CS61C

Great Ideas in Computer Architecture
(a.k.a. Machine Structures)

UC Berkeley
Teaching Professor
Dan Garcia

UC Berkeley
Teaching Professor
Lisa Yan

These slides are from CS61C Fall 2020, taught by Dan Garcia and Bora Nikolić. They have been adjusted visually to fit the Fall 2022 format.

Dependability, Parity, ECC, RAID

UC Berkeley
Professor
Bora Nikolić

6 Great Ideas in Computer Architecture

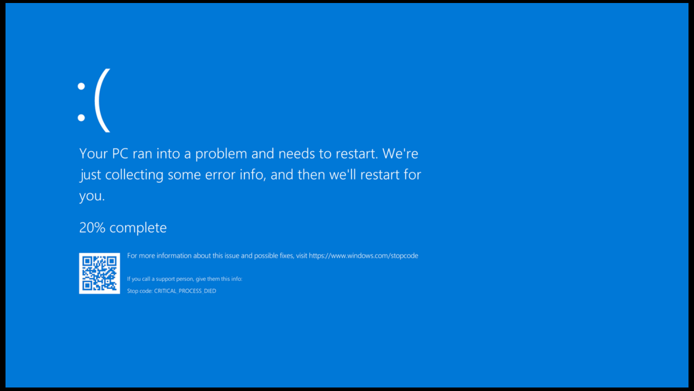
1. Abstraction (Layers of Representation/Interpretation)
2. Moore's Law
3. Principle of Locality/Memory Hierarchy
4. Parallelism
5. Performance Measurement & Improvement
6. Dependability via Redundancy

6 Great Ideas in Computer Architecture

1. Abstraction (Layers of Representation/Interpretation)
2. Moore's Law
3. Principle of Locality/Memory Hierarchy
4. Parallelism
5. Performance Measurement & Improvement
6. Dependability via Redundancy

Computers Fail…

- **May fail transiently…**

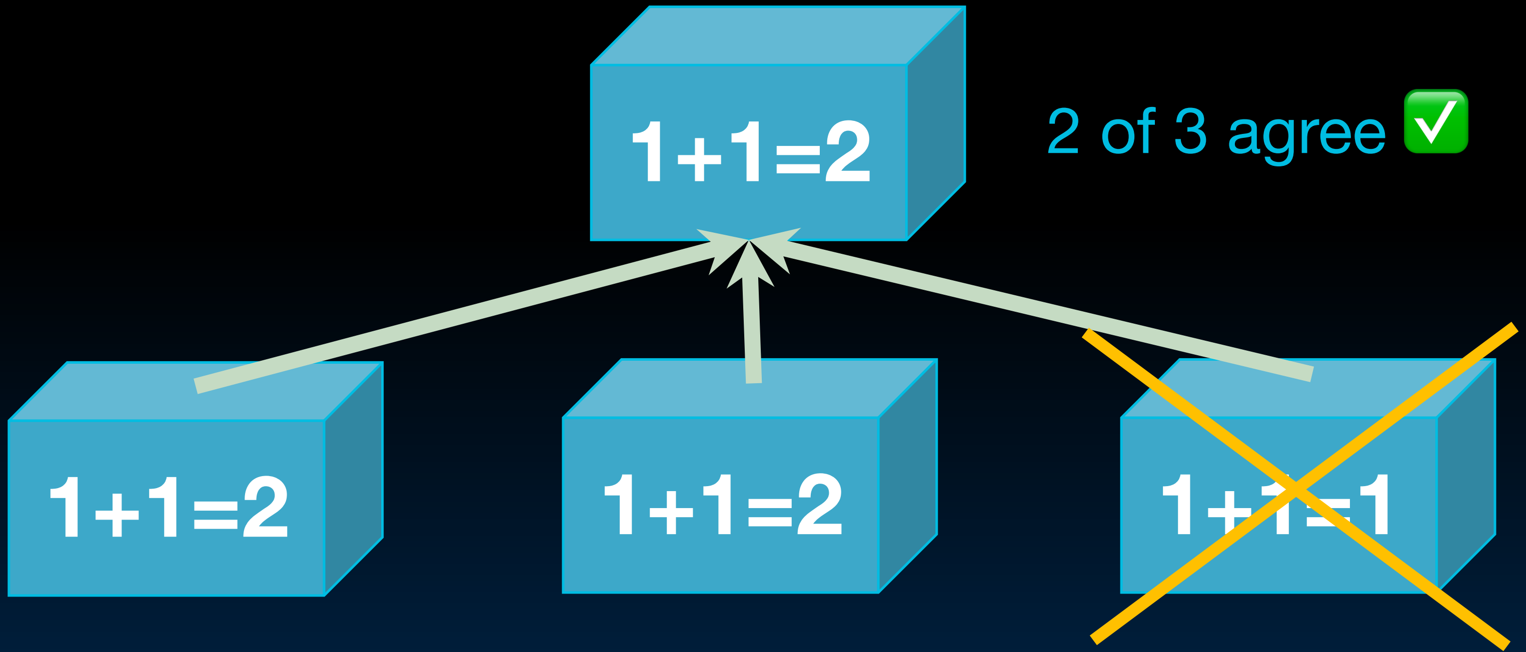


- **…or permanently**

We will discuss hardware failures and methods to mitigate them

Great Idea #6: Dependability via Redundancy

- Design with redundancy so that a failing piece doesn't make the whole system fail.



