15-853:Algorithms in the Real World

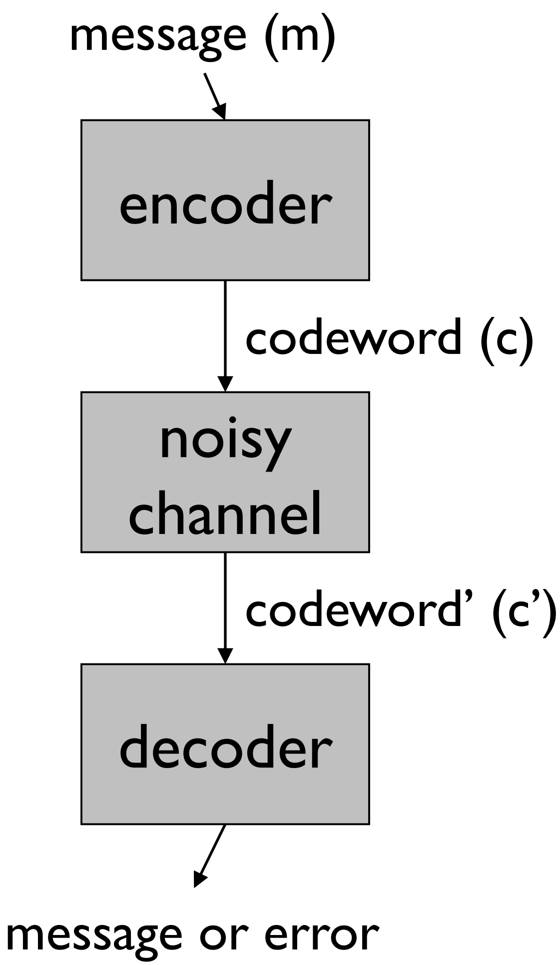
Error Correcting Codes (cont..)

Scribe volunteers: ?

Announcement:

Scribe notes template and instructions on the course webpage

General Model



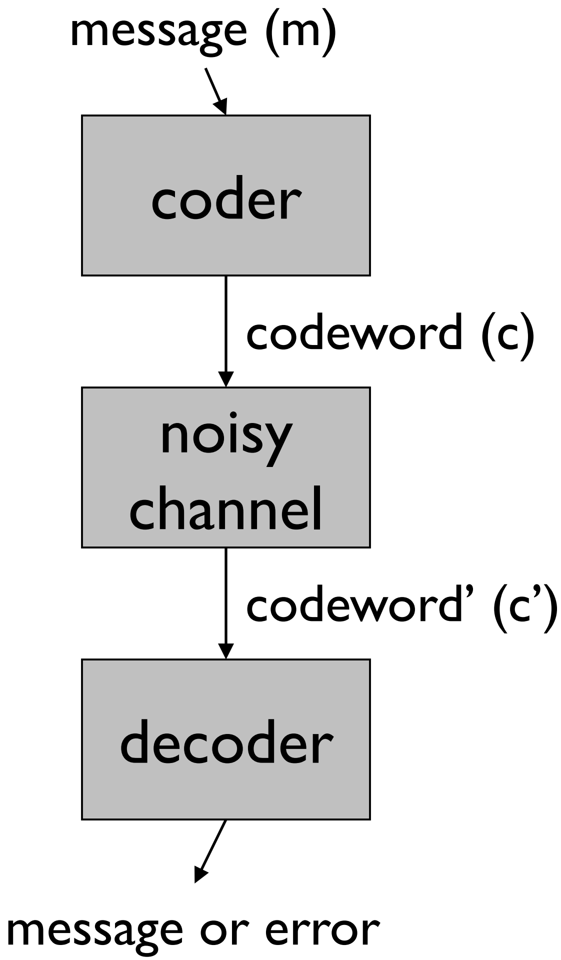
"Noise" introduced by the channel:

- changed fields in the codeword vector (e.g. a flipped bit).
  - Called **errors**
- missing fields in the codeword vector (e.g. a lost byte).
  - Called **erasures**

How the decoder deals with errors and/or erasures?

- **detection** (only needed for errors)
- **correction**

Block Codes



Each message and codeword is of fixed size

Σ = codeword alphabet

k $=|m|$ **n** $= |c|$ **q** $= |\Sigma|$

C = "code" = set of codewords

C $\subseteq \Sigma^n$ (codewords)

$\boldsymbol{\Delta(x,y)}$ = number of positions s.t. $x_i \neq y_i$

d $= \min\{\Delta(x,y) : x,y \in C, x \neq y\}$

Code described as: $\mathbf{(n,k,d)_q}$

Role of Minimum Distance

Theorem:

A code C with minimum distance “d” can:

1. detect any (d-1) errors
2. recover any (d-1) erasures
3. correct any <write> errors

Stated another way:

For s-bit error detection $d \geq s + 1$

For s-bit error correction $d \geq 2s + 1$

To correct a erasures and b errors if

$$d \geq a + 2b + 1$$

Next we will see

an application of erasure codes in

today's large-scale data storage systems

Large-scale distributed storage systems

1000s of interconnected servers

100s of petabytes of data

- Commodity components
- Software issues, power failures, maintenance shutdowns

Large-scale distributed storage systems

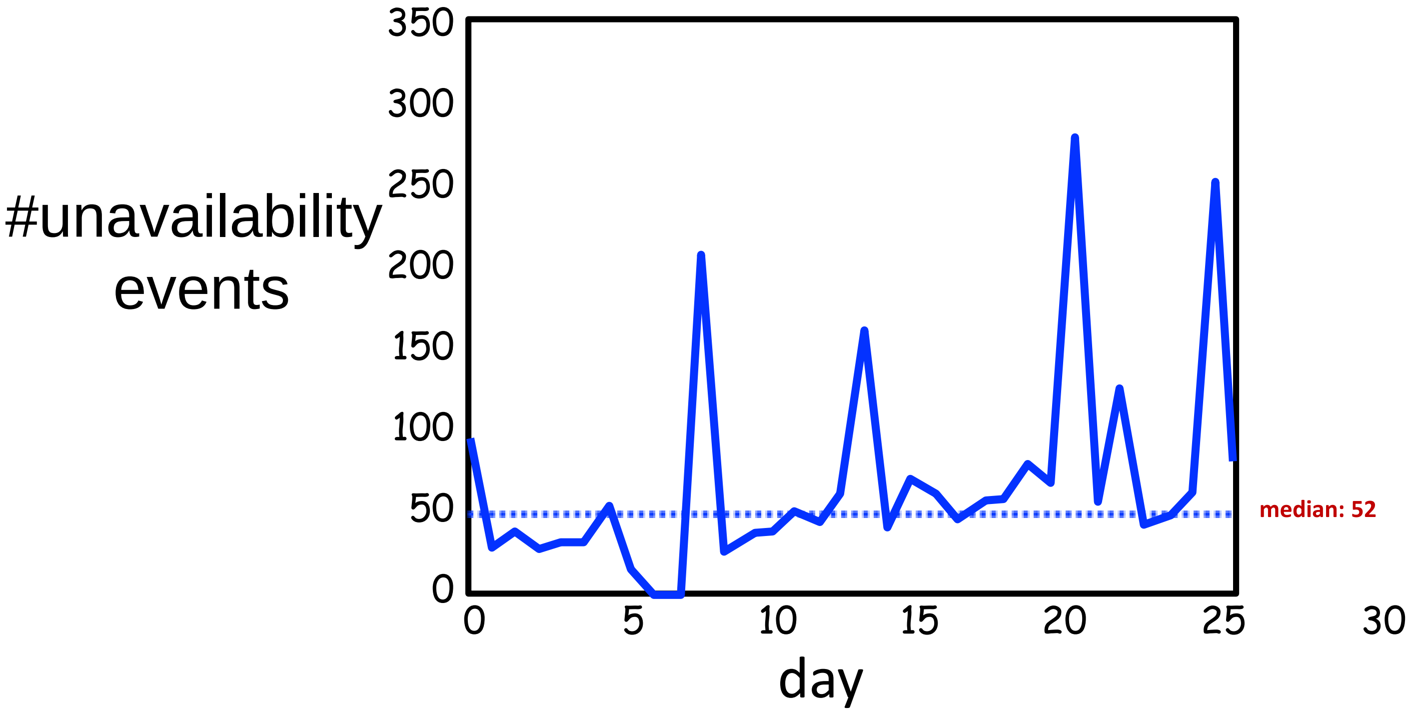
1000s of interconnected servers

Unavailabilities are the norm rather than the exception

- Commodity components
- Software issues, power failures, maintenance shutdowns

Facebook analytics cluster in production: unavailability statistics

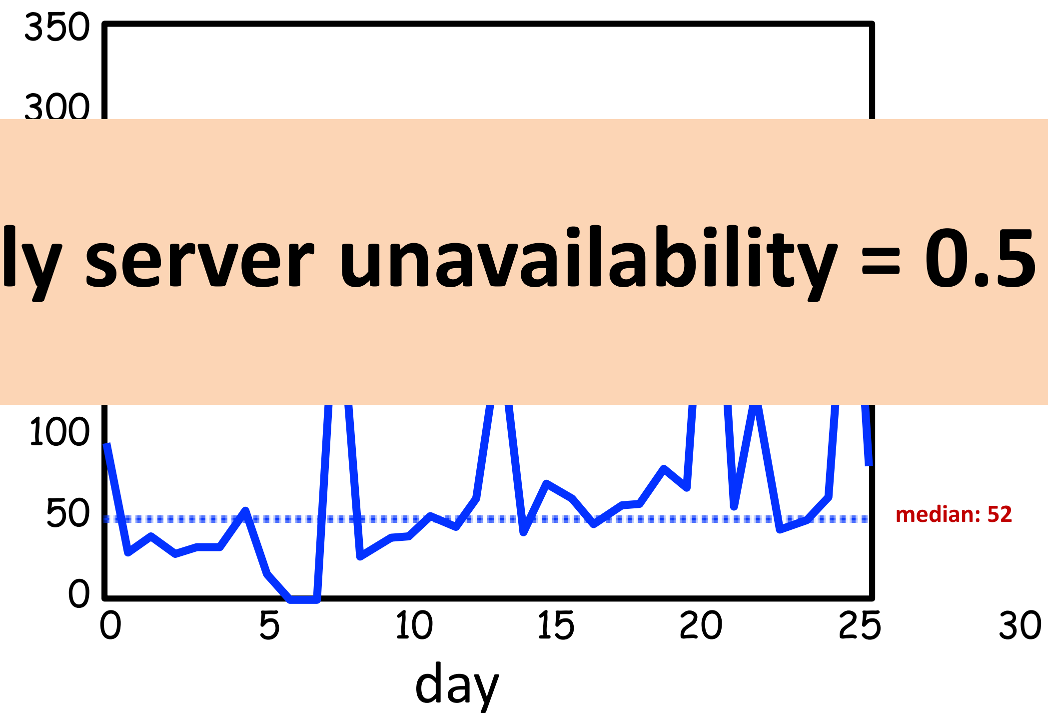
- Multiple thousands of servers
- Unavailability event: server unresponsive for > 15 min



