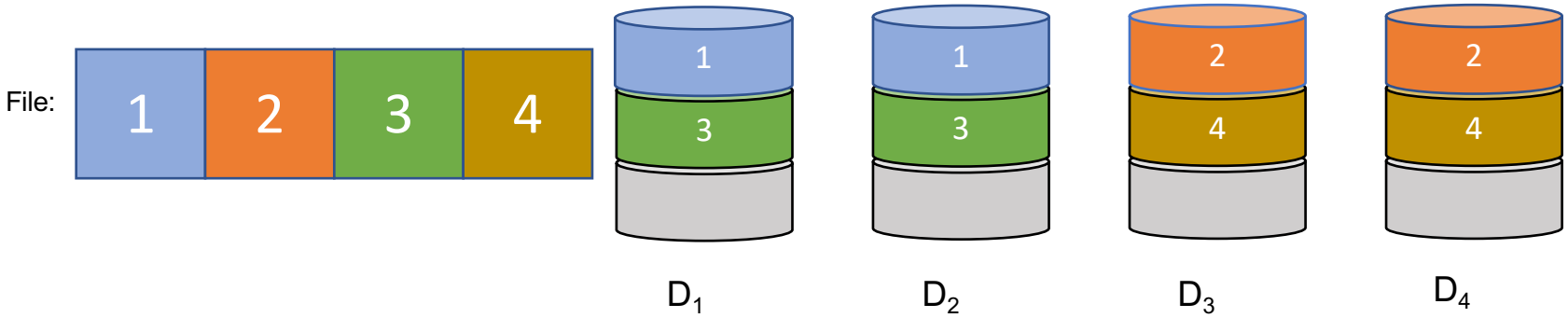
Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID 0	N*B	0	N*R	D	N*R	D

What If A Disk Fails?

- In a striped system
  - Part of the file system is lost
- Periodic Backups?
  - Takes time and effort
  - Newer data after last backup will be lost