Increasing transistor density reduces the cost of redundancy

Garcia, Yan

Great Idea #6: Dependability via Redundancy

- **Applies to everything from datacenters to storage to memory to instructors!**
  - Redundant datacenters so that can lose 1 datacenter but Internet service stays online;
  - Redundant disks so that can lose 1 disk but not lose data (Redundant Arrays of Independent Disks/RAID);
  - Redundant memory bits so that can lose 1 bit but no data (Error Correcting Code/ECC Memory).

Garcia, Yan

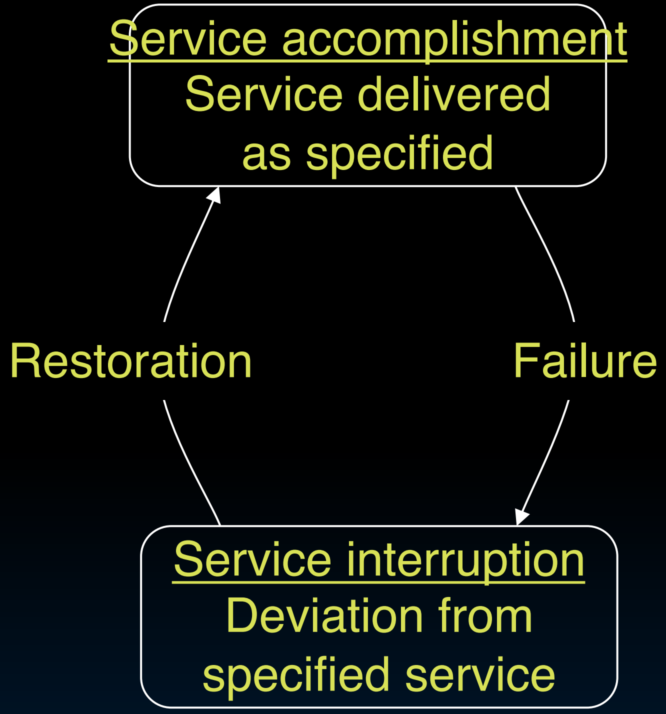
Dependability Metrics

- Dependability Metrics
- Error Detection
- Error Detection and Correction
- Error Correcting Code (ECC) Examples
- Redundancy with RAID

Garcia, Yan

Dependability

- Fault: **failure of a component**
  - May or may not lead to system failure



Dependability via Redundancy: Time vs. Space

- Spatial Redundancy – **replicated data or check information or hardware to handle hard and soft (transient) failures**
- Temporal Redundancy – **redundancy in time (retry) to handle soft (transient) failures**

Garcia, Yan

Dependability Measures

- Reliability: **Mean Time To Failure (MTTF)**
- Service interruption: **Mean Time To Repair (MTTR)**
- **Mean time between failures (MTBF)**
  - MTBF = MTTF + MTTR
- Availability = **MTTF / (MTTF + MTTR)**
- **Improving Availability**
  - Increase MTTF: More reliable hardware/software + Fault Tolerance
  - Reduce MTTR: improved tools and processes for diagnosis and repair

Garcia, Yan

Availability Measures

- **Availability = MTTF / (MTTF + MTTR) as %**
  - MTTF, MTBF usually measured in hours
- **Since hope rarely down, shorthand is "number of 9s of availability per year"**
  - 1 nine: 90% => 36 days of repair/year
  - 2 nines: 99% => 3.6 days of repair/year
  - 3 nines: 99.9% => 526 minutes of repair/year
  - 4 nines: 99.99% => 53 minutes of repair/year
  - 5 nines: 99.999% => 5 minutes of repair/year

Reliability Measures

- **Another is average number of failures per year: Annualized Failure Rate (AFR)**
  - E.g., 1000 disks with 100,000 hour MTTF
  - 365 days * 24 hours = 8760 hours
  - (1000 disks * 8760 hrs/year) / 100,000
    = 87.6 failed disks per year on average
  - 87.6/1000 = 8.76% annual failure rate
- **Google's 2007 study* found that actual AFRs for individual drives ranged from 1.7% for first year drives to over 8.6% for three-year old drives**