[Rashmi, Shah, Gu, Kuang, Borthakur, Ramchandran, USENIX HotStorage 2013 and ACM SIGCOMM 2014]

Facebook analytics cluster in production: unavailability statistics

- Multiple thousands of servers
- Unavailability event: server unresponsive for > 15 min



Daily server unavailability = 0.5 - 1%

[Rashmi, Shah, Gu, Kuang, Borthakur, Ramchandran, USENIX HotStorage 2013 and ACM SIGCOMM 2014]

Servers unavailable

↓

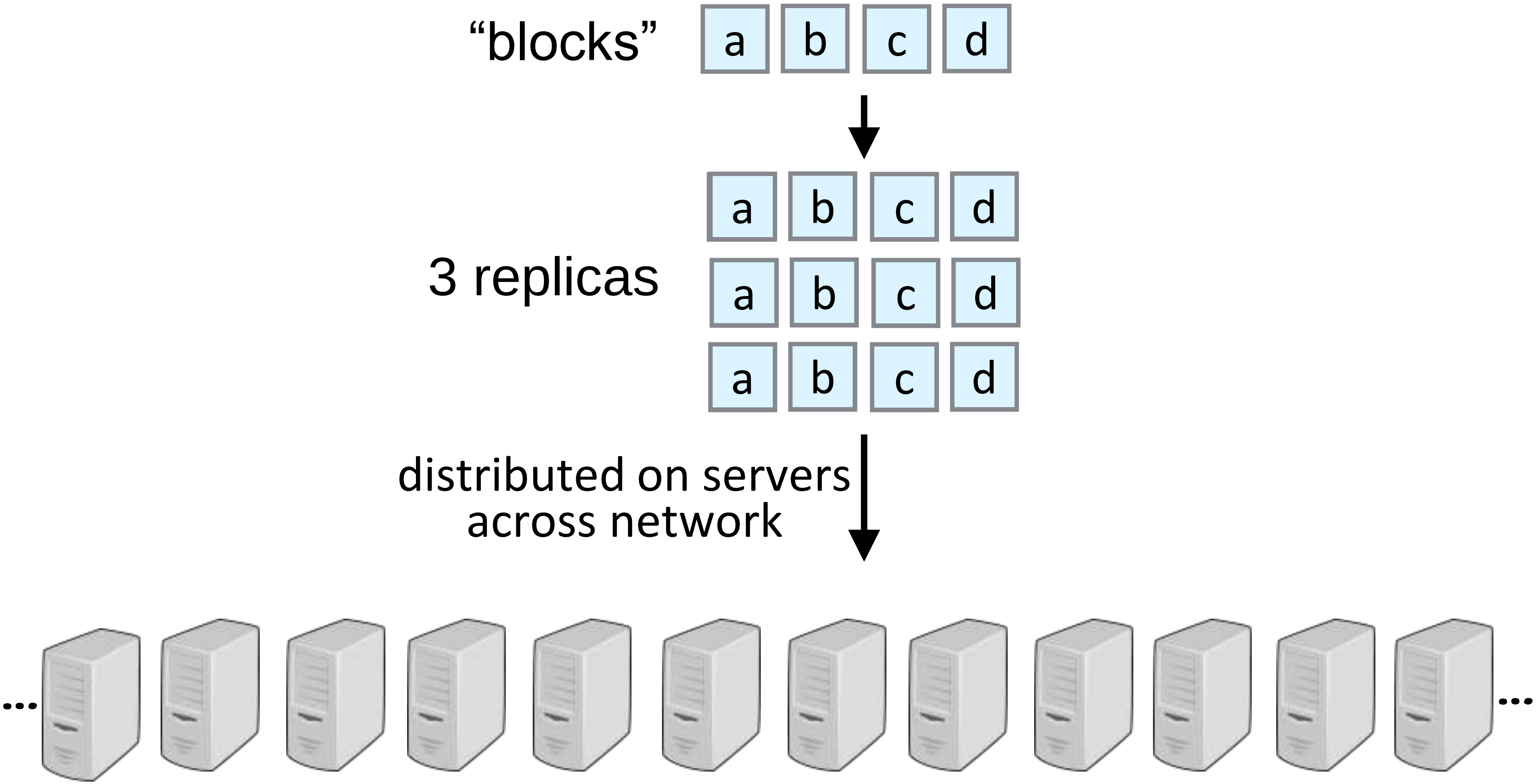
Data inaccessible

↓ Applications cannot wait,
Data cannot be lost

Data needs to be stored in a redundant fashion