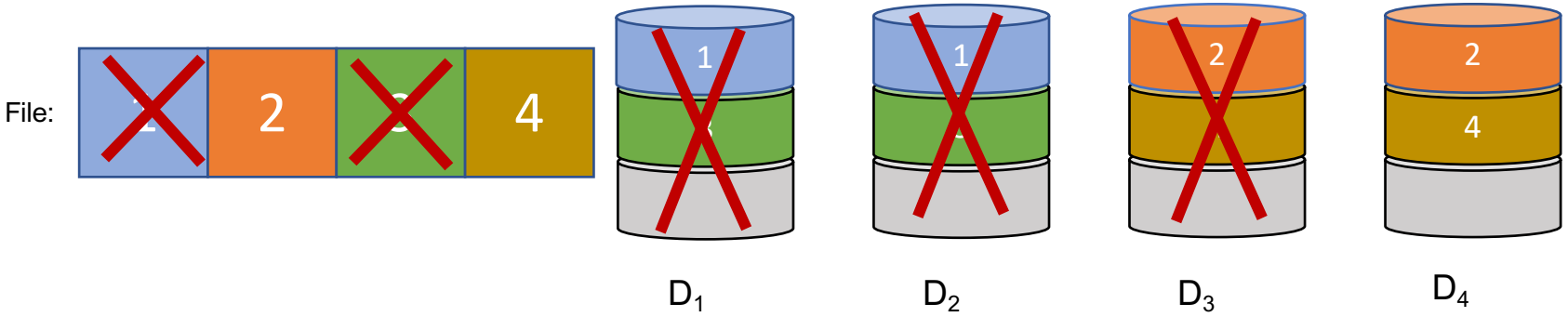
RAID-1: Mirroring

- To achieve better reliability
- Two (or more) copies
- Write both, read either



RAID-1: Mirroring

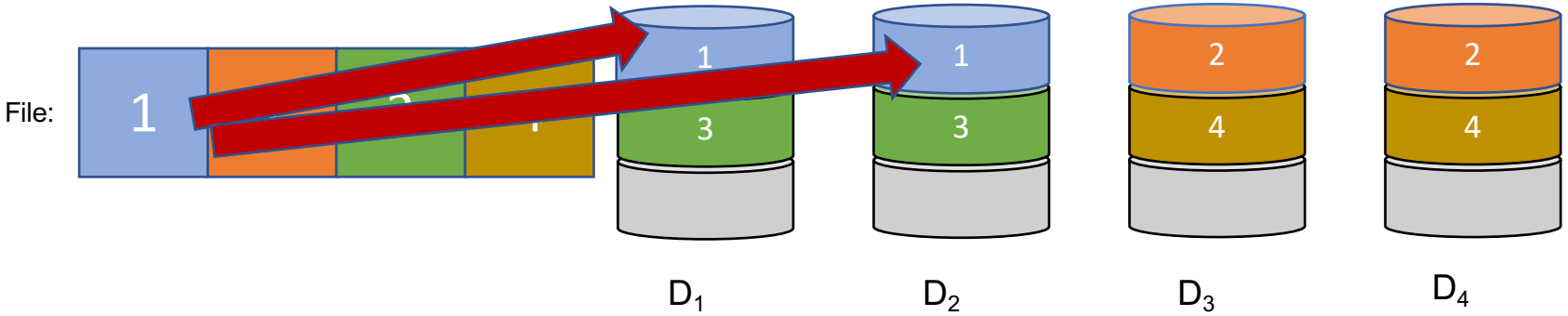
B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block



Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID 1	N/2 *B	1; N/2 (if lucky)				

RAID-1: Mirroring

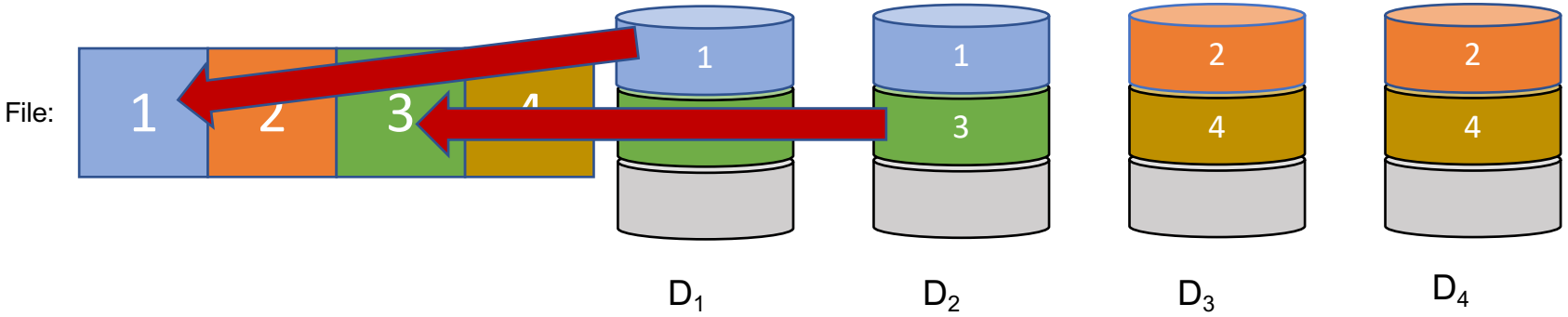
B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block



Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID 1	N/2 *B	1; N/2 (if lucky)	N/2 *R	D		

RAID-1: Mirroring

B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block



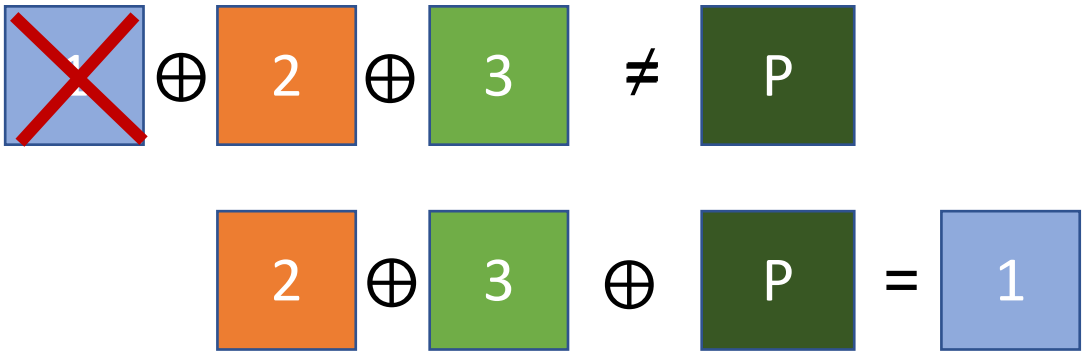
Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID 1	N/2 *B	1; N/2 (if lucky)	N/2 *R	D	N*R	D

Is Mirroring the Best Approach?

