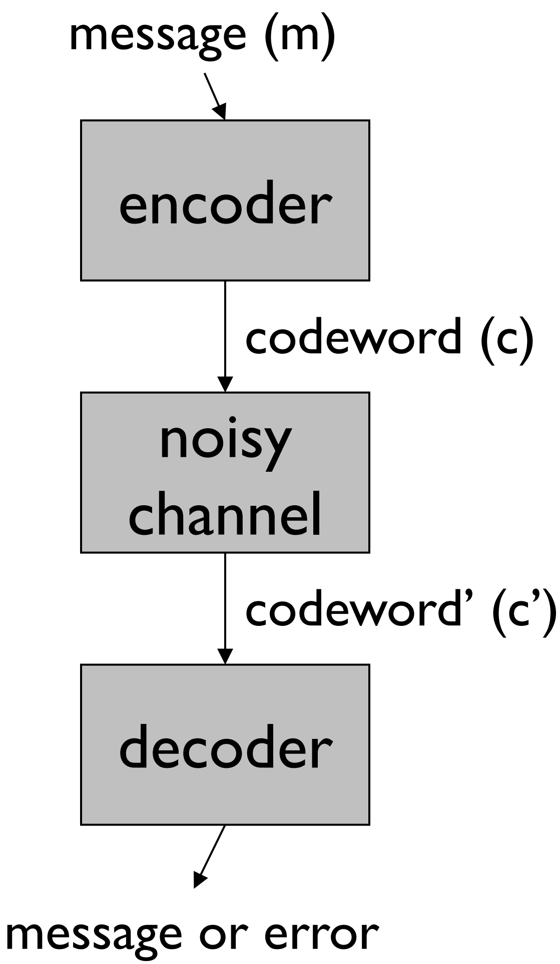
Recap: Notation for Block Codes



- Each message and codeword is of fixed size
- Notation:

k = |m| length of the message — "dimension of the code"

n = |c| length of the codeword — "length of the code"

C = "code" = set of codewords

Simple Examples

3-Repetition code: k=1, n=3

Message		Codeword
0	->	000
1	->	111

- How many **erasures** can be recovered?
- How many **errors** can be **detected**?
- Up to how many **errors** can be **corrected**?

Errors are much harder to deal with than erasures.

Why?

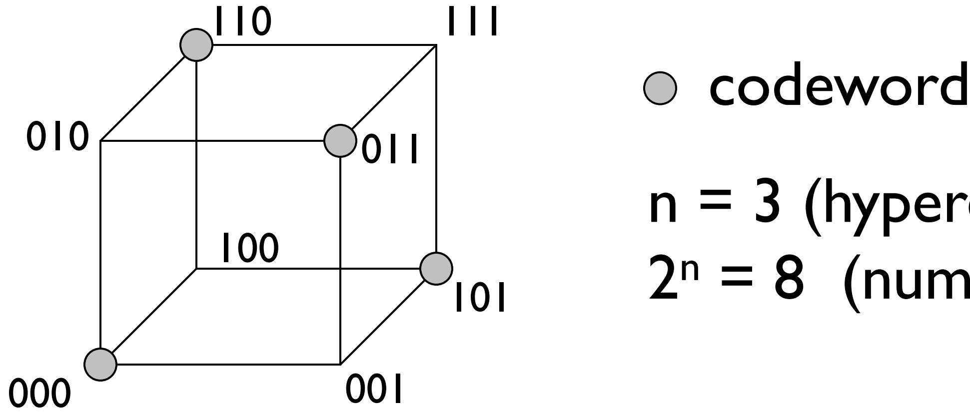
Need to find out **where** the errors are!

Simple Examples

Single parity check code: k=2, n=3

Message		Codeword
00	->	000
01	->	011
10	->	101
11	->	110

Consider codewords as vertices on a hypercube.