Hard Drive Failures

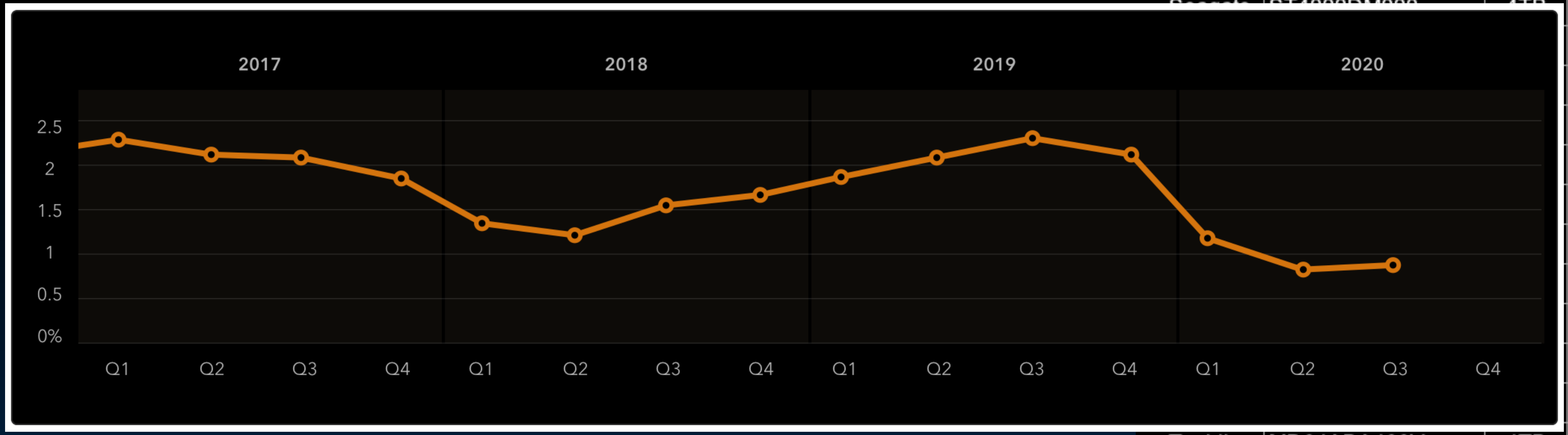
- Annualized hard-drive failure rates

Backblaze Lifetime Annualized Hard Drive Failure Rates

Reporting period: April 20, 2013 through 30 September 2020 inclusive

MFG	Model	Drive Size	Avg Age	Drive Count	Drive Days	Drive Failures	AFR
HGST	HMS5C4040ALE640	4TB	53.8	3,023	12,476,131	170	0.50%
HGST	HMS5C4040BLE640	4TB	47.5	12,737	23,069,669	270	0.43%
HGST	HUH728080ALE600	8TB	33.2	1,032	1,113,086	20	0.66%
HGST	HUH721212ALE600	12TB	12.1	2,600	908,168	10	0.40%
HGST	HUH721212ALE604	12TB	1.5	1,909	67,002	2	1.09%
HGST	HUH721212ALN604	12TB	18.0	10,838	5,891,093	75	0.46%
[illegible]	[illegible]	[illegible]	59.4	19,024	57,835,040	4,029	2.54%
			65.9	886	3,145,481	86	1.00%
			48.1	9,801	14,491,613	414	1.04%
			38.4	14,425	17,133,800	569	1.21%
			36.0	1,200	1,335,422	28	0.77%
			26.9	28,867	30,486,342	1,739	2.08%
			7.2	18,339	3,909,226	108	1.01%
			3.8	6,139	685,622	18	0.96%
			0.7	2,400	21,120	-	0.00%
			9.9	60	15,895	1	2.30%
			0.4	60	300	-	0.00%
Toshiba	MD04ABA400V	4TB	64.3	99	261,874	5	0.70%
Toshiba	MG07ACA14TA	14TB	5.9	17,318	2,983,751	84	1.03%
			TOTALS	150,757	175,830,635	7,628	1.58%

BACKBLAZE



Failures in Time (FIT) Rate

- The Failures In Time (FIT) rate of a device is the number of failures that can be expected in one billion (109) device-hours of operation
  - Or 1000 devices for 1 million hours,
    1 million devices for 1000 hours each
- MTBF = 1,000,000,000 x 1/FIT
- Relevant: Automotive safety integrity level (ASIL) defines FIT rates for different classes of components in vehicles

Dependability Design Principle

- Dependability Design Principle: **No single points of failure**
  - "Chain is only as strong as its weakest link"
- **Dependability corollary of Amdahl's Law**
  - Doesn't matter how dependable you make one portion of system
  - Dependability limited by part you do not improve

Error Detection

- Dependability Metrics
- Error Detection
- Error Detection and Correction
- Error Correcting Code (ECC) Examples
- Redundancy with RAID

Error Detection/Correction Codes

- **Memory systems generate errors (accidentally flipped bits)**
  - DRAMs store very little charge per bit

    "Soft" errors occur occasionally when cells are struck by alpha particles or other environmental upsets

    "Hard" errors" can occur when chips permanently fail
  - Problem gets worse as memories get denser and larger
- **Memories protected against soft errors with** EDC/ECC
- **Extra bits are added to each data-word**
  - Used to detect and/or correct faults in the memory system
  - Each data word value mapped to unique *code word*
  - A fault changes valid code word to invalid one, which can be detected

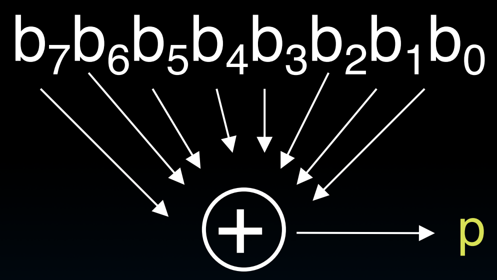
Block Code Principles

- Hamming distance = difference in # of bits
- `p = 011011`, `q = 001111`, Ham. distance (p,q) = 2
- `p = 011011`,
  `q = 110001`,
  distance (p,q) = ?
- Can think of extra bits as creating a code with the data
- What if minimum distance between codewords is 2 and get a 1-bit error?

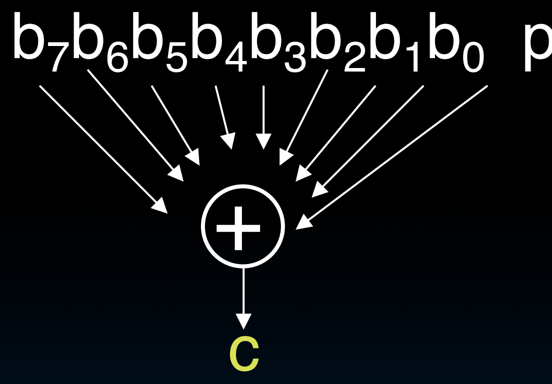
Richard Hamming, 1915-98
Turing Award Winner

Parity: Simple Error-Detection Coding

- Each data value, before it is written to memory is "tagged" with an extra bit to force the stored word to have *even parity*:



- Each word, as it is read from memory is "checked" by finding its parity (including the parity bit).