- High data redundancy
- Only tolerate 1 disk failure
- Parity disks: same reliability, higher usable capacity
- RAID 4/5

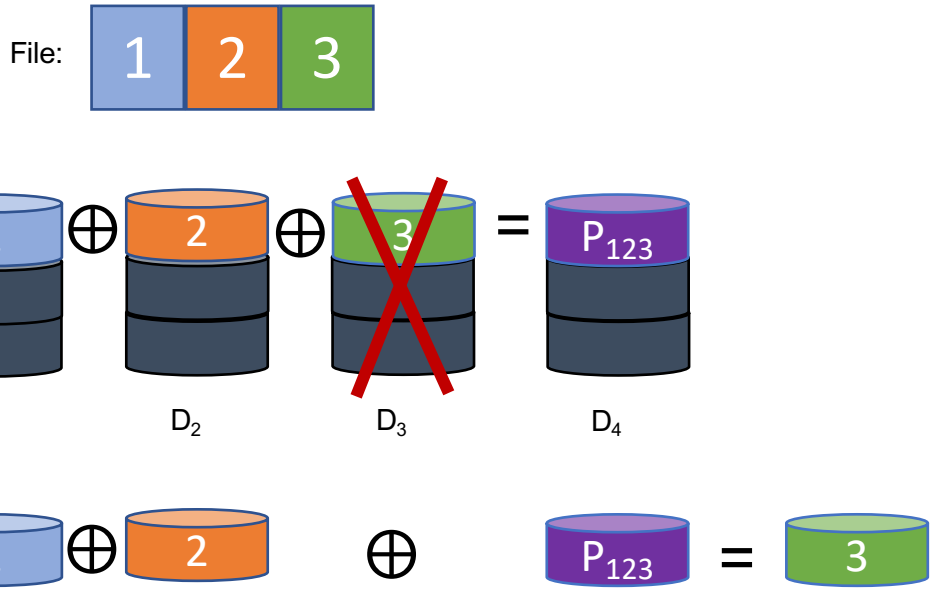
Calculating Parity

- Erasure code:
  - Forward error correction code
  - Detect and correct errors
  - Common example: XOR



RAID-4: Parity Disk

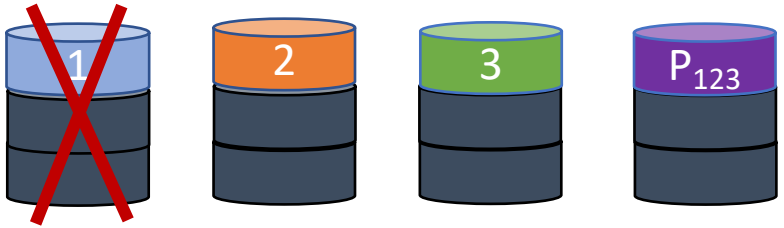
- Capacity: one extra disk needed per stripe



RAID-4: Parity Disk

B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block

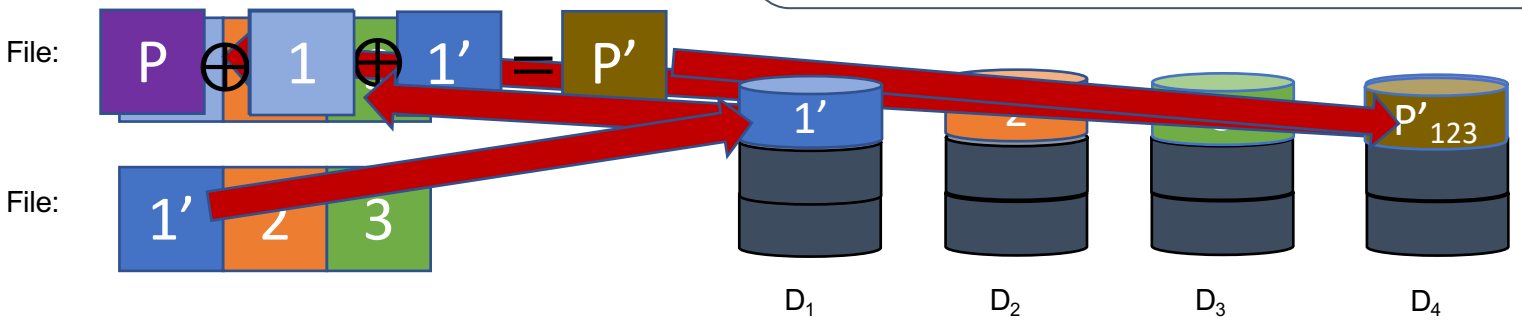
File:



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

RAID-4: Parity Disk

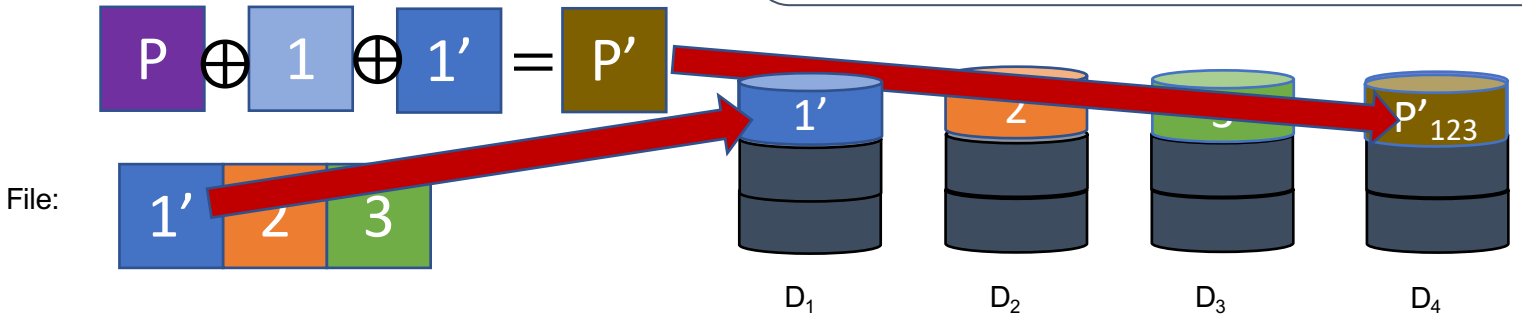
B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block



Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID 4	(N-1)B	1	R/2			

RAID-4: Parity Disk

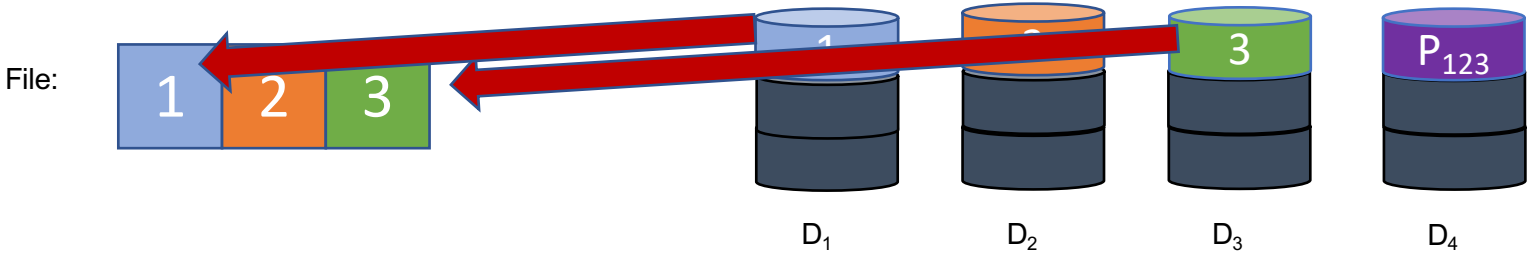
B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block



Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID 4	(N-1)B	1	R/2	2D		

RAID-4: Parity Disk

B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block

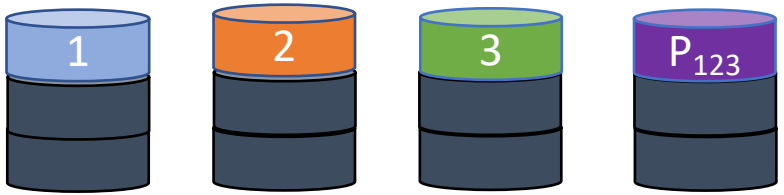


Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID 4	(N-1)B	1	R/2	2D	(N-1)R	

RAID-4: Parity Disk

B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block

File:



D_1 D_2 D_3 D_4

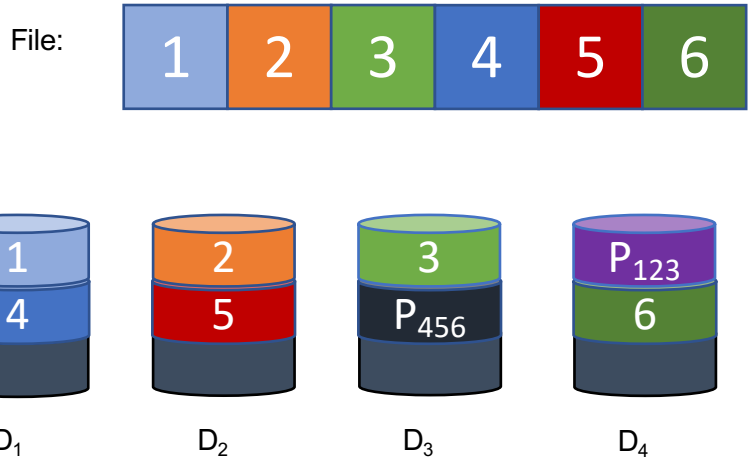
Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID 4	(N-1)B	1	R/2	2D	(N-1)R	D

Parity Disk Bottleneck

- Reads go to the data disks
  - Hopefully load balanced across the disks
- All writes go to the parity disk
  - Even worse: usually result in Read-Modify-Write sequence
  - Parity disk can easily be a bottleneck
  - Wear out very fast! → prone to failures
- Adding disk does not provide any performance gain

RAID-5: Rotating Parity

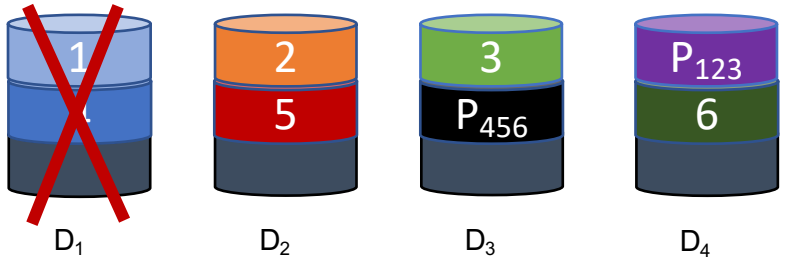
- Distribute parity blocks in round-robin manner



RAID-5: Rotating Parity

B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block

File:

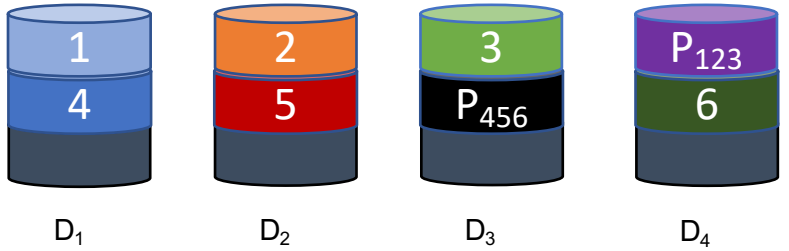


Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID 5	(N-1)B	1				

RAID-5: Rotating Parity

B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block

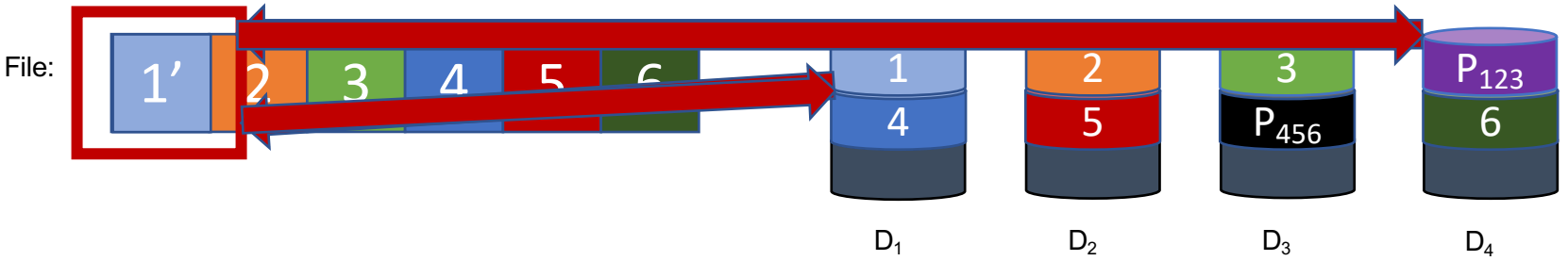
File:



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 per disk
N: # of disks **D**: time to R/W 1 block



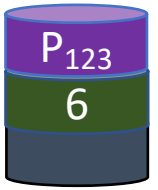
Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID 5	(N-1)B	1	N/4 * R			

RAID-5: Rotating Parity

B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block

File: 1

1. Read old Data block from Data disk
2. Read old Parity block from Parity disk
3. Write (update) new Data block to Data disk
4. Write (update) new Parity block to Parity disk



D_4

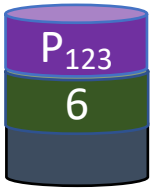
Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID 5	(N-1)B	1	N/4 * R			

RAID-5: Rotating Parity

B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block

File: 1

1. Read old Data block from Data disk
2. Read old Parity block from Parity disk
3. Write (update) new Data block to Data disk
4. Write (update) new Parity block to Parity disk

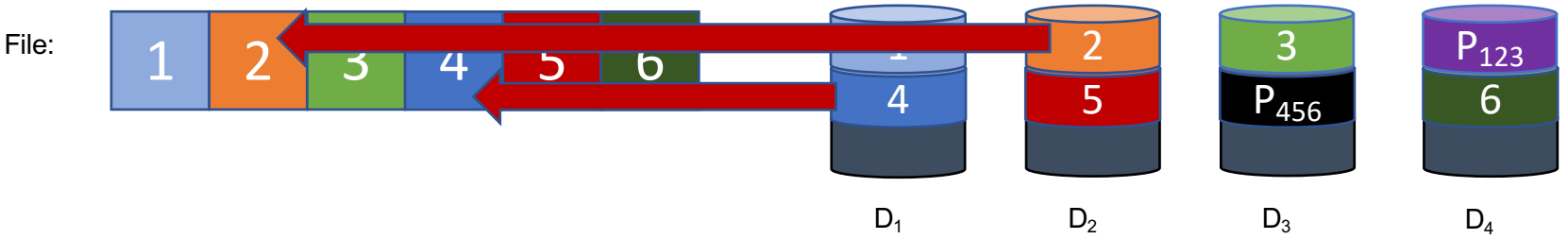


D_4

Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID 5	(N-1)B	1	N/4 * R	2D		

RAID-5: Rotating Parity

B: # of blocks per disk
R: R/W throughput per disk
N: # of disks **D**: time to R/W 1 block



Level	Capacity	Reliability	Write Throughput	Write Latency	Read Throughput	Read Latency
RAID 5	(N-1)B	1	N/4 * R	2D	N * R	D

Recap: RAID

**** Latency Omitted***

Level	Scheme	Capacity	Reliability	Write Throughput	Read Throughput
Single Disk		B	0	R	R
RAID 0	Striping	N * B	0	N * R	N * R
RAID 1	Mirroring	N/2 * B	1 (for sure) N/2 (lucky)	N/2 * R	N * R
RAID 4	Parity Disk	(N-1)B	1	R/2	(N-1)R
RAID 5	Rotating Parity	(N-1)B	1	N/4 * R	N * R

Outline

- Errors (focus on errors in DRAM and disk drives)
  - Presence, impact, availability metric
- Tools/options for dealing with errors
  - Retries, checksums, CRC, error detection/correction codes, …
- RAID levels and performance
  - For disk systems
- Estimating availability

Availability / Reliability Metrics

- Mean Time To First Data Loss (MTTDL) for RAID
  - Calculate from single disk MTTF
- Back of envelop calculation, with simplifying assumptions
  - Independent failures across devices
  - Independent failures over time
  - Failure rate from bottom of bathtub curve

