A Performance Evaluation of Open Source Erasure Codes for Storage Applications

James S. Plank
Catherine D. Schuman
(Tennessee)

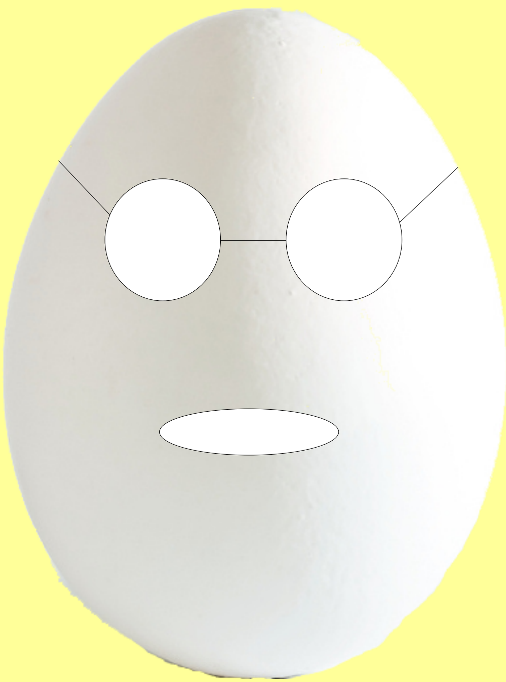
Jianqiang Luo
Lihao Xu
(Wayne State)

Zooko Wilcox-O'Hearn Allmydata

Usenix FAST
February 27, 2009

My Perspective on Storage

A code $\mathcal{C}$ over $\mathbb{F}^b_q$ is $\mathbb{F}_q$-linear if $\mathcal{C}$ is a vector space over $\mathbb{F}_q$...



Coding Theorist

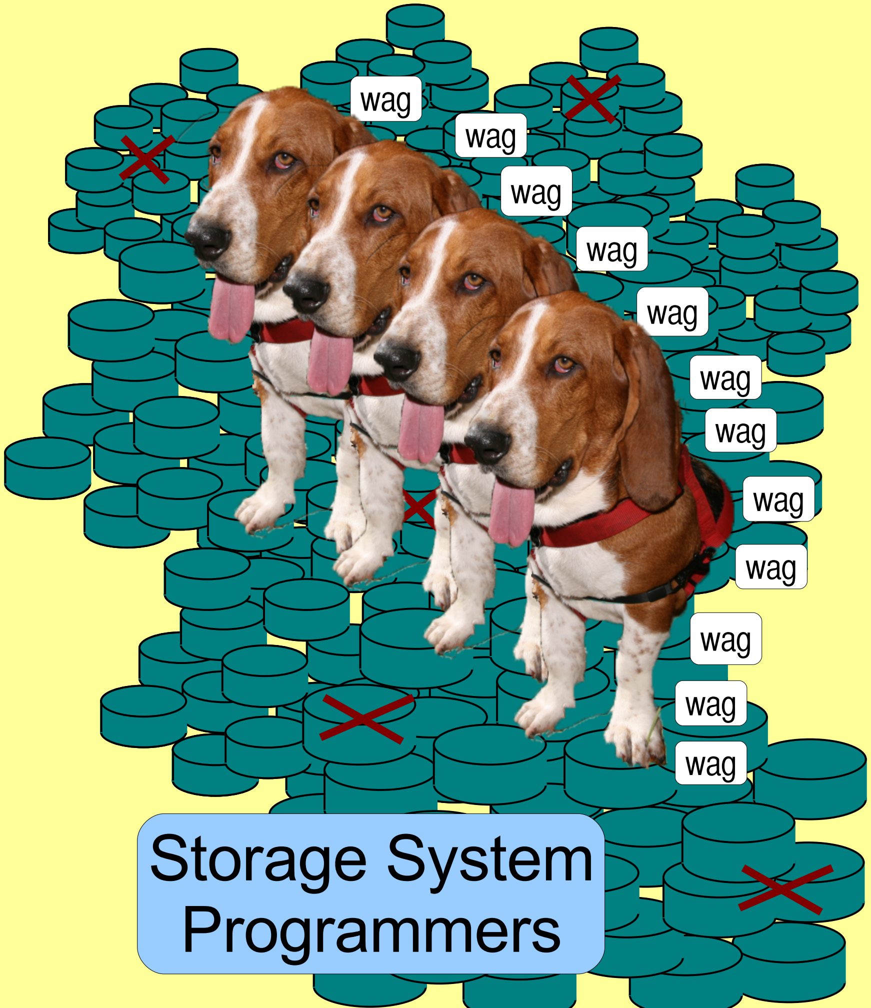
Woof?

Storage System Programmers

My Perspective on Storage

Open Source Libraries

Here's your starting point!



Storage System Programmers

The Point of This Talk

To inform you of the current state of open-source erasure code libraries.

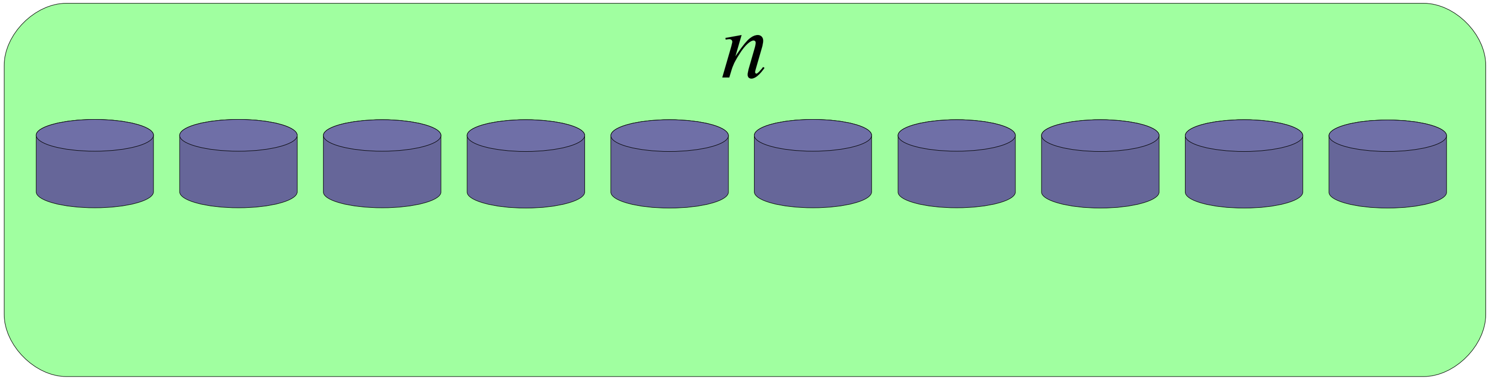
To compare how various codes and implementations perform.

To understand some of the implications of various design decisions.

When you go home, you can converse about erasure codes with your friends & families.