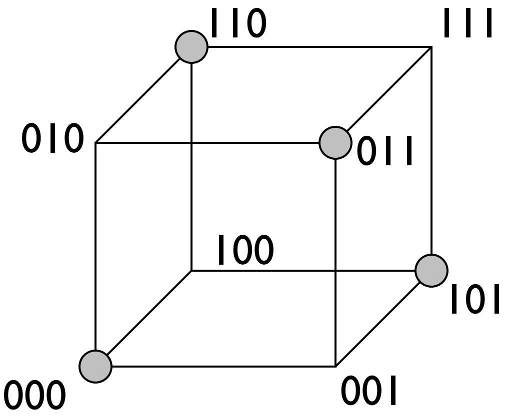


n = 3 (hypercube dimensionality)

$2^n = 8$ (number of nodes)

Simple Examples

Single parity check code: k=2, n=3



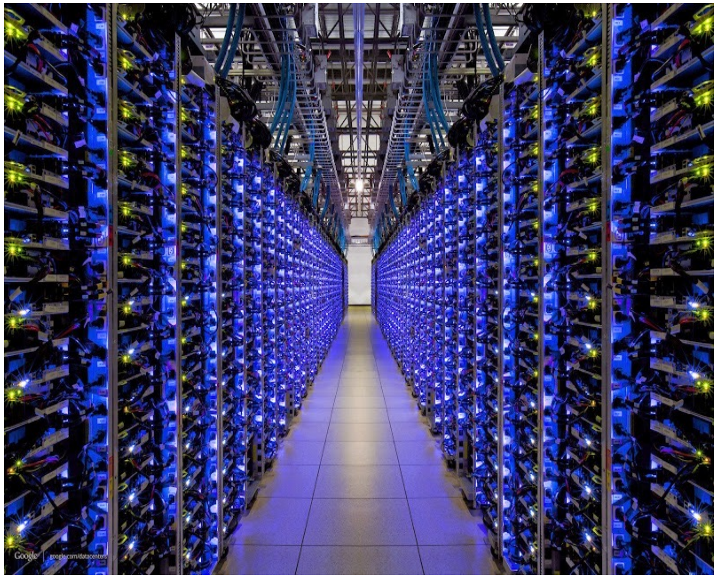
- How many **erasures** can be recovered?
- How many **errors** can be **detected**?
- Up to how many **errors** can be **corrected**?

Systematic codes

Definition: A **Systematic code** is one in which the message symbols appear in the codeword in uncoded form

message	codeword
000	**000**000
001	**001**011
010	**010**101
011	**011**110
100	**100**110
101	**101**101
110	**110**011
111	**111**000