- Minimum Hamming distance of parity code is 2
- A non-zero parity check indicates an error occurred:
  - 2 errors (on different bits) are not detected
  - Nor any even number of errors, just odd numbers of errors are detected

Garcia, Nikolić

Parity Example

- Data 0101 0101
- 4 ones, even parity now
- Write to memory:
  0101 0101 0
  to keep parity even
- Data 0101 0111
- 5 ones, odd parity now
- Write to memory:
  0101 0111 1
  to make parity even
- Read from memory
  0101 0101 0
- 4 ones => even parity, so no error
- Read from memory
  1101 0101 0
- 5 ones => odd parity, so error
- What if error is in parity bit?

Error Detection and Correction

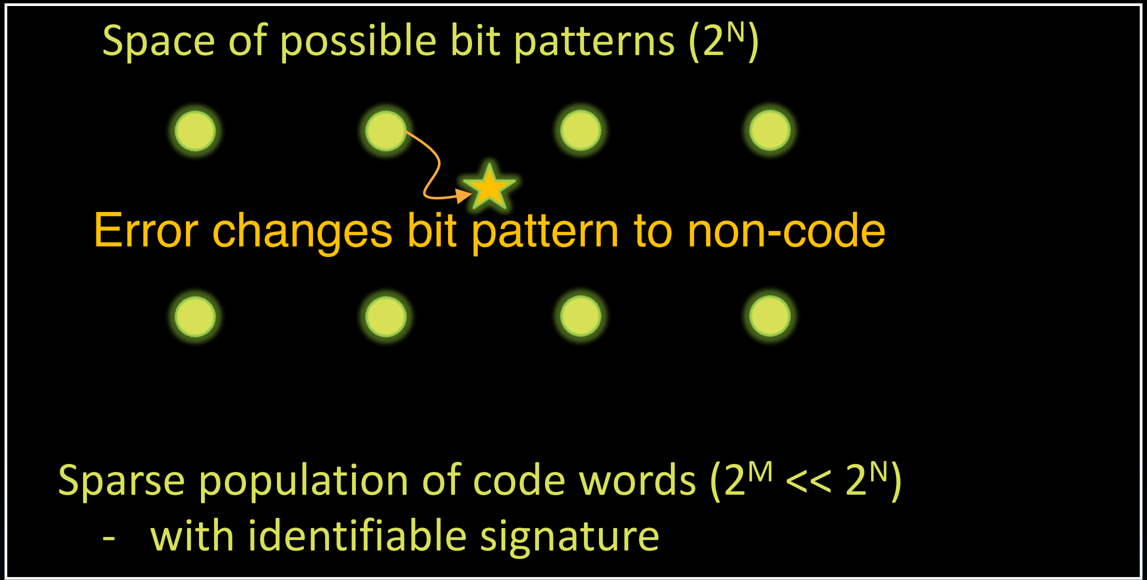
- Dependability Metrics
- Error Detection
- Error Detection and Correction
- Error Correcting Code (ECC) Examples
- Redundancy with RAID

Garcia, Yan

Suppose Want to Correct One Error?

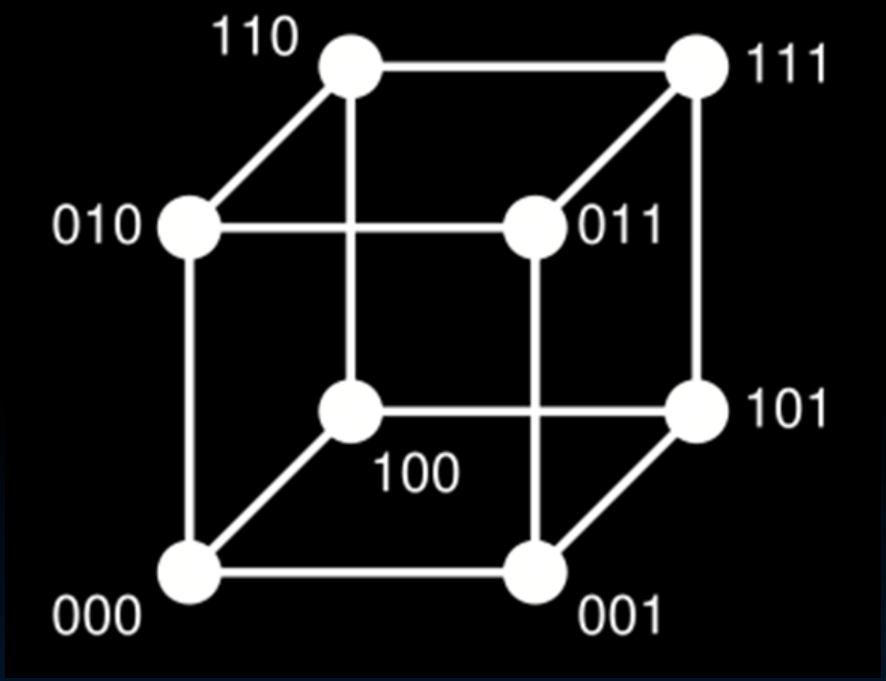
- **Hamming came up with simple to understand mapping to allow Error Correction at minimum distance of three**
  - Single error correction, double error detection
- **Called "Hamming ECC," or Hamming Error Correction Code**
  - Worked weekends on relay computer with unreliable card reader, frustrated with manual restarting
  - Got interested in error correction; published 1950
  - R. W. Hamming, "Error Detecting and Correcting Codes," *The Bell System Technical Journal*, Vol. XXVI, No 2 (April 1950) pp 147-160.