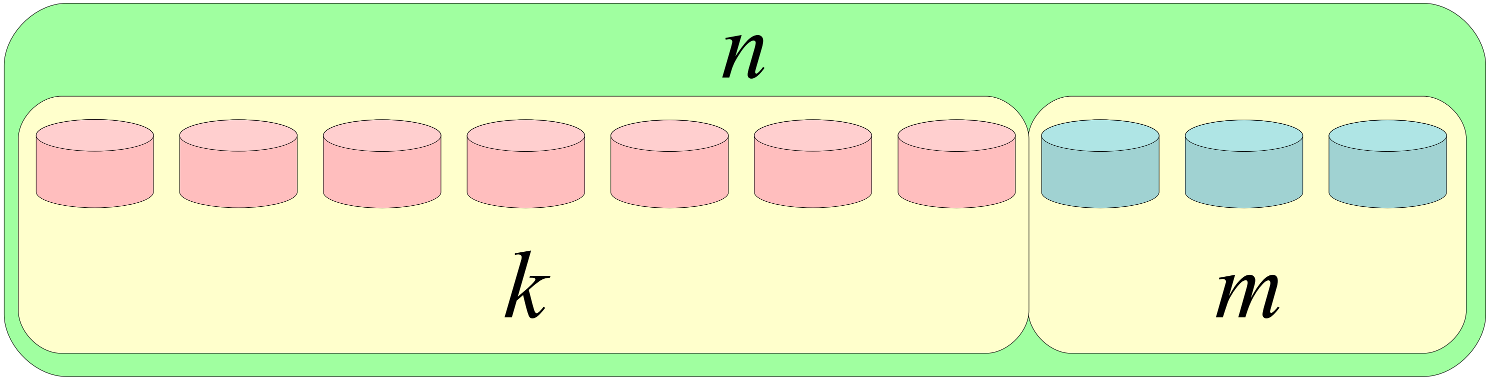
Erasure Coding Basics/Nomenclature

You start with *n* disks:



Erasure Coding Basics/Nomenclature

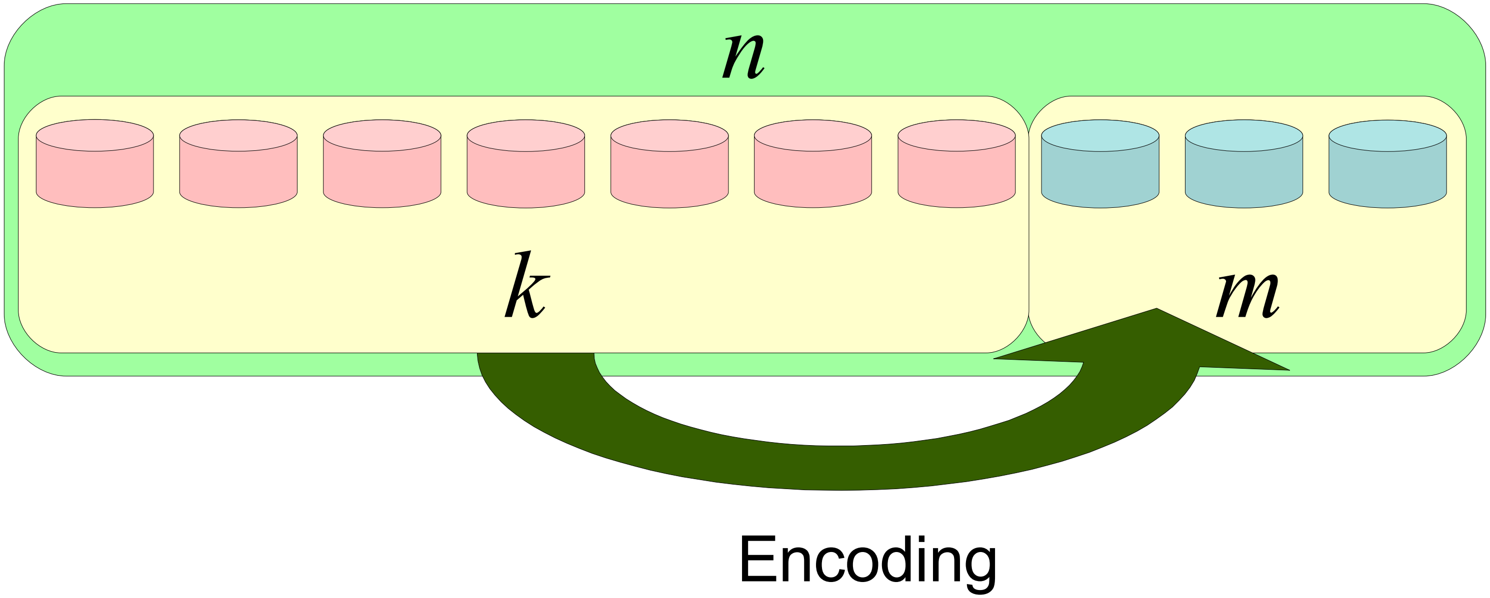
Partition them into *k* data and *m* coding disks.



Call it what you want:
"*k* of *n*."
"*k* and *m*,"
"*[k,m]*."
But please use *k*, *m* and *n*.

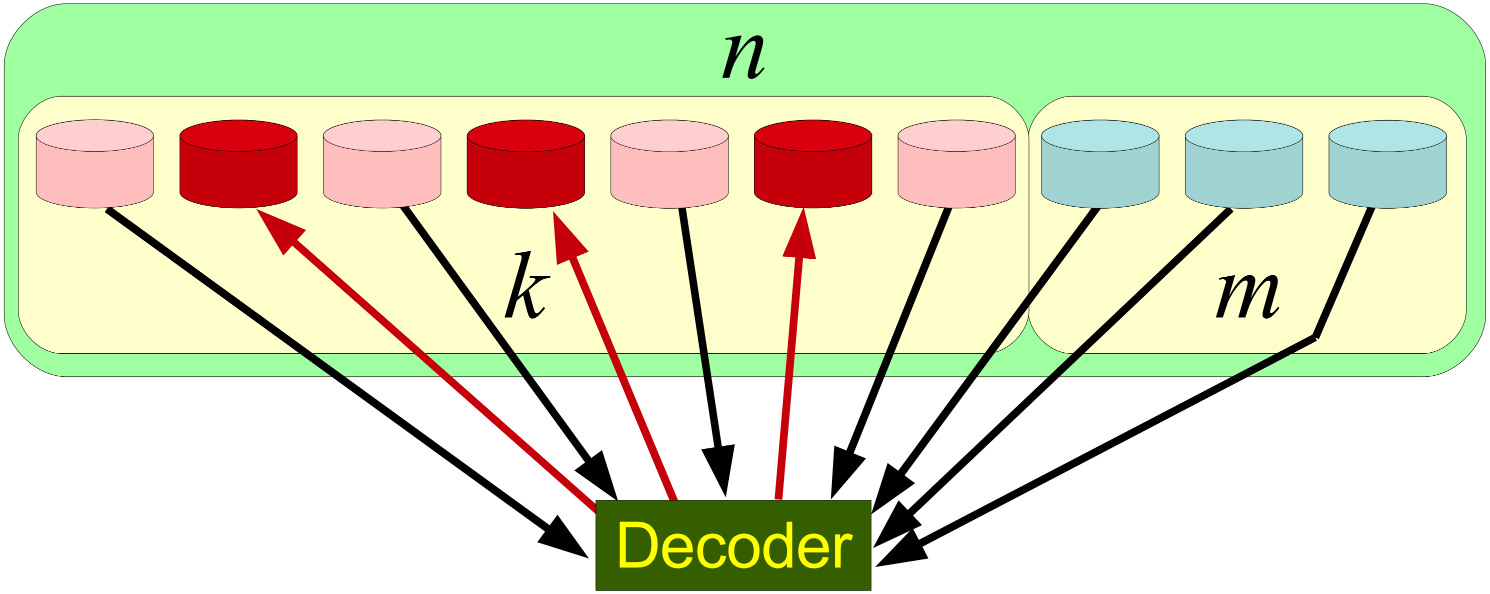
Erasure Coding Basics/Nomenclature

You *encode* by calculating the *m* coding disks from the data.