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

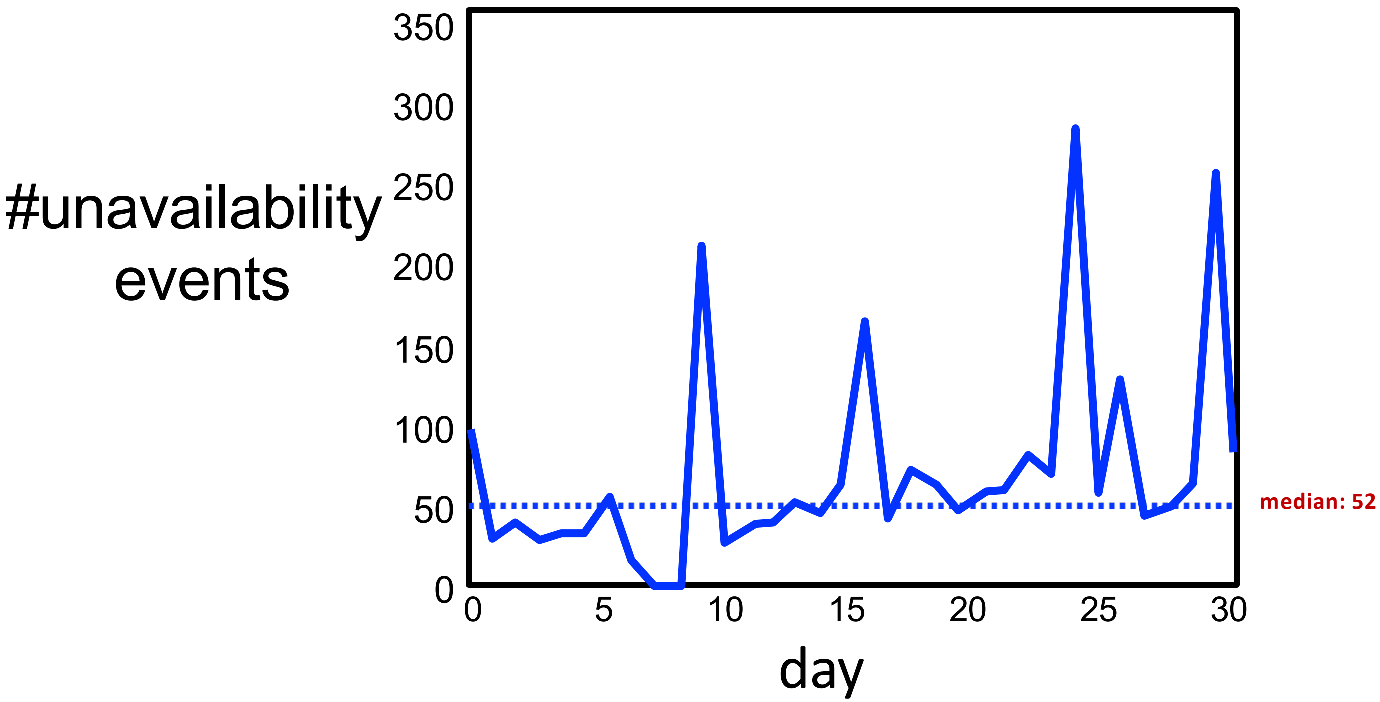
1000s of interconnected servers

Unavailabilities are the norm rather than the exception

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

Facebook analytics cluster in production: unavailability statistics

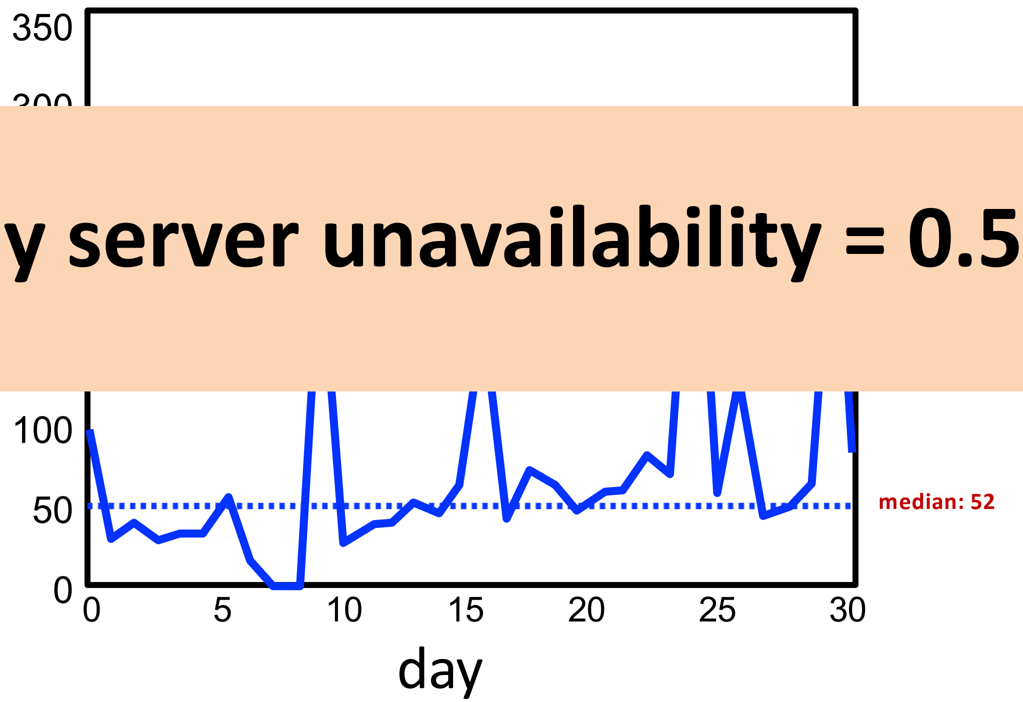
- Multiple thousands of servers
- Unavailability event: server unresponsive for > 15 min