How often are failures

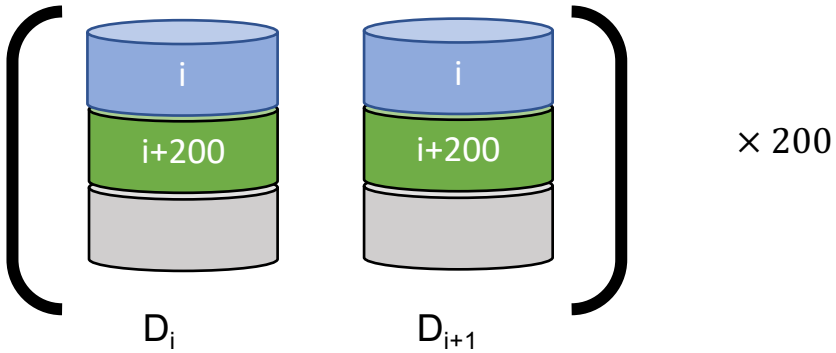
- MTTF of a disk (Mean Time to Failure)
  - $MTTF_{disk}$ ~ 1,200,000 hours (~136 years, <1% per year)
- With 2 disks,
  - Twice more likely to fail
  - Mean time to first disk failure ~ $MTTF_{disk}$ / **2**
- So, we approximate
  - With ***n*** disks, ***n*** times more likely to fail, and
  - Mean time to **first** disk failure ~ $MTTF_{disk}$ / **n**

Reliability without Rebuild

- Assumption: failed disks stay failed
  - We do not try to rebuild it
- If we want to store data size of **200 disks**
- RAID-0: lose data after first disk fails
- $MTTDL_{RAID\text{-}0}$ == mean time to first disk failure
- $MTTDL_{RAID\text{-}0}$ = 136 years / 200 drives
  = 0.65 years

Reliability without Rebuild

- Add 200 more disks to build RAID-1 (Mirroring)
- In total: 200 disk pairs (D_i & D_{i+1}), 400 drives
- RAID-1 fails if at least one disk pair fails



Reliability without Rebuild

- $MTTDL_{pair}$ = **$(MTTF_{drive} / 2)$** + **$MTTF_{drive}$** = 1.5 * $MTTF_{drive}$

  Mean time to first disk failure

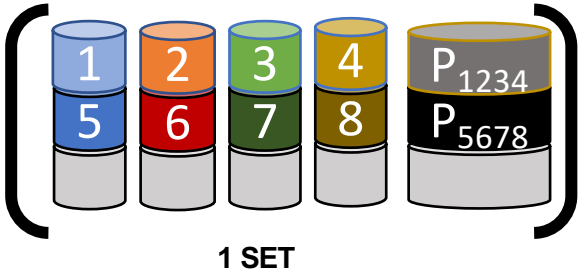
  Mean time to second disk failure

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

- RAID-1 of 400 drives (***200*** mirrored pairs)
- $MTTDL_{RAID\text{-}1}$ = 1.5 * 136 years / 200 pairs = 1.02 years

Reliability without Rebuild

- RAID-4 with capacity of 200 drives
- In total: 250 drives (with 50 parity drives)
- 50 disk sets
  - 1 Set : 4 data disk + 1 parity disk
- RAID-4 fails if at least one set fails



Reliability without Rebuild

- RAID-4: 50 sets (250 disks in total)
- $MTTDL_{RAID\text{-}4}$
  - $MTTDL_{set} = \underline{MTTF_{drive} / 5} + \underline{MTTF_{drive} / 4} = 0.45 * MTTF$

    Mean time to first disk failure

    Mean time to second disk failure

  - $MTTDL_{RAID\text{-}4} = MTTDL_{set}$ / (# of 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

- After a drive failure
  - Data still accessible
  - A second failure is BAD

- Reconstruct the lost 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)

- Simplified math works with approximation
- Based on the canonical Markov model for storage systems
- Interested students: check out this paper (esp. Section 2)
  Greenan, Kevin M., James S. Plank, and Jay J. Wylie. "Mean Time to Meaningless: MTTDL, Markov Models, and Storage System Reliability." In *HotStorage*, pp. 1-5. 2010.

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 **MTTR_drive / MTTF_secondDrive**

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

Rebuild Mechanisms

- 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
  - To what degree should a system mask failures?
  - Replications 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?

Summary

- RAID turns multiple disks into a larger, faster, more reliable disk
- **RAID-0**: Striping
  High performance and capacity, but poor reliability
- **RAID-1**: Mirroring
  High reliability and write performance, but poor capacity
- **RAID-5**: Rotating Parity
  High capacity, save cost, good performance for read-heavy workload

  *Good compromise choice