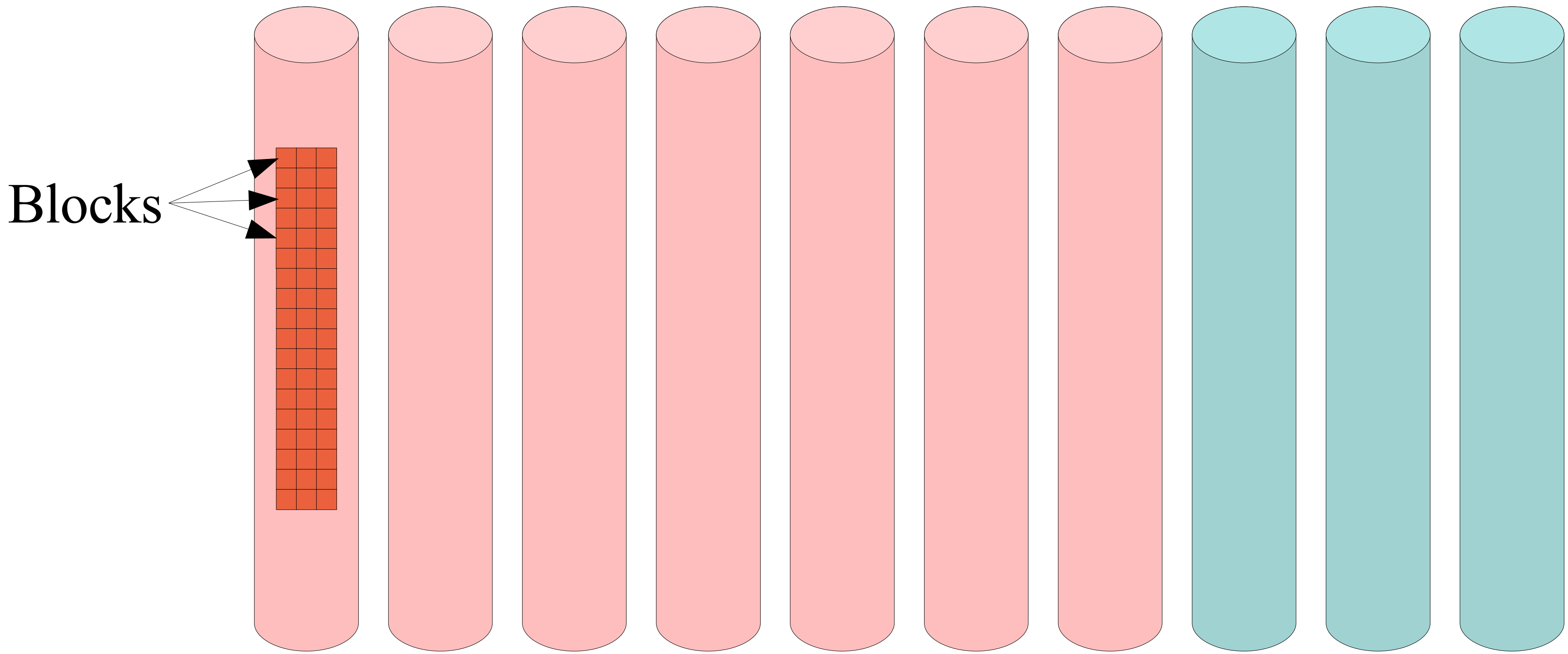
Erasure Coding Basics/Nomenclature

You *decode* by recalculating lost data from the survivors.



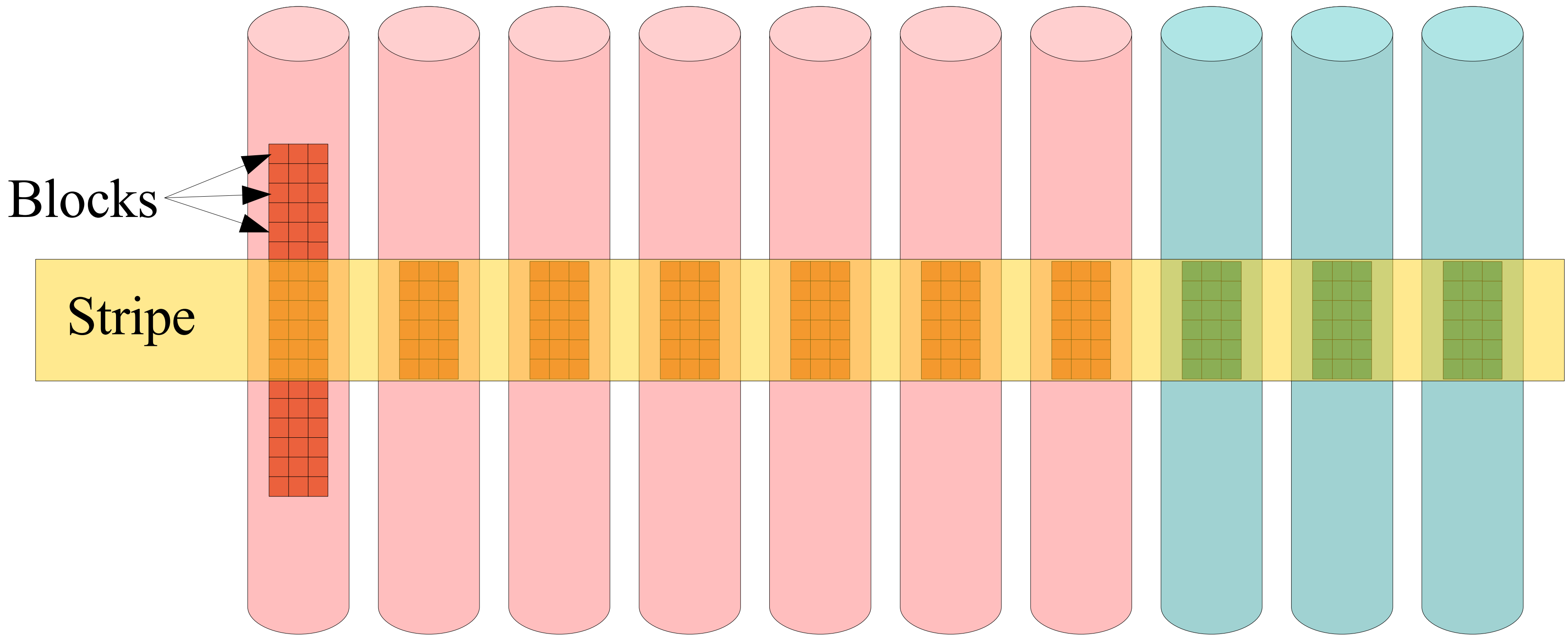
An “MDS” code will tolerate any *m* failures.

Erasure Coding Basics/Nomenclature



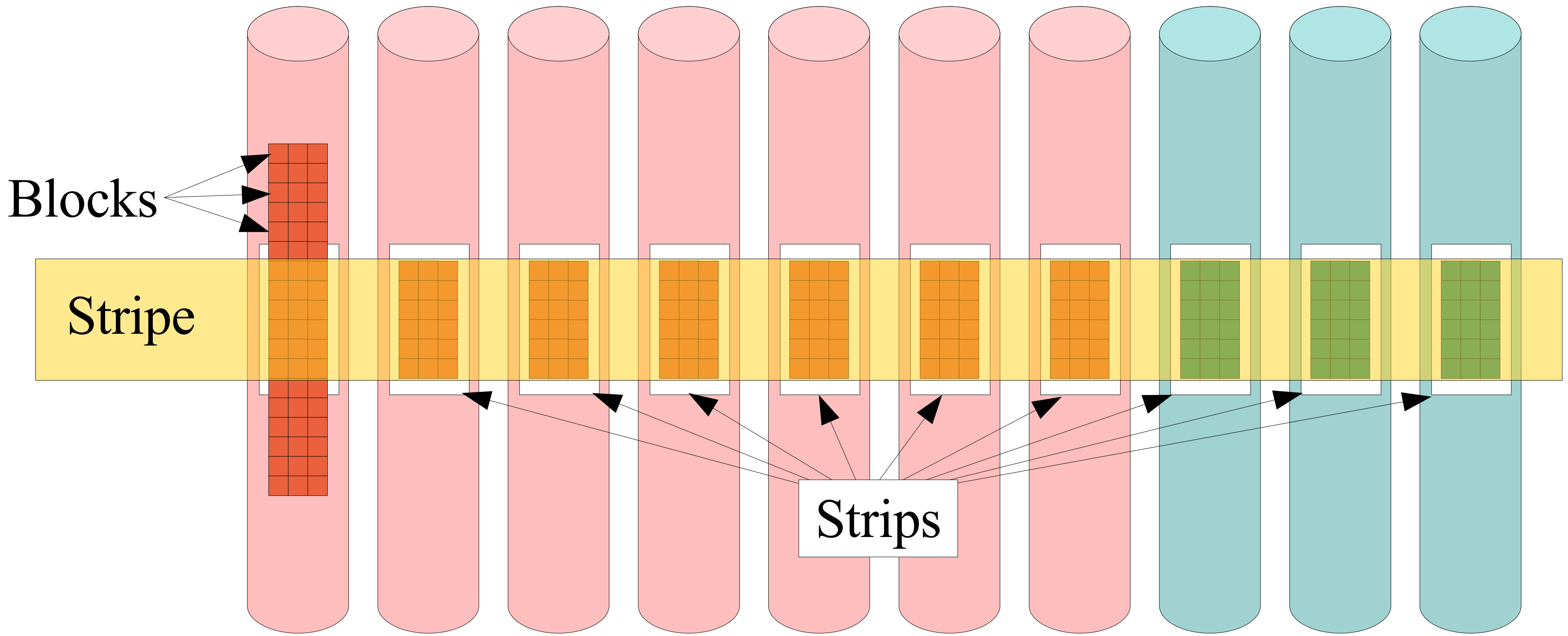
Disks are composed of blocks, stripes, and strips.