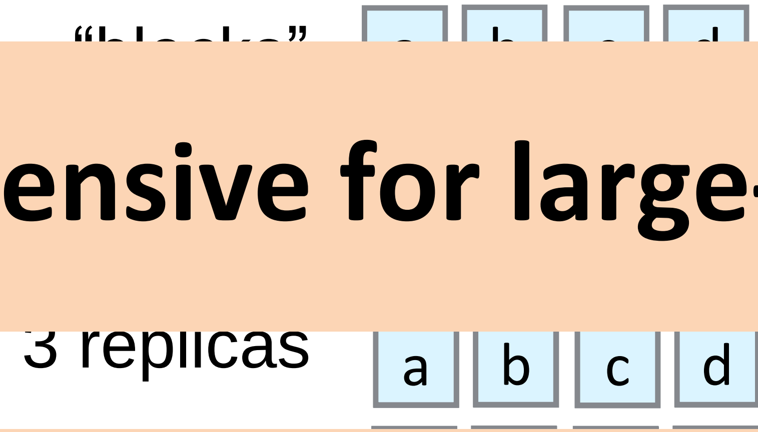
Traditional approach: Replication

- Storing **multiple copies** of data: Typically 3x-replication



Traditional approach: Replication

- Storing **multiple copies** of data: Typically 3x-replication

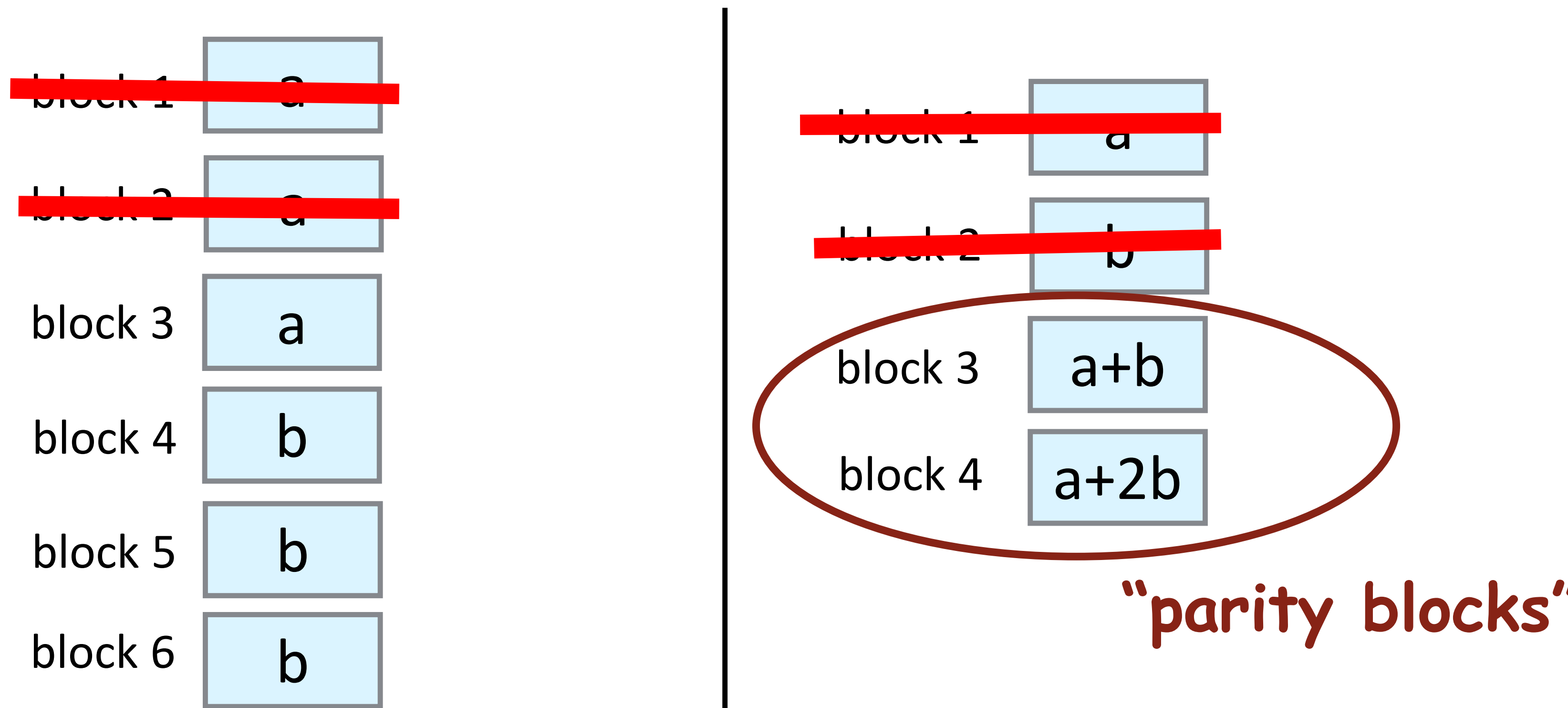


Too expensive for large-scale data

Better alternative: sophisticated codes

Two data blocks to be stored: a and b

Tolerate any 2 failures



3-replication

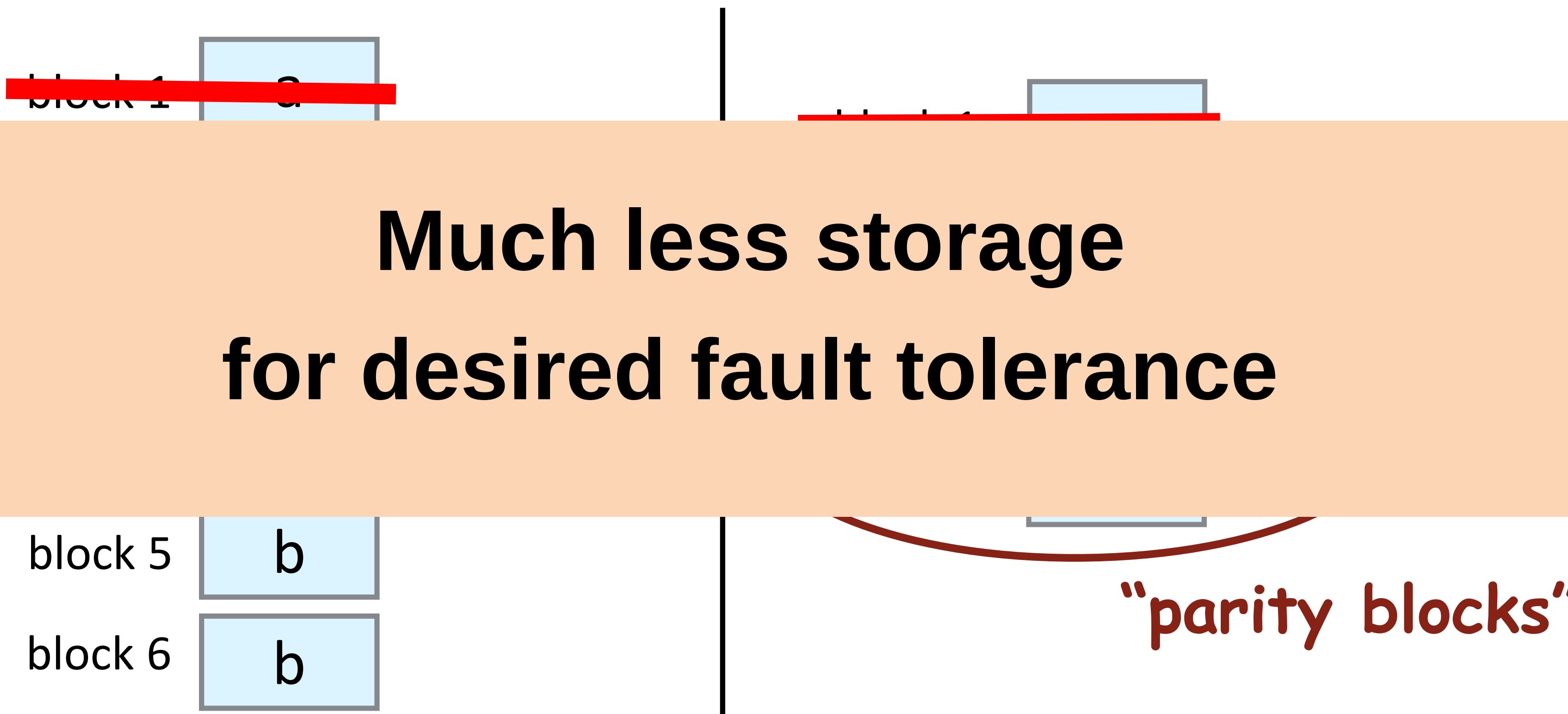
Storage overhead = 3x

Erasure code

Storage overhead = 2x