[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

↓

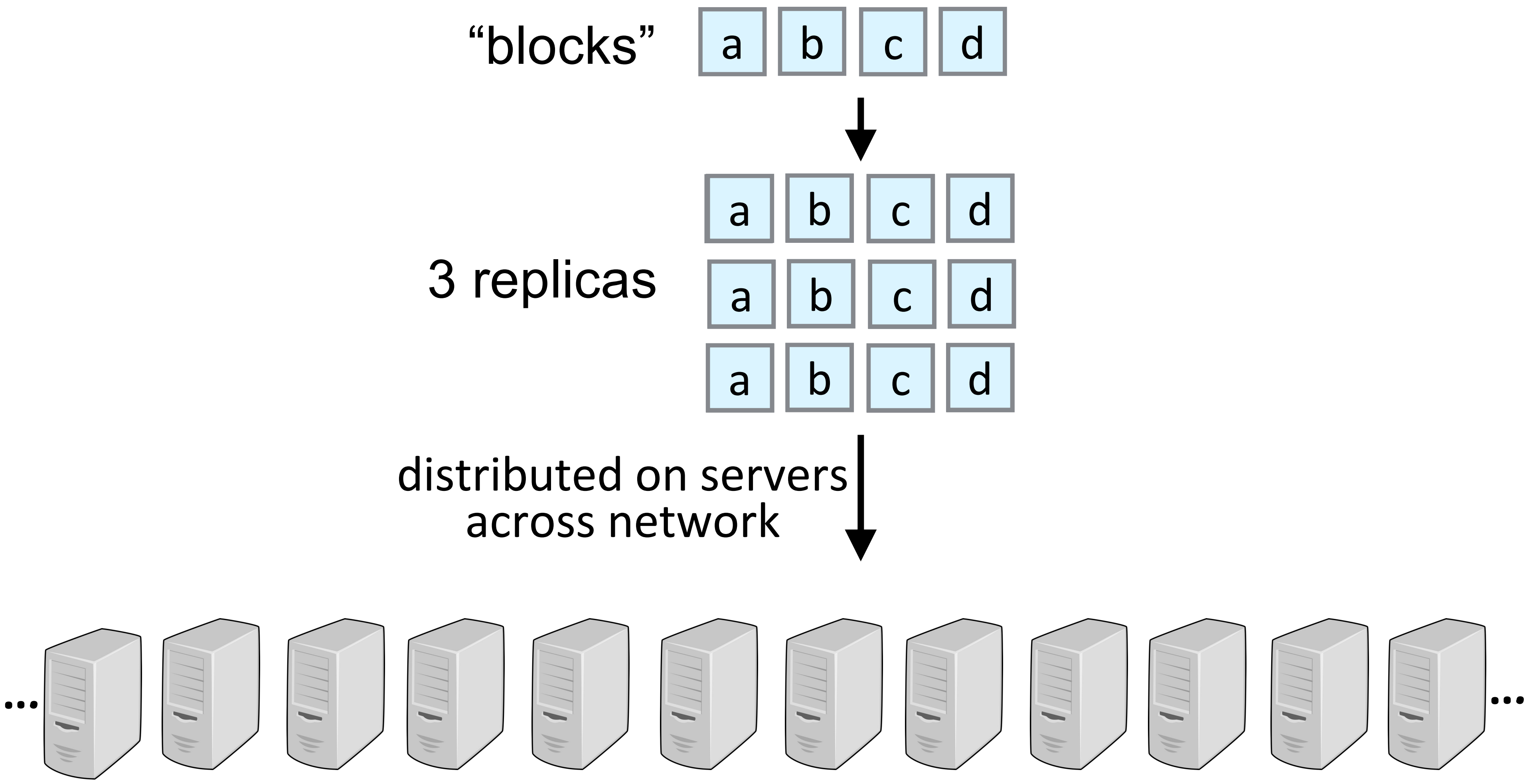
Data inaccessible

↓ 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

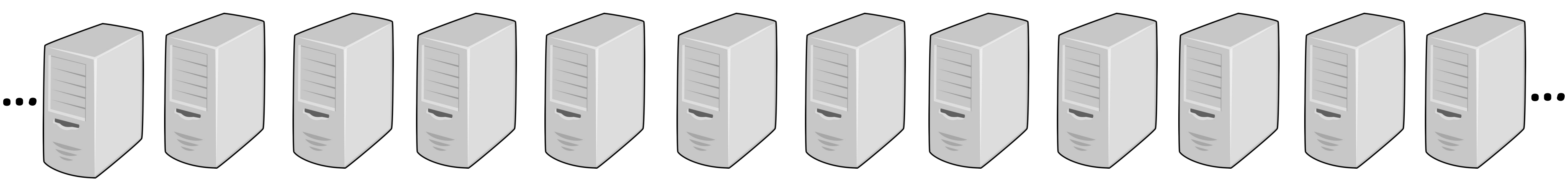
- Storing **multiple copies** of data: Typically 3x-replication

3 replicas

a	b	c	d

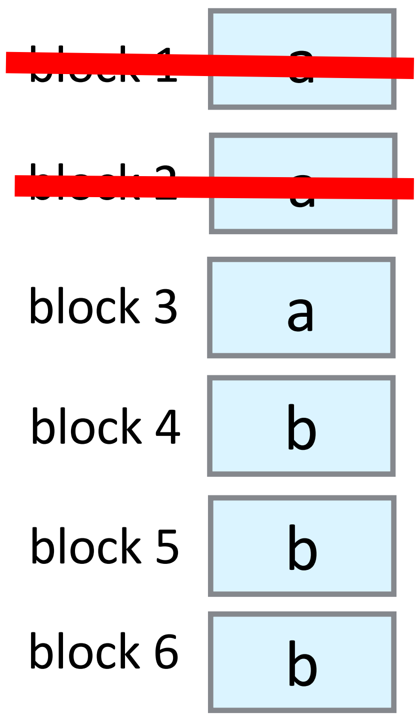
Too expensive for large-scale data

Better alternative: codes!