Erasure Coding Basics/Nomenclature



Disks are composed of blocks, stripes, and strips.

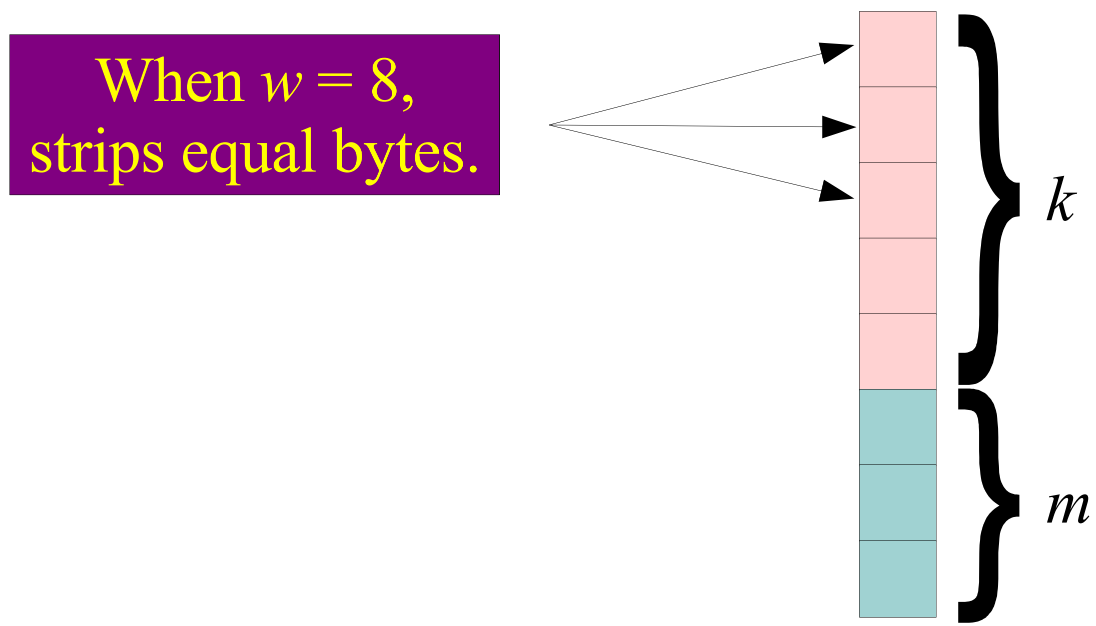
Erasure Coding Basics/Nomenclature



Disks are composed of blocks, stripes, and strips.

Reed-Solomon Codes

Strips are w-bit words, where $n \leq 2^w$.



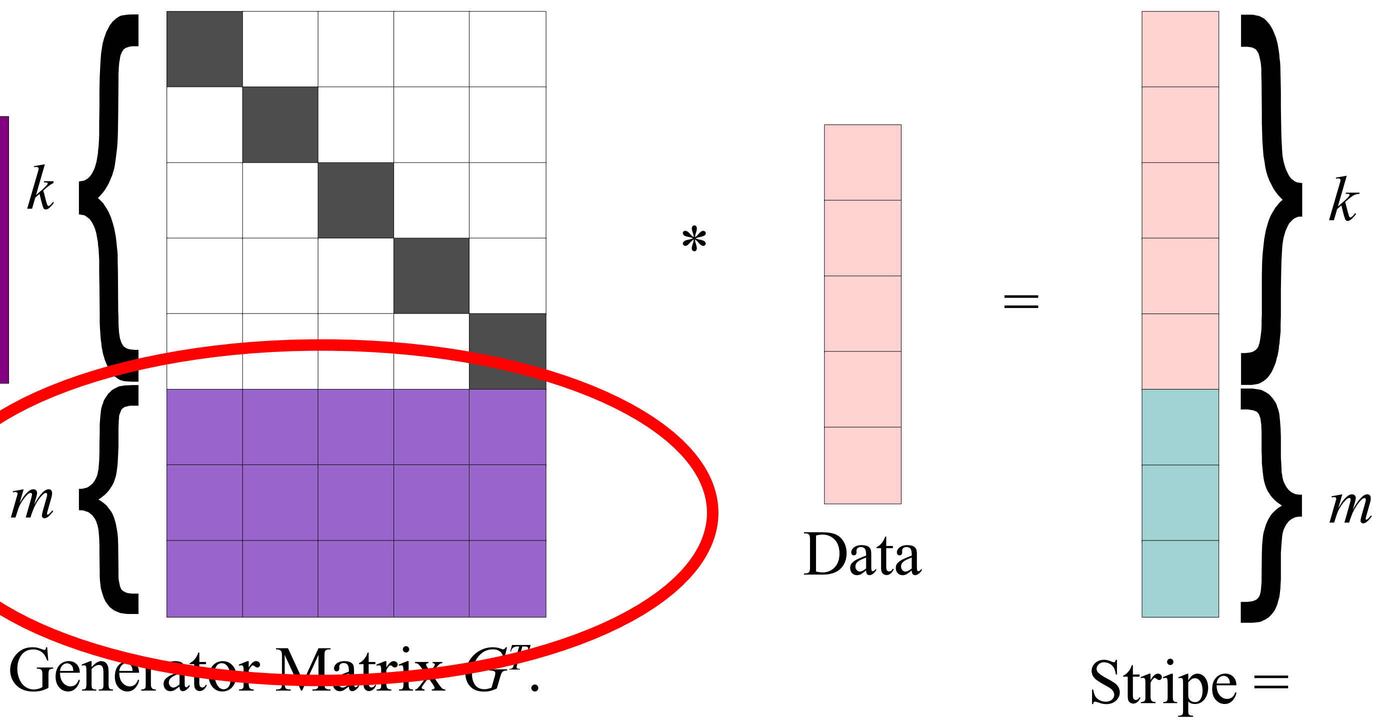
Stripe =
"Codeword"

Reed-Solomon Codes

Coding is described by a matrix-vector product.

Arithmetic is special and expensive.