Two data blocks to be stored: a and b

Tolerate any 2 failures



Much less storage

for desired fault tolerance

3-replication

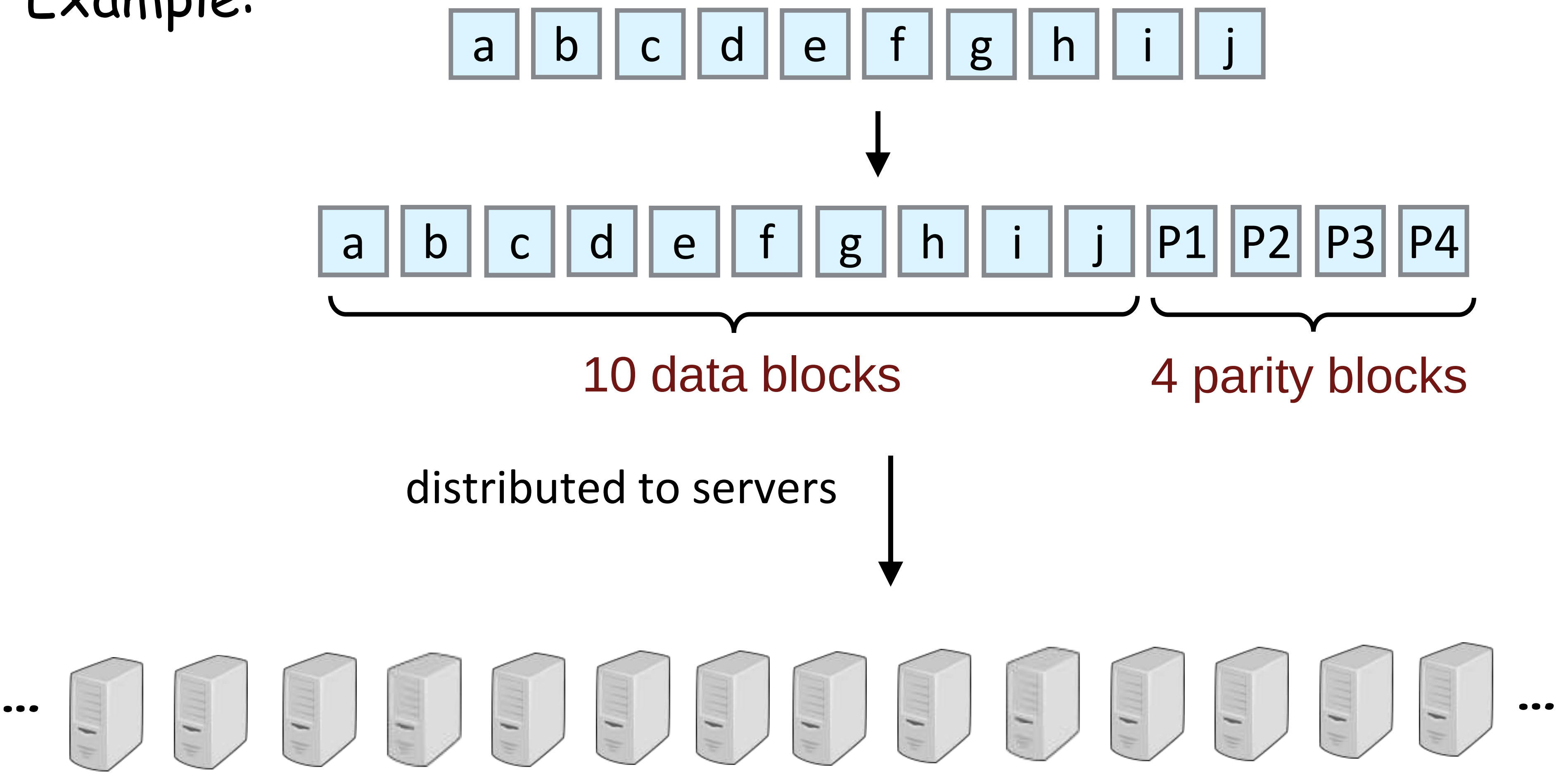
Storage overhead = 3x

Erasure code

Storage overhead = 2x

Erasure codes: how are they used in distributed storage systems?

Example:



Almost all large-scale storage systems today employ erasure codes

Facebook, Google, Amazon, Microsoft...

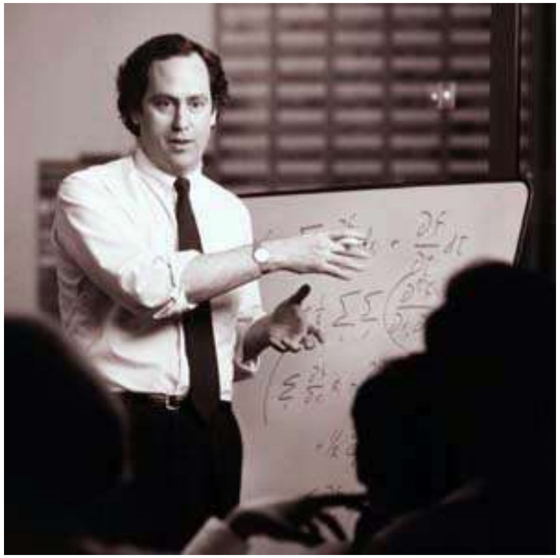
"Considering trends in data growth & datacenter hardware, we foresee HDFS **erasure coding** being an **important feature in years to come**"

- Cloudera Engineering (September, 2016)

Error Correcting Multibit Messages

We will first discuss **Hamming Codes**

Named after Richard Hamming (1915-1998), a pioneer in error-correcting codes and computing in general.



Error Correcting Multibit Messages

We will first discuss **Hamming Codes**

Codes are of form: $(2^r-1, 2^r-1 - r, 3)$ for any r > 1

e.g. (3,1,3), (7,4,3), (15,11,3), (31, 26, 3), …

which correspond to 2, 3, 4, 5, … "parity bits" (i.e. n-k)

Question: Error detection and correction capability?

(Can detect 2-bit errors, or correct 1-bit errors.)

The high-level idea is to "localize" the error.

Hamming Codes: Encoding

r = 4

Localizing error to top or bottom half 1xxx or 0xxx

m_{15}	m_{14}	m_{13}	m_{12}	m_{11}	m_{10}	m_9	p_8	m_7	m_6	m_5		m_3			p_0

$$p_8 = m_{15} \oplus m_{14} \oplus m_{13} \oplus m_{12} \oplus m_{11} \oplus m_{10} \oplus m_9$$

Localizing error to x1xx or x0xx

m_{15}	m_{14}	m_{13}	m_{12}	m_{11}	m_{10}	m_9	p_8	m_7	m_6	m_5	p_4	m_3			p_0

$$p_4 = m_{15} \oplus m_{14} \oplus m_{13} \oplus m_{12} \oplus m_7 \oplus m_6 \oplus m_5$$

Localizing error to xx1x or xx0x

m_{15}	m_{14}	m_{13}	m_{12}	m_{11}	m_{10}	m_9	p_8	m_7	m_6	m_5	p_4	m_3	p_2		p_0

$$p_2 = m_{15} \oplus m_{14} \oplus m_{11} \oplus m_{10} \oplus m_7 \oplus m_6 \oplus m_3$$

Localizing error to xxx1 or xxx0

m_{15}	m_{14}	m_{13}	m_{12}	m_{11}	m_{10}	m_9	p_8	m_7	m_6	m_5	p_4	m_3	p_2	p_1	p_0

$$p_1 = m_{15} \oplus m_{13} \oplus m_{11} \oplus m_9 \oplus m_7 \oplus m_5 \oplus m_3$$

Hamming Codes: Decoding

m_{15}	m_{14}	m_{13}	m_{12}	m_{11}	m_{10}	m_9	p_8	m_7	m_6	m_5	p_4	m_3	p_2	p_1	p_0

We don't need p_0, so we have a (15,11,?) code.

After transmission, we generate

$b_8 = p_8 \oplus m_{15} \oplus m_{14} \oplus m_{13} \oplus m_{12} \oplus m_{11} \oplus m_{10} \oplus m_9$

$b_4 = p_4 \oplus m_{15} \oplus m_{14} \oplus m_{13} \oplus m_{12} \oplus m_7 \oplus m_6 \oplus m_5$

$b_2 = p_2 \oplus m_{15} \oplus m_{14} \oplus m_{11} \oplus m_{10} \oplus m_7 \oplus m_6 \oplus m_3$

$b_1 = p_1 \oplus m_{15} \oplus m_{13} \oplus m_{11} \oplus m_9 \oplus m_7 \oplus m_5 \oplus m_3$

With no errors, these will all be zero

With one error $b_8b_4b_2b_1$ gives us the error location.

e.g. **0100** would tell us that $\mathbf{p_4}$ is wrong, and
1100 would tell us that $\mathbf{m_{12}}$ is wrong

Hamming Codes

Can be generalized to any power of 2

- $n = 2^r - 1$ (15 in the example)
- $(n-k) = r$ (4 in the example)
- Can correct one error
- $d \geq 3$ (since we can correct one error)
- Gives $(2^r-1, 2^r-1-r, 3)$ code

(We will later see an easy way to prove the minimum distance)

Extended Hamming code

- Add back the parity bit at the end
- Gives $(2^r, 2^r-1-r, 4)$ code
- Can still correct one error, but now can detect 3

A Lower bound on parity bits: Hamming bound

How many nodes in hypercube do we need so that d = 3?

Each of 2^k codewords eliminates n neighbors plus itself, i.e. n+1

$$2^n \geq (n+1)2^k$$

$$n \geq k + \log_2(n+1)$$

$$n \geq k + \lceil \log_2(n+1) \rceil$$

In above Hamming code, $15 \geq 11 + \lceil \log_2(15+1) \rceil = 15$.

Hamming Codes are called **perfect codes** since they match the lower bound exactly.

A Lower bound on parity bits: Hamming bound

What about fixing 2 errors (i.e. d=5)?

Each of the 2^k codewords eliminates itself, its neighbors and its neighbors' neighbors, giving:

<board>

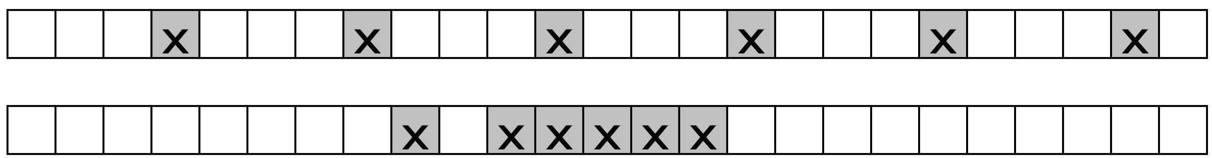
Generally to correct s errors:

$$n \geq k + \log_2\left(1 + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{s}\right)$$

Lower Bounds: a side note

The lower bounds assume arbitrary placement of bit errors.