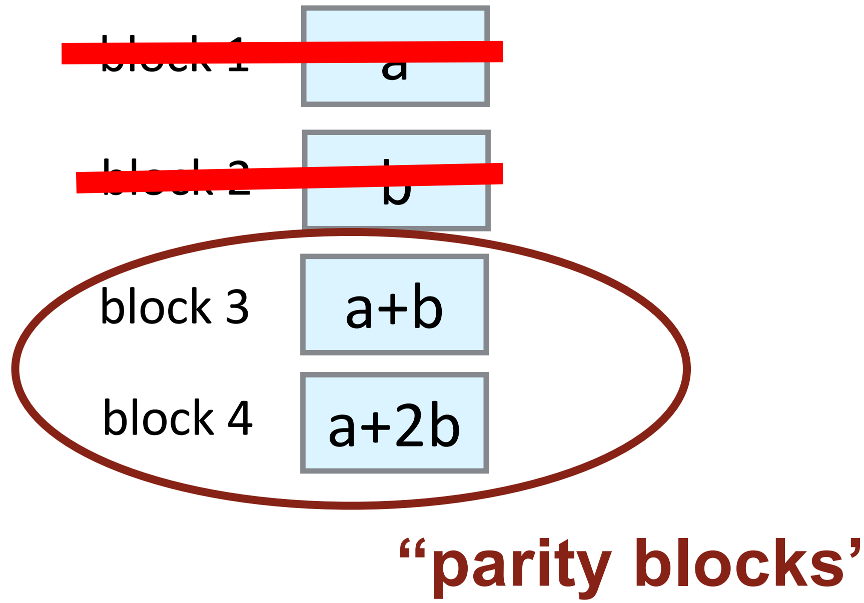
Two data blocks to be stored: a and b

Tolerate any 2 failures



3-replication

Storage overhead = 3x

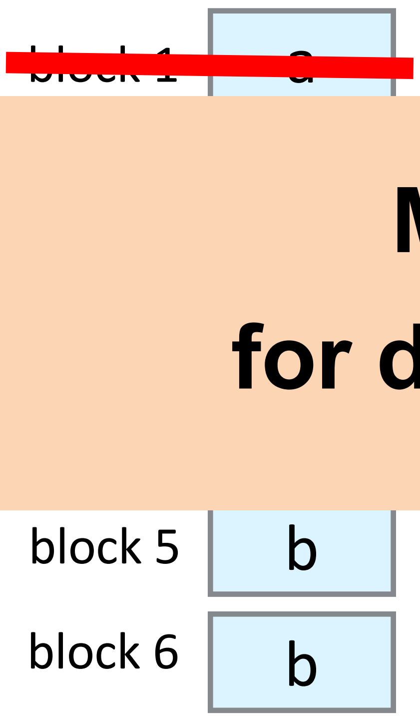


Erasure code

Storage overhead = 2x

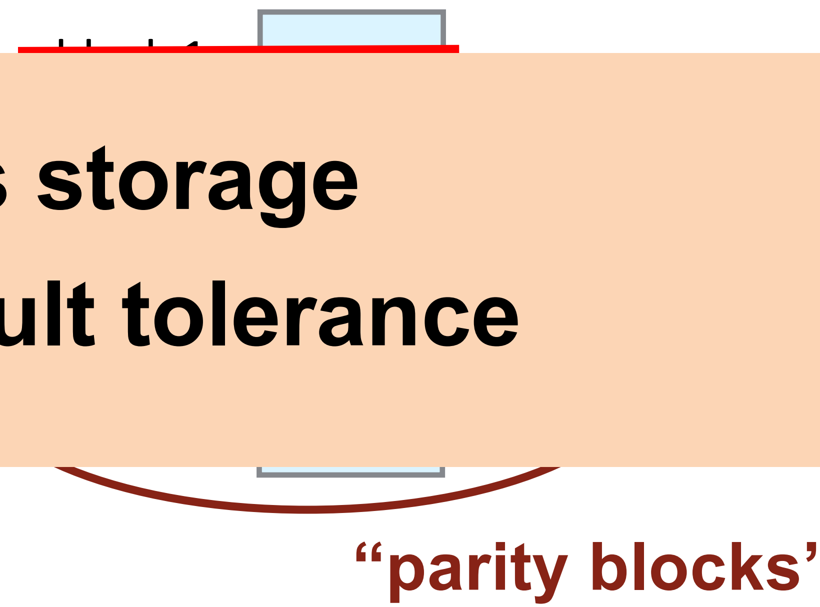
Two data blocks to be stored: a and b

Tolerate any 2 failures



3-replication

Storage overhead = 3x



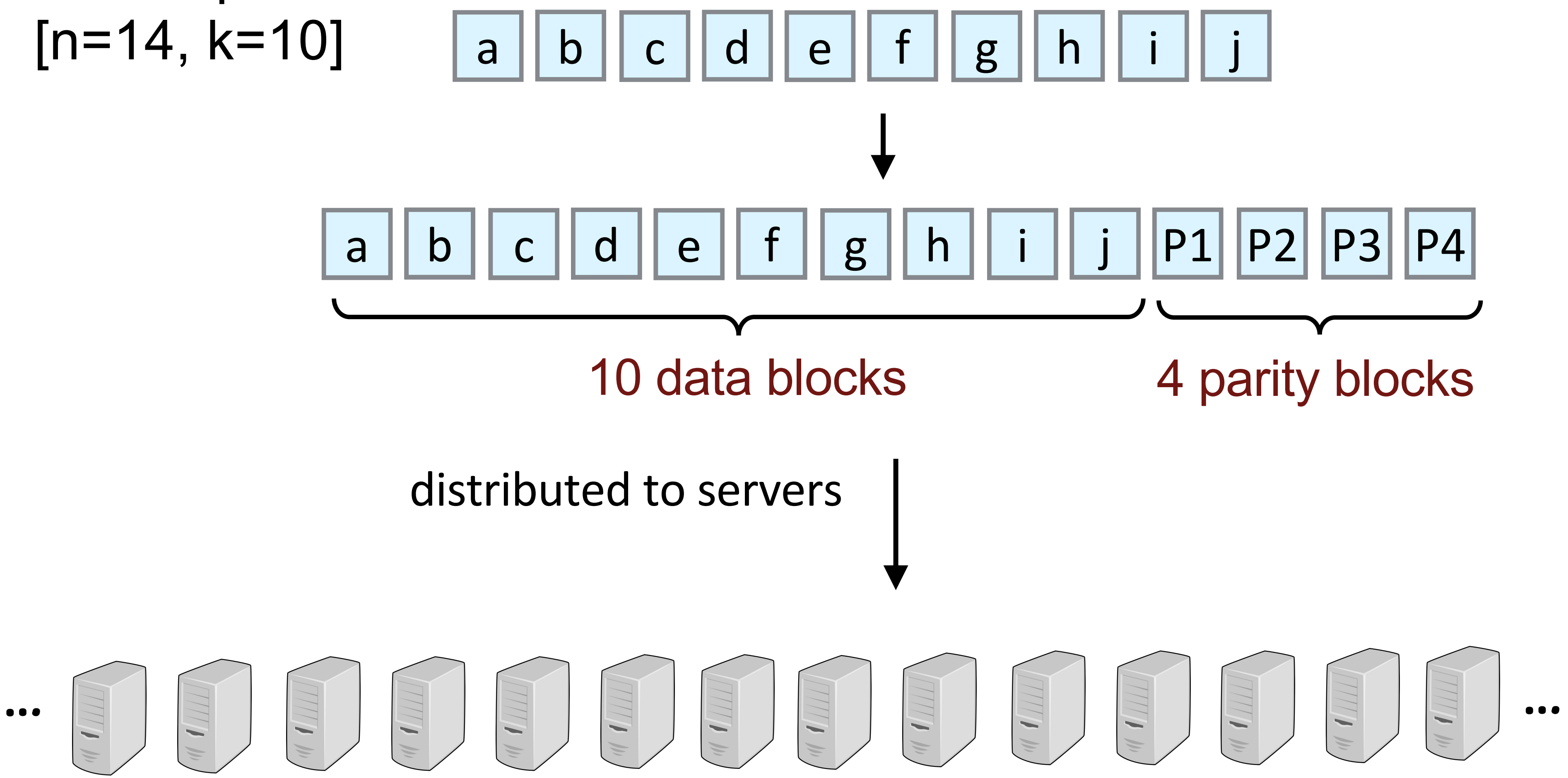
Erasure code

Storage overhead = 2x

Much less storage
for desired fault tolerance