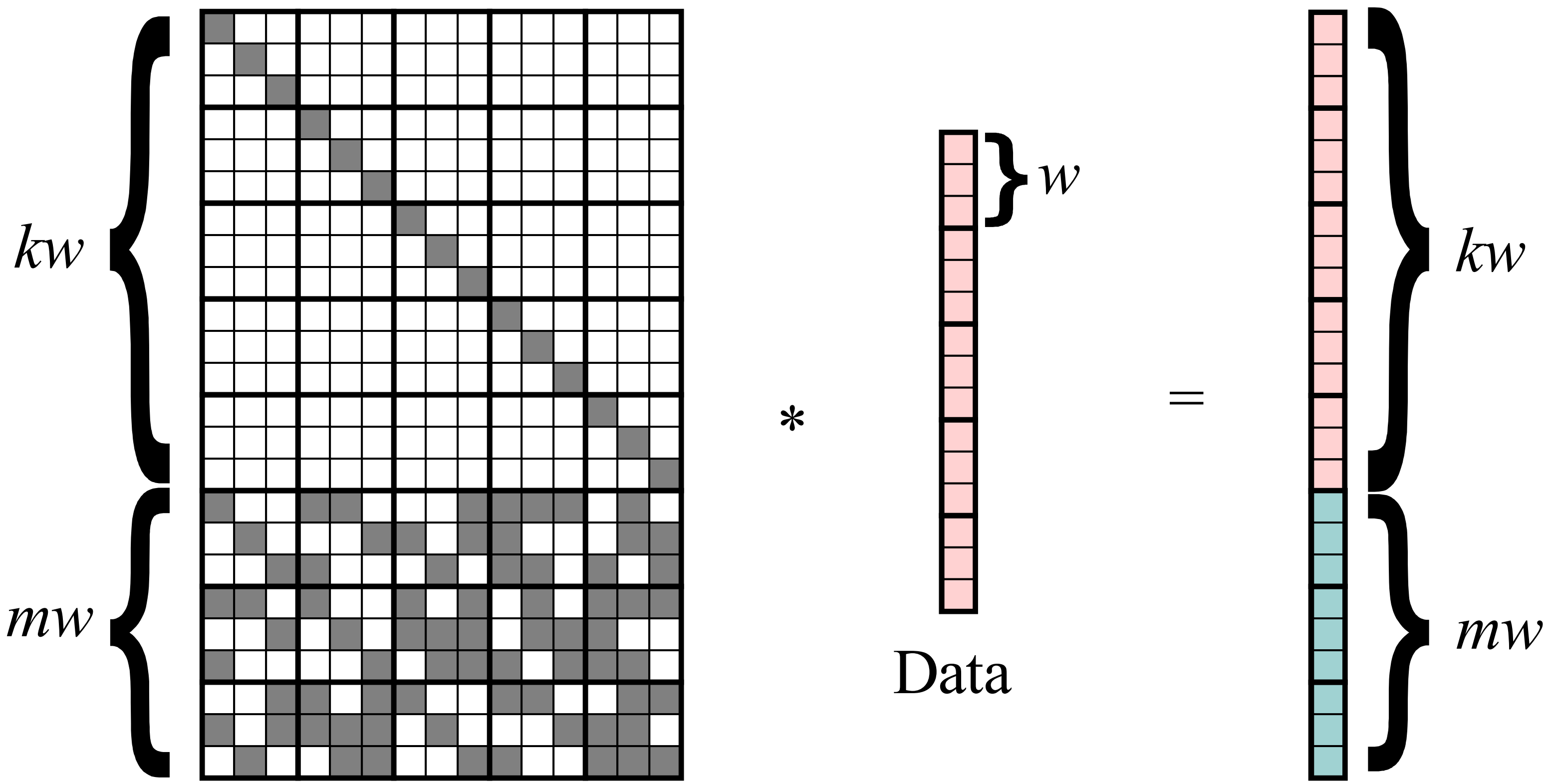
This is all that matters.



Bit Matrix Codes

Strips are each *w* individual bits.

Arithmetic is binary: Addition = XOR, Multiplication = AND



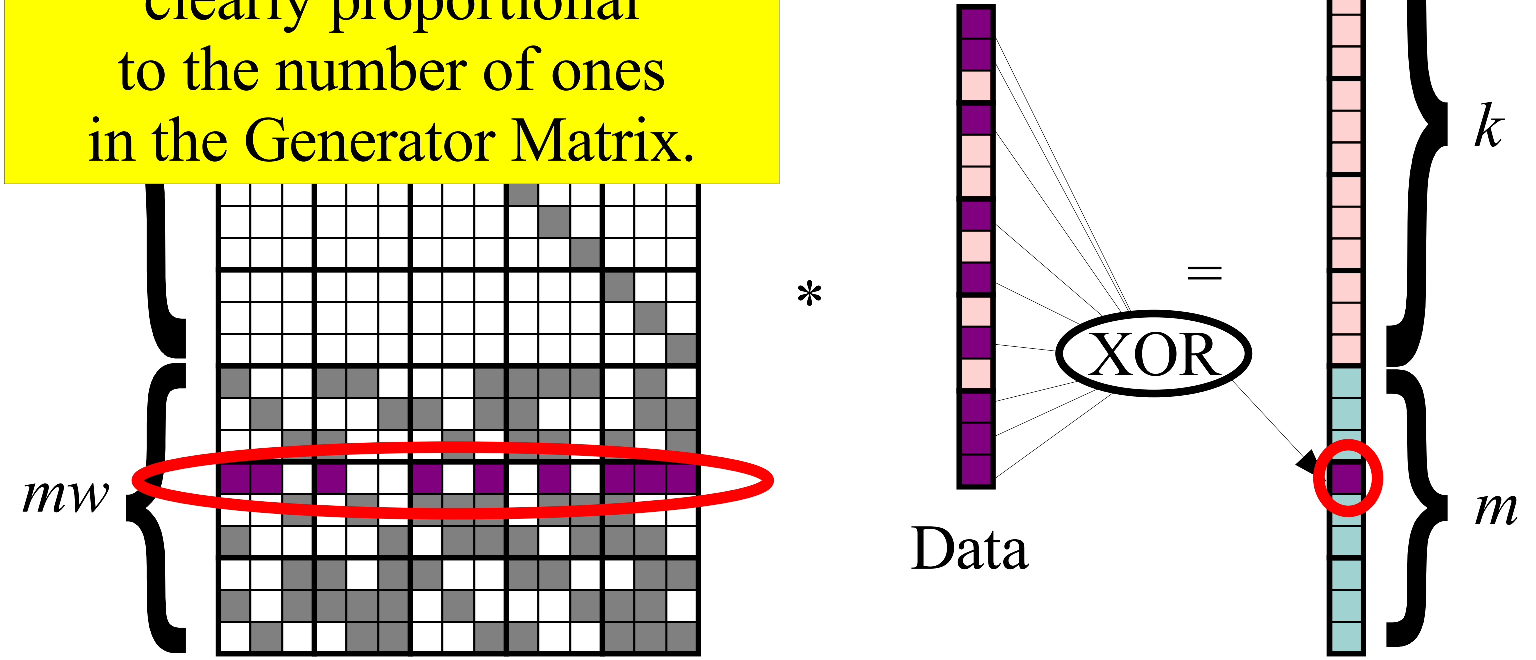
Generator Matrix G^T.

Stripe = "Codeword"

Bit Matrix Codes

Thus, coding bits are XOR sums of various data bits:

Performance is clearly proportional to the number of ones in the Generator Matrix.