Detecting/Correcting Code Concept



- **Detection**: bit pattern fails codeword check
- **Correction**: map to nearest valid code word

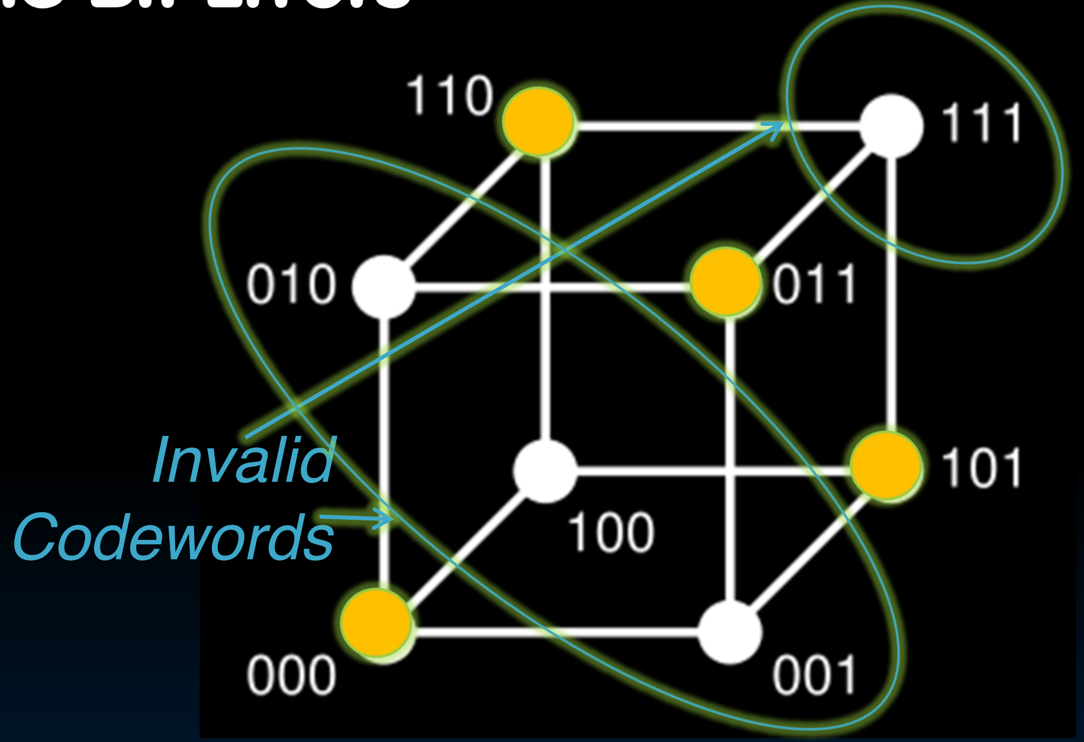
Hamming distance: Eight Code Words



Garcia, Yan

Hamming Distance 2: Detection

- Detect Single Bit Errors

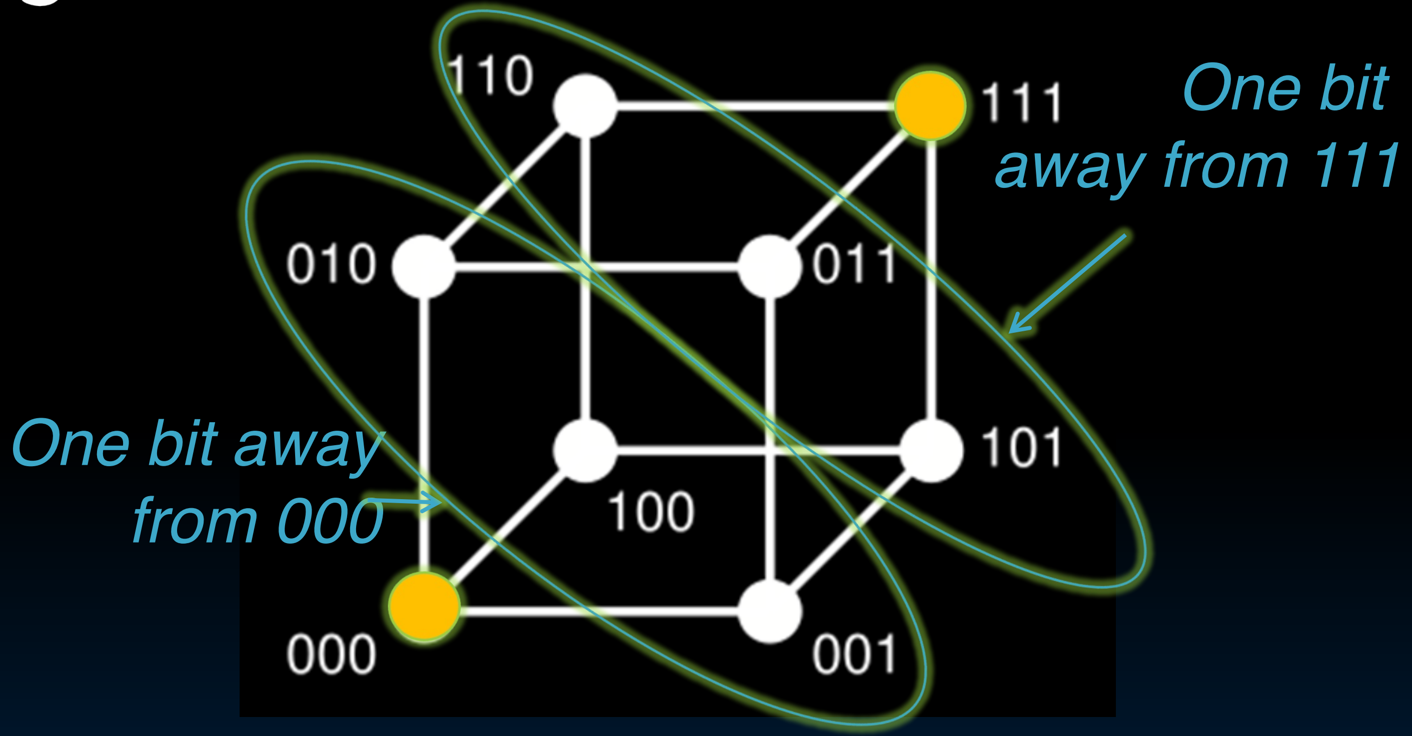


- No 1-bit error goes to another valid codeword
- ½ codewords are valid

Garcia, Yan

Hamming Distance 2: Correction

- Correct Single Bit Errors



- 1-bit errors near valid codewords
- ½ codewords are valid

Error Correcting Code (ECC) Example

- Dependability Metrics
- Error Detection
- Error Detection and Correction
- Error Correcting Code (ECC) Examples
- Redundancy with RAID