In practice errors are likely to have patterns: maybe evenly spaced, or clustered:



Can we do better if we assume **regular errors**?

We will come back to this later when we talk about **Reed-Solomon** codes. This is a big reason why Reed-Solomon codes are used much more than Hamming-codes.

Q:

If no structure in the code, how would one perform encoding?

<board>

Gigantic lookup table!

If no structure in the code, encoding is highly inefficient.

A common kind of structure added is **linearity**

Linear Codes

If Σ is a field, then Σ^n is a vector space

Definition: C is a linear code if it is a linear subspace of Σ^n of dimension k.

This means that there is a set of k independent vectors $v_i \in \Sigma^n$ $(1 \leq i \leq k)$ that span the subspace.

i.e. every codeword can be written as:

$$c = a_1 v_1 + a_2 v_2 + \ldots + a_k v_k \quad \text{where } a_i \in \Sigma$$

"Basis (or spanning) Vectors"

Some Properties of Linear Codes

1. Linear combination of two codewords is a codeword.

<board>

2. Minimum distance (d) = weight of least weight (non-zero) codewords

<Write proof>

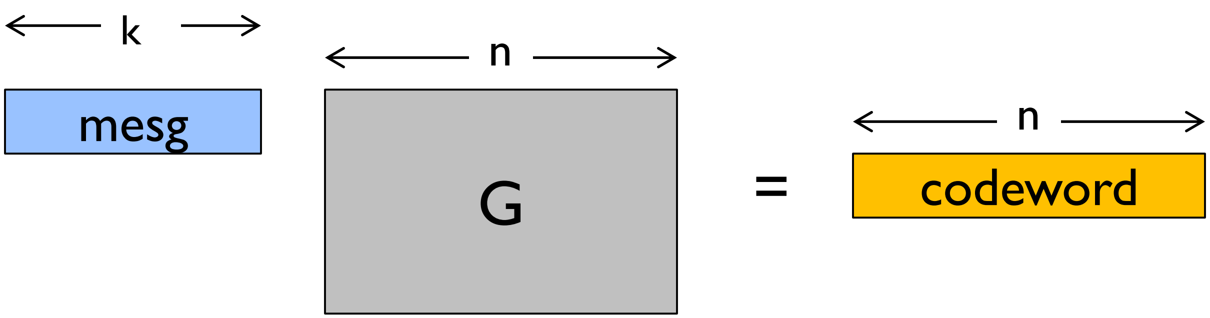
Generator and Parity Check Matrices

3. Every linear code has two matrices associated with it.

1. Generator Matrix:

A k x n matrix **G** such that: $C = \{ xG \mid x \in \Sigma^k \}$

Made from stacking the spanning vectors

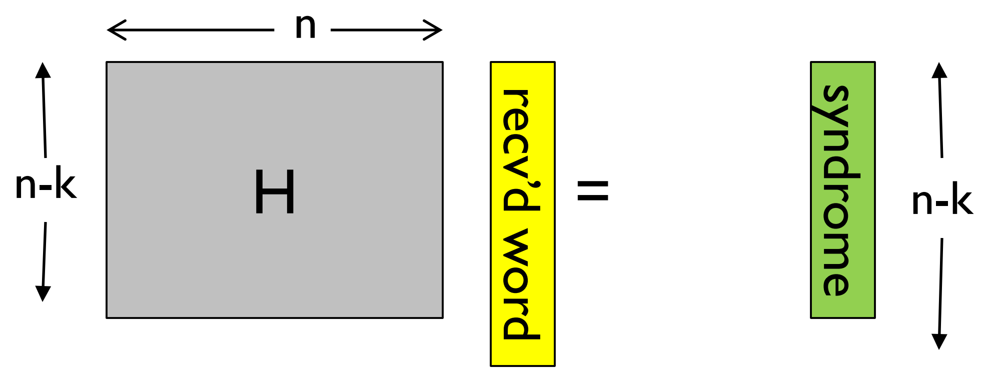


Generator and Parity Check Matrices

2. Parity Check Matrix:

An (n – k) x n matrix **H** such that: $C = \{y \in \Sigma^n \mid Hy^T = 0\}$

(Codewords are the null space of H.)



if syndrome = 0, received word = codeword

else have to use syndrome to get back codeword ("decode")

Advantages of Linear Codes

- Encoding is efficient (vector-matrix multiply)
- Error detection is efficient (vector-matrix multiply)
- **Syndrome** (Hy^T) has error information
- How to decode? In general, have q^{n-k} sized table for decoding (one for each syndrome).
  Useful if n-k is small, else want other approaches.

Linear Codes

Basis vectors for the $(7,4,3)_2$ Hamming code:

		m_7	m_6	m_5	p_4	m_3	p_2	p_1
v_1	=	1	0	0	1	0	1	1
v_2	=	0	1	0	1	0	1	0
v_3	=	0	0	1	1	0	0	1
v_4	=	0	0	0	0	1	1	1

Another way to see that d = 3 for Hamming codes?

What is the least Hamming weight among non-zero codewords?

In the next class we will continue studying linear codes starting with
additional properties of generator and parity check matrices and relationship between them