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

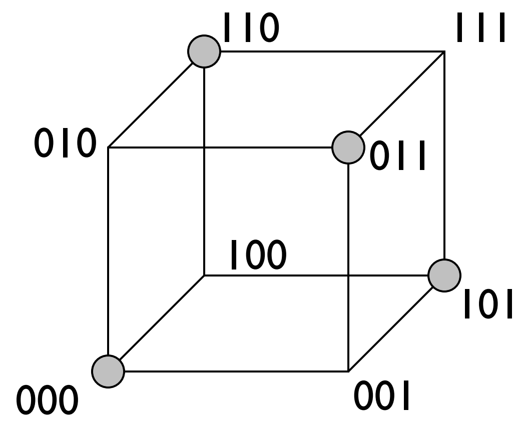
Example:
[n=14, k=10]



Almost all large-scale storage systems today employ erasure codes for most of the stored data

Google, Amazon, Microsoft, Meta, IBM, ...

Simple Examples



Single parity check code: k=2, n=3

- How many **erasures** can be recovered?
- How many **errors** can be **detected**?
- Up to how many **errors** can be **corrected**?

Erasure correction = 1, error detection = 1, error correction = 0

Cannot even correct single error. Why?

Codewords are too "close by"

Let's formalize this notion of distance..

Block Codes

Notion of distance between codewords: **Hamming distance**

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

Minimum distance of a code

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

Code described as: **(n, k, d)$_q$** ←

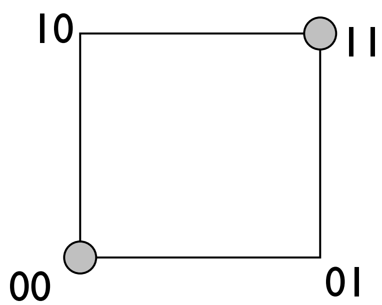
- Σ = alphabet
- **q** = $|\Sigma|$ = alphabet size
- **C** $\subseteq \Sigma^n$ (codewords)