Hamming ECC

- **The following example is of a Hamming Error Code (ECC).**
- **Interleave data and parity bits**
- **Place parity bits at binary positions 1, 10, 100, etc .**
  - p1 covers all positions with LSB = 1
  - p2 covers all positions with next to LSB = 1, etc
  - Can continue indefinitely

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
Parity bit coverage	p1																					
	p2																					...
	p4																					
	p8																					
	p16																					

Garcia, Yan

Hamming ECC Encoding (1/3)

Set parity bits to create even parity for each group

- A byte of data: `10011010`
- Create the coded word, leaving spaces for the parity bits:
- `_ _ 1 _ 0 0 1 _ 1 0 1 0`

  `1 2 3 4 5 6 7 8 9 a b c` – bit position

- Calculate the parity bits

Garcia, Yan

Hamming ECC Encoding (2/3)

- Position 1 checks bits 1,3,5,7,9,11:
  ? _ 1 _ 0 0 1 _ 1 0 1 0. set position 1 to a _:
- Position 2 checks bits 2,3,6,7,10,11:
  0 ? 1 _ 0 0 1 _ 1 0 1 0. set position 2 to a _:
- Position 4 checks bits 4,5,6,7,12:
  0 1 1 ? 0 0 1 _ 1 0 1 0. set position 4 to a _:
- Position 8 checks bits 8,9,10,11,12:
  0 1 1 1 0 0 1 ? 1 0 1 0. set position 8 to a _:

Hamming ECC Encoding (3/3)

- **Final code word:** 011100101010
- **Data word:** 1 001 1010

Garcia, Yan

Hamming ECC Decoding (1/4)

- **Suppose receive**
  011100101110

0 1 1 1 0 0 1 0 1 1 1 0

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
Parity bit coverage	p1																					
	p2																					…
	p4																					
	p8																					
	p16																					

Garcia, Yan

Hamming ECC Decoding (2/4) – Error Check

- Suppose receive

```
011100101110
0 1 0 1 1 1      ✅
 11  01  11      ❌-Parity 2 in error
   1001    0     ✅
       01110     ❌-Parity 8 in error
```

- Implies position 8+2=10 is in error

```
011100101110
```

Hamming ECC Decoding (2/4) – Error Correct

- Flip the incorrect bit …

  0111001010 10

Hamming ECC Decoding (2/4) – Error Correct

- Suppose receive

```
011100101010
0 1 0 1 1 1   ✅
 11  01  01   ✅
   1001    0  ✅
       01010  ✅
```

What if More Than 2-Bit Errors?

- Use double-error correction, triple-error detection (DECTED)
- Network transmissions, disks, distributed storage common failure mode is bursts of bit errors, not just one or two bit errors
  - Contiguous sequence of *B* bits in which first, last and any number of intermediate bits are in error
  - Caused by impulse noise or by fading in wireless
  - Effect is greater at higher data rates
- Solve with Cyclic Redundancy Check (CRC), interleaving or other more advanced codes

Garcia, Yan

Redundancy with RAID

- Dependability Metrics
- Error Detection
- Error Detection and Correction
- Error Correcting Code (ECC) Examples
- Redundancy with RAID

Garcia, Yan

RAID: Redundant Arrays of (Inexpensive) Disks

- Data is stored across multiple disks
- Files are "striped" across multiple disks
- Redundancy yields high data availability
  - Availability: service still provided to user, even if some components failed
- Disks will still fail
- Contents reconstructed from data redundantly stored in the array
  - Capacity penalty to store redundant info
  - Bandwidth penalty to update redundant info