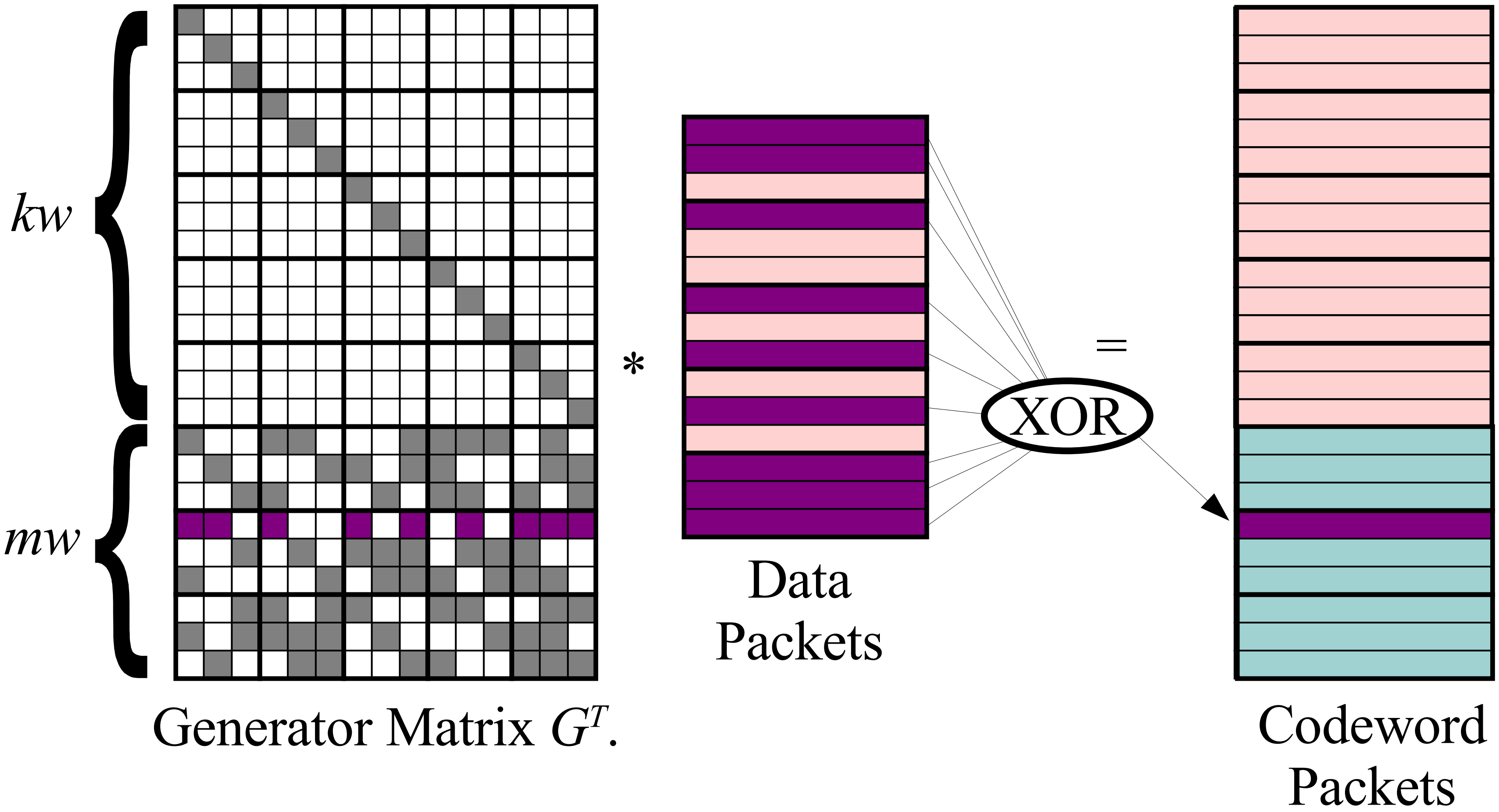


Generator Matrix G^T.

Stripe = "Codeword"

Bit Matrix Codes

For good performance, strips are composed of *packets* rather than bits.



Generator Matrix G^T.

Bit Matrix Codes

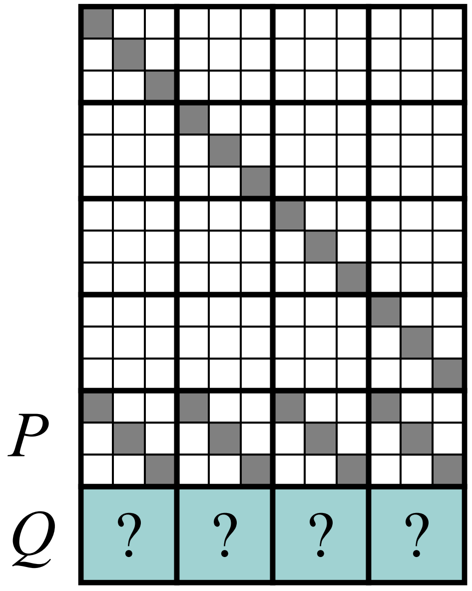
Cauchy Reed Solomon (CRS) Codes [Blomer95]

- Bit Matrix derived from Reed-Solomon code.
- Same constraints: All good as long as $n \leq 2^w$.
- [Plank&Xu06]: Optimization to reduce ones.
- Further optimization [Plank07].

The Special Case of RAID-6

- Two coding disks: *P* & *Q*.
- *P* drive is parity (superset of RAID-4/RAID-5).
- Last row (or last w rows) of Generator Matrix all that matter.

	1	0	0	0
	0	1	0	0
	0	0	1	0
	0	0	0	1
P	1	1	1	1
Q	?	?	?	?



The Special Case of RAID-6

Reed-Solomon Coding Optimization [Anvin07]:

- Multiplication by two can be implemented faster than general multiplication in *GF(2^w)*.
- Arrange the Q row to take advantage of this.

	1	0	0	0
	0	1	0	0
	0	0	1	0
	0	0	0	1
P	1	1	1	1
Q	**1**	**2**	**4**	**8**

Improves encoding but not decoding.

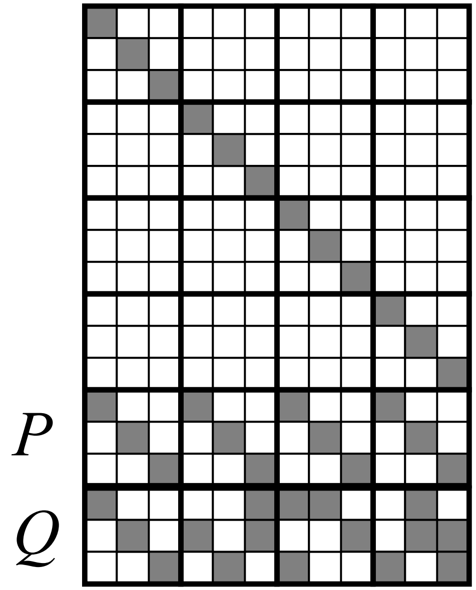
The Special Case of RAID-6

Optimized Cauchy Reed-Solomon Codes [Plank07]:

- For all w, enumerate best values for the Q row.
- Different w have different properties based on the underlying Galois Field arithmetic.

E.g: $k = 14$: Average ones per row:

$w = 7$	-	22.3
$w = 8$	-	28.5
$w = 9$	-	20.1



The Special Case of RAID-6

Minimal Density RAID-6 Codes *($k \leq w$)*:

- Provably minimal number of ones.
  - (w+1) is prime: Blaum-Roth codes [1999]
  - w is prime: Liberation codes [Plank08]
  - w = 8: Liber8tion code [Plank08]
- Performance improves when w increases.
- Requires a scheduling technique [Hafner05] for good decoding.