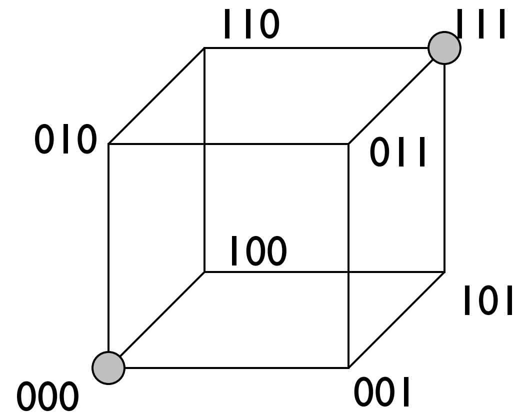
Question:

What alphabet did we use so far?

Error Correcting One Bit Messages

How many bits do we need to correct a one bit error on a one bit message?





2 bits
0 -> 00, 1-> 11
(n=2,k=1,d=2)

3 bits
0 -> 000, 1-> 111
(n=3,k=1,d=3)

In general need $d \geq 3$ to correct one error. Why?

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 $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors

Intuition: <board>

Stated another way:

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

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

Desired Properties

We look for codes with the following properties:

1. Good rate: k/n should be high (low overhead)
2. Good distance: d should be large (good error correction)
3. Small block size k (helps with latency)
4. Fast encoding and decoding
5. Others: want to handle bursty/random errors, local decodability, ...

Q:

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

Gigantic lookup table!

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

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

(Slight detour into number theory)

Groups

A **Group** (G,*,I) is a set *G* with operator * such that:

1. **Closure**. For all $a,b \in G$, $a * b \in G$
2. **Associativity.** For all $a,b,c \in G$, $a*(b*c) = (a*b)*c$
3. **Identity.** There exists $I \in G$, such that for all $a \in G$, $a*I=I*a=a$
4. **Inverse.** For every $a \in G$, there exist a unique element $b \in G$, such that $a*b=b*a=I$

An **Abelian or Commutative Group** is a Group with the additional condition

5. **Commutativity.** For all $a,b \in G$, $a*b=b*a$

Examples of groups

Q: Examples?

- Integers, Reals or Rationals with Addition
- The nonzero Reals or Rationals with Multiplication
- Non-singular n x n real matrices with
  Matrix Multiplication
- Permutations over n elements with composition
  [0→1, 1→2, 2→0] o [0→1, 1→0, 2→2] = [0→0, 1→2, 2→1]

Often we will be concerned with **finite groups**, I.e., ones with a finite number of elements.

Groups based on modular arithmetic

The group of positive integers modulo a prime *p*

$Z_p^* \equiv \{1, 2, 3, \ldots, p\text{-}1\}$ $\quad\quad$ $*_p \equiv$ multiplication modulo p

Denoted as: $(Z_p^*, *_p)$

Required properties

1. Closure. Yes.
2. Associativity. Yes.
3. Identity. 1.
4. Inverse. Yes. (HW)

Example: $Z_7^* = \{1,2,3,4,5,6\}$

$1^{-1} = 1,\ 2^{-1} = 4,\ 3^{-1} = 5,\ 6^{-1} = 6$

Fields

A **Field** is a set of elements F with binary operators * and + such that

1. (F, +) is an **abelian group**
2. (F \ I_+, *) is an **abelian group**
   the "multiplicative group"
3. **Distribution**: a*(b+c) = a*b + a*c
4. **Cancellation**: $a*I_+ = I_+$

Example: The reals and rationals with + and * are fields.

The **order (or size)** of a field is the number of elements.

A field of finite order is a **finite field**.

Finite Fields

$\mathbb{Z}_p$ (p prime) with + and * mod p, is a **finite** field.

1. $(\mathbb{Z}_p, +)$ is an **abelian group** (0 is identity)
2. $(\mathbb{Z}_p \setminus 0, *)$ is an **abelian group** (1 is identity)
3. **Distribution**: a*(b+c) = a*b + a*c
4. **Cancellation**: a*0 = 0

We denote this by $\mathbb{F}_p$ or GF(p)

Are there other finite fields?

What about ones that fit nicely into bits, bytes and words (i.e with 2^k elements)?