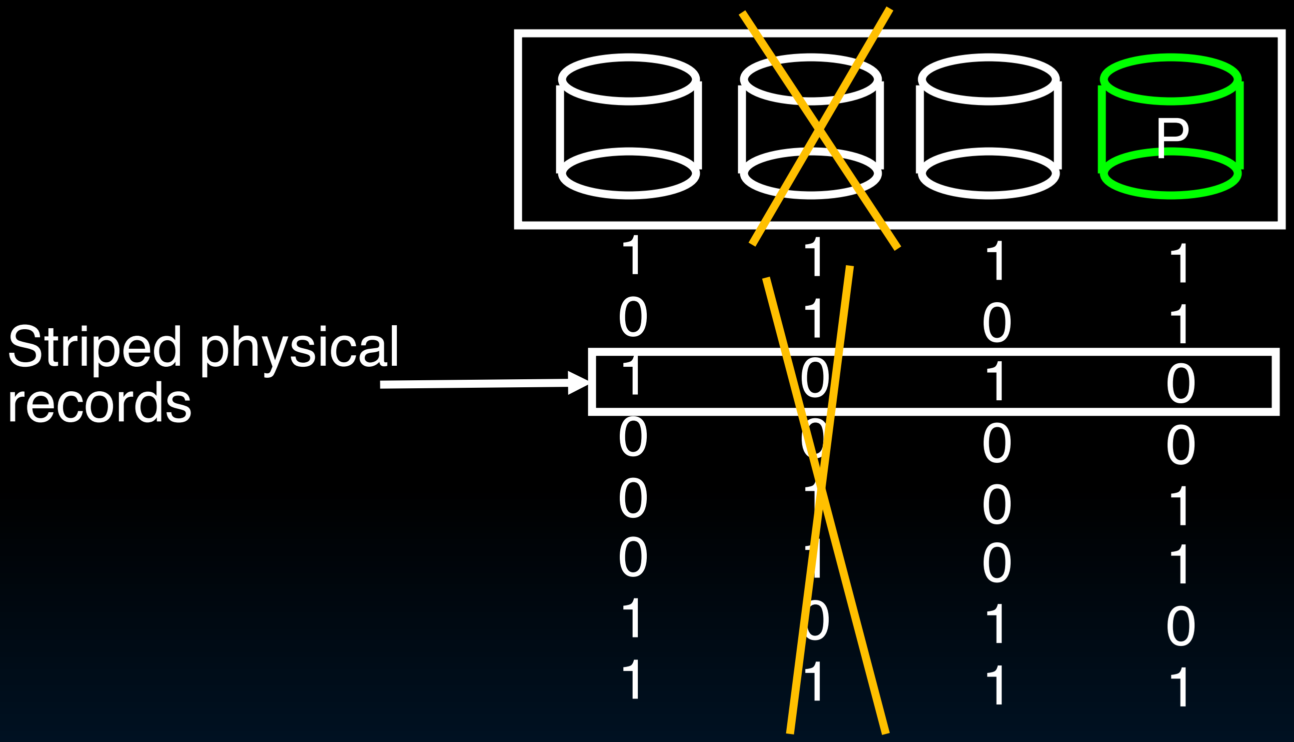
Redundant Arrays of Inexpensive Disks RAID 1: Disk Mirroring/Shadowing

- Each disk is fully duplicated onto its "mirror"
  - Very high availability can be achieved
- Writes limited by single-disk speed
- Reads may be optimized
- Most expensive solution: 100% capacity overhead

RAID 3: Parity Disk

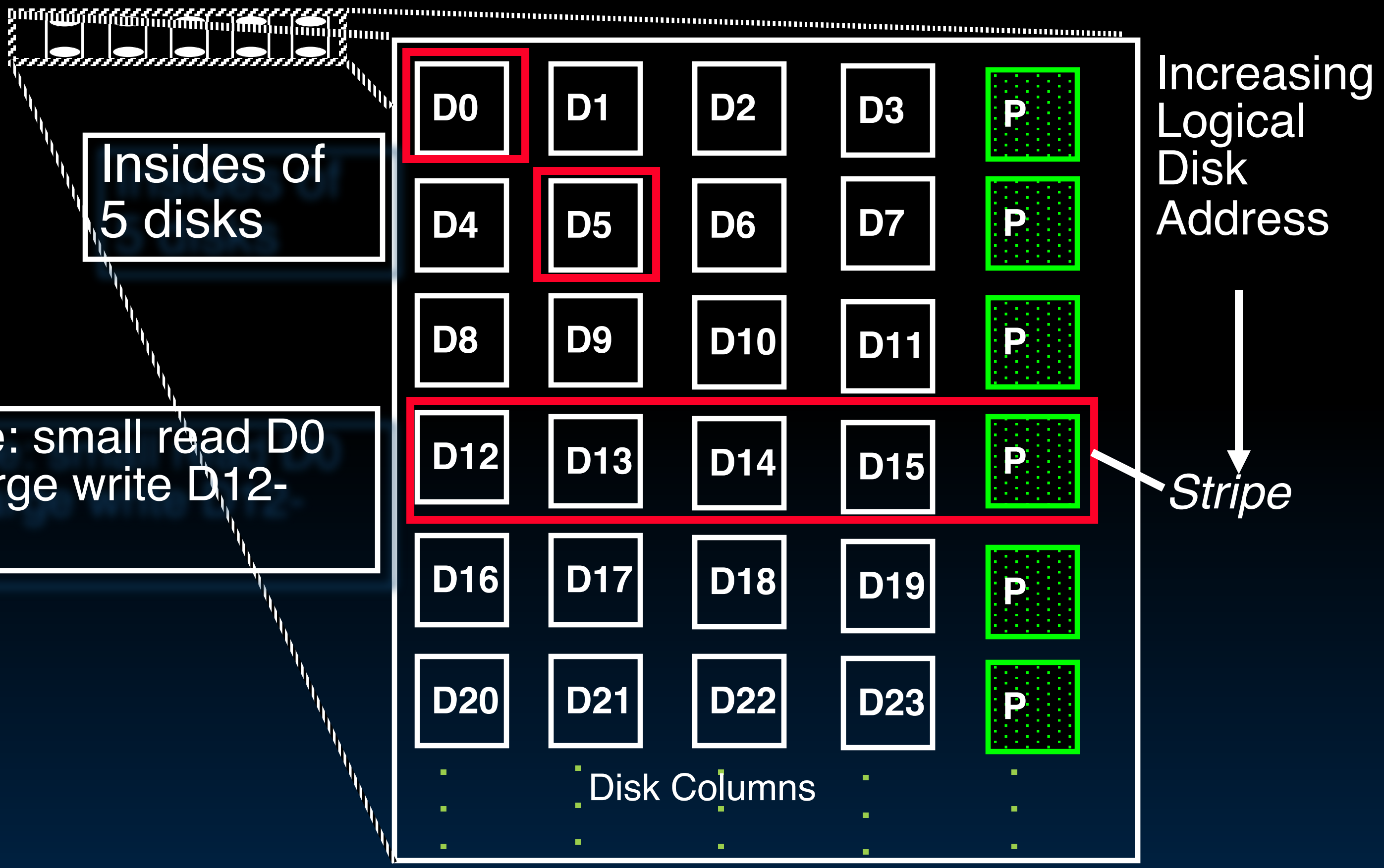
10010011
11001101
10010011
. . .