The Special Case of RAID-6

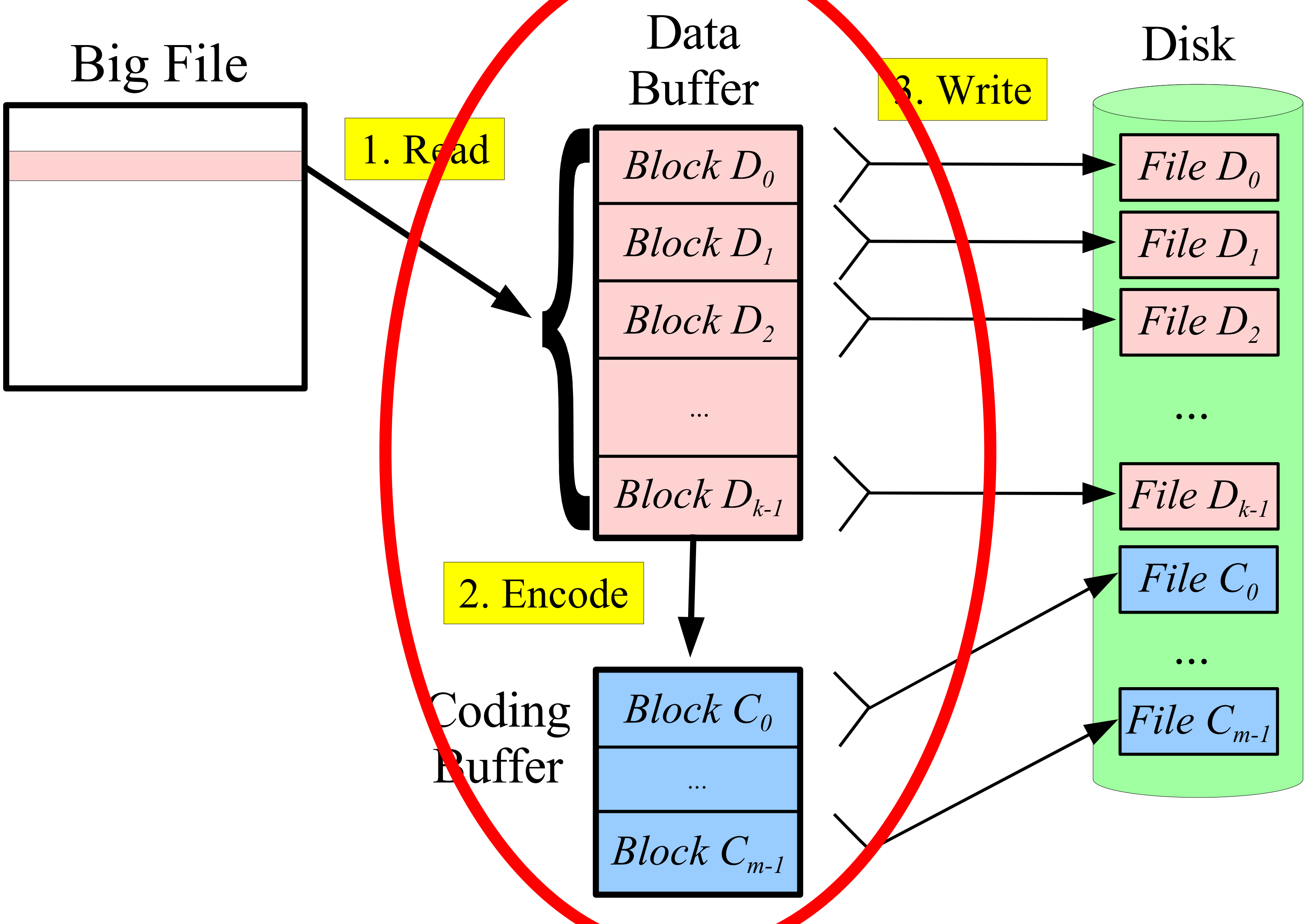
EVENODD [Blaum94] & RDP [Corbett04]:

- *(w+1)* prime, $k \le w$.
- Scheduled non-minimal bit matrices.
- Perform better when w is smaller.
- When $w = k$ or $k+1$, RDP is provably optimal.
- Patented.

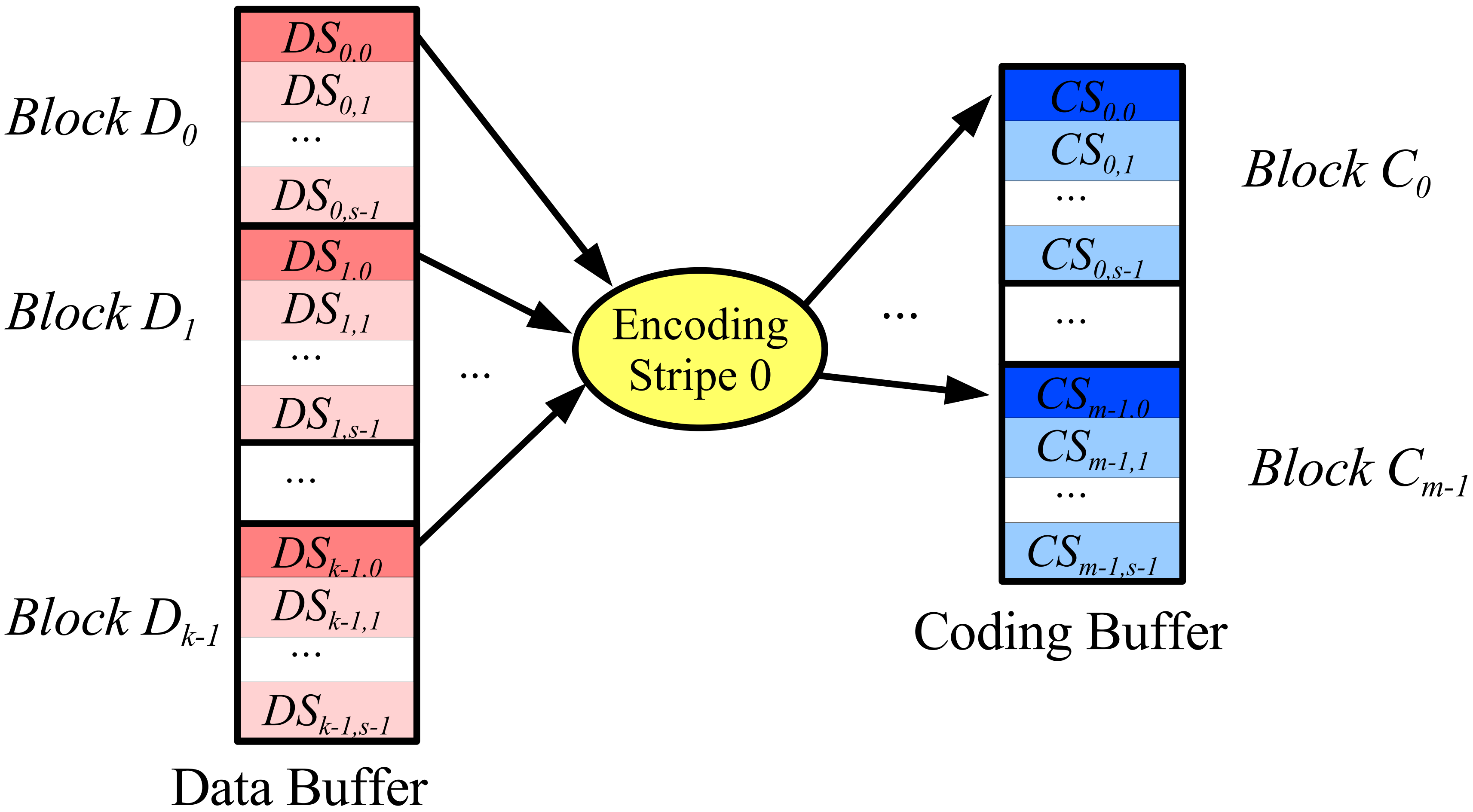
Open Source Libraries

- **Luby**: Original CRS code.
  - (1990 – C)
- **Zfec**: Reed-Solomon coding, $w = 8$.
  - (2007 - C, based on Rizzo 1997)
- **Jerasure**: All of the codes described above.
  - (2007 – C)
- **Cleversafe**: CRS from cleversafe.org, $w = 8$.
  - (2008 – Java, based on **Luby**)
- **RDP/EVENODD**: Added to Jerasure.

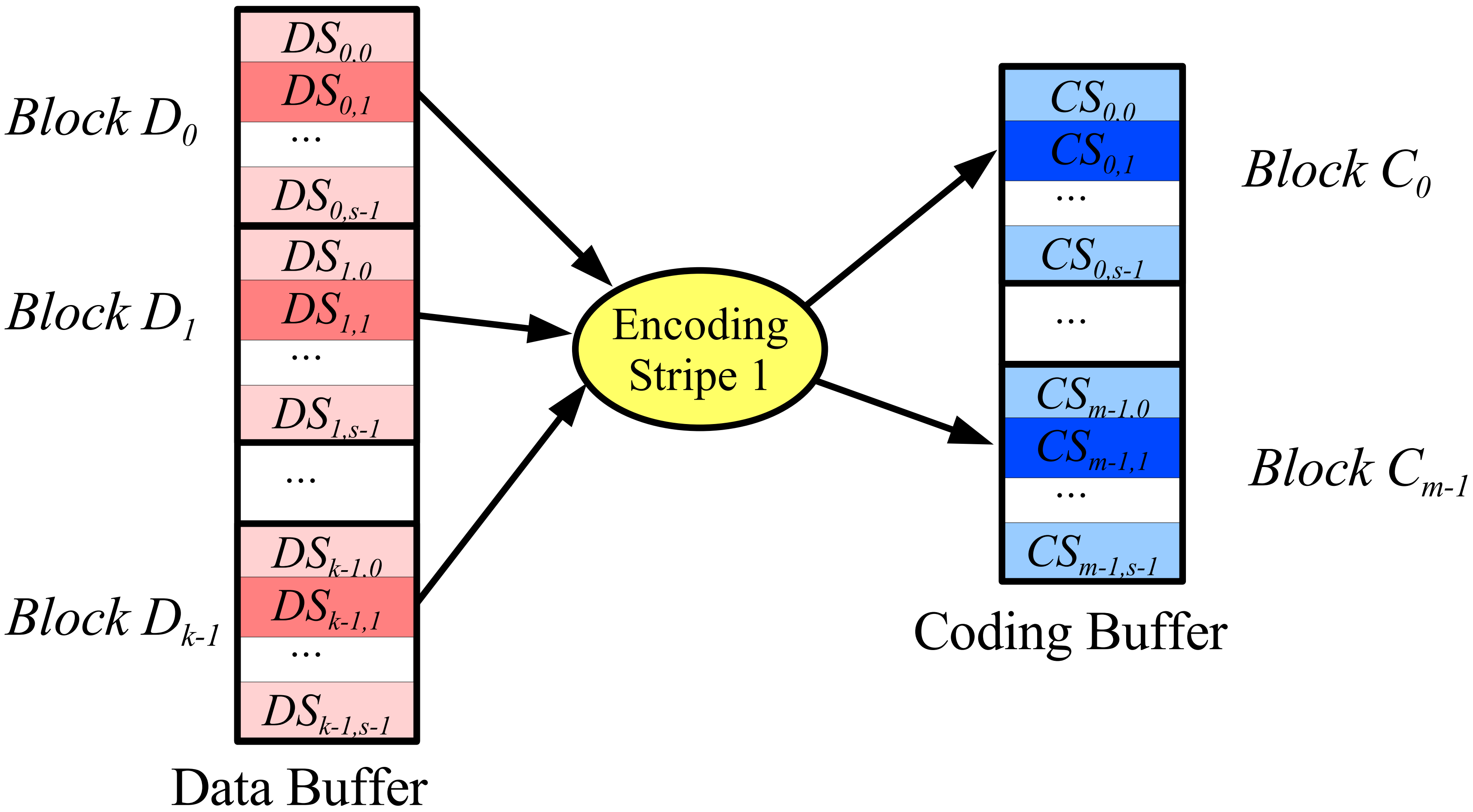
Open Source Tests - Encoding



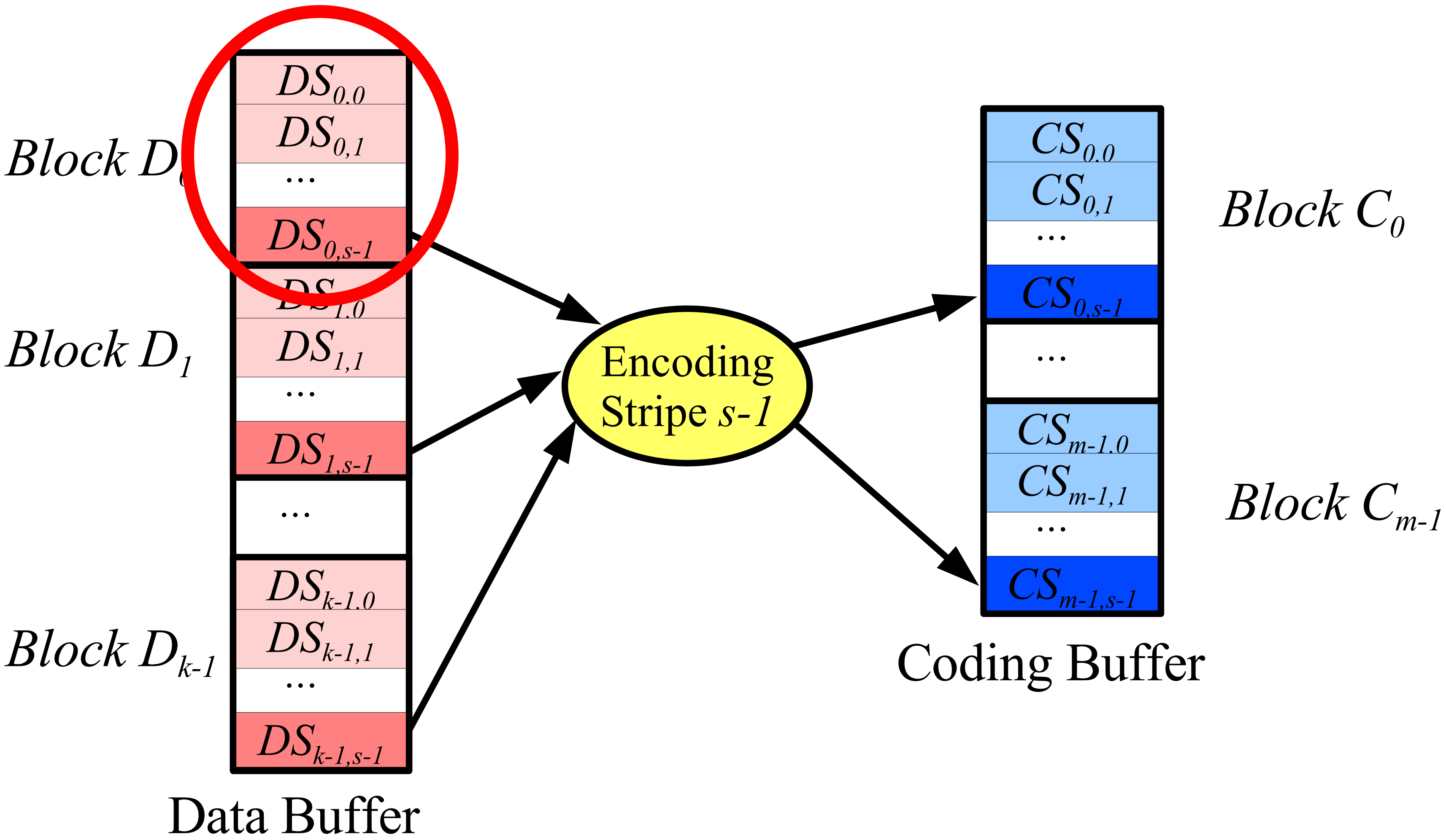
Open