logical record



- **P contains sum of other disks per stripe mod 2 ("parity")**
- **If disk fails, subtract P from sum of other disks to find missing information**

Raid 4: High I/O Rate Parity

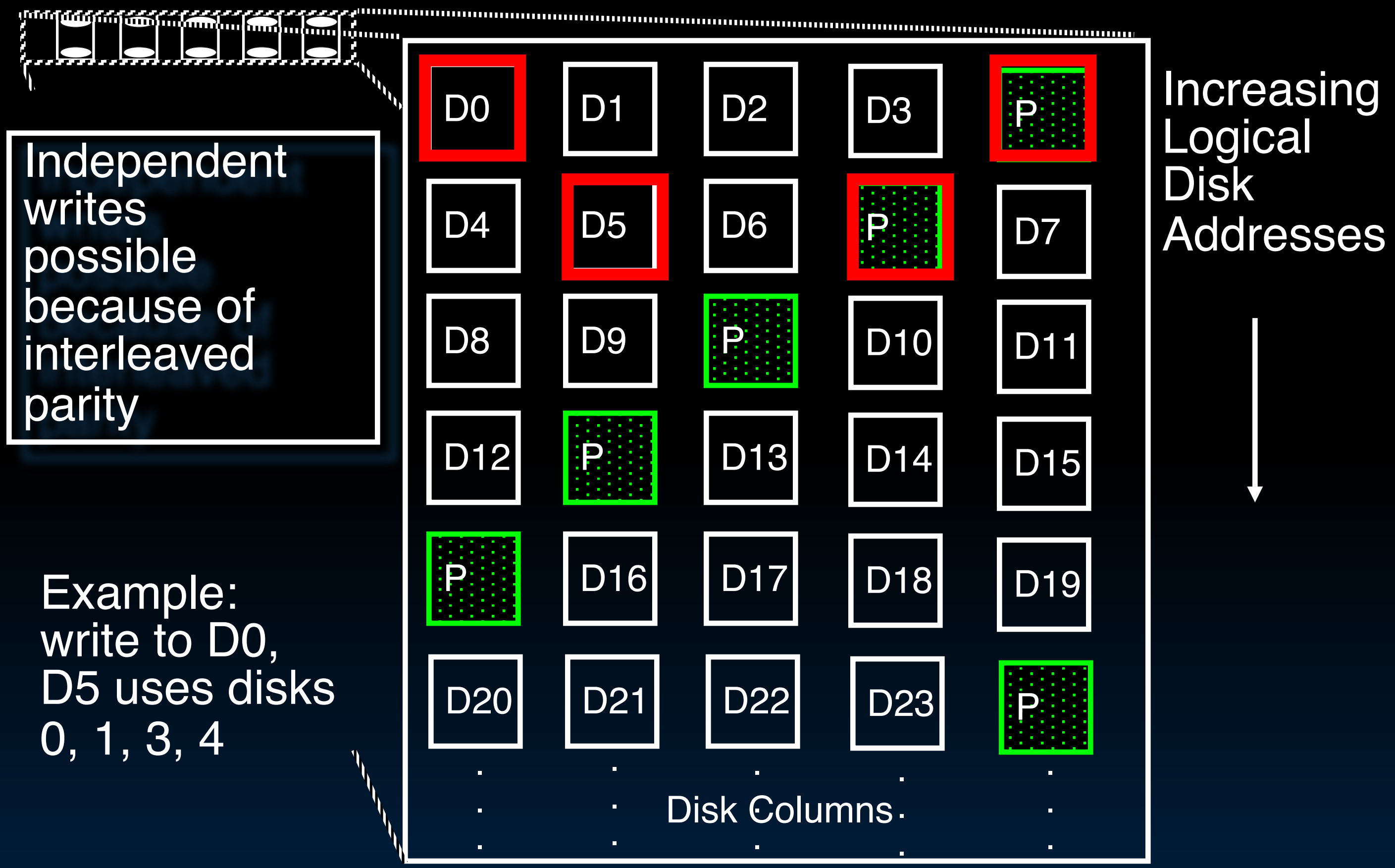


Example: small read D0 & D5, large write D12-D15

Inspiration for RAID 5

- **RAID 4 works well for small reads**
- **Small writes (write to one disk):**
  - Option 1: read other data disks,create new sum and write to Parity Disk
  - Option 2: since P has old sum, compare old data to new data, add the difference to P
- **Small writes are limited by Parity Disk: Write to D0, D5 both also write to P disk**

RAID 5: High I/O Rate



Garcia, Yan

"And in Conclusion…"

- **Great Idea: Redundancy to Get Dependability**
  - Spatial (extra hardware) and Temporal (retry if error)
- **Reliability: MTTF, Annualized Failure Rate (AFR), and FIT**
- **Availability: % uptime (MTTF/MTTF+MTTR)**
- **Memory**
  - Hamming distance 2: Parity for Single Error Detect
  - Hamming distance 3: Single Error Correction Code + encode bit position of error
- **Treat disks like memory, except you know when a disk has failed—erasure makes parity an Error Correcting Code**
- **RAID-2, -3, -4, -5 (and -6, -10): Interleaved data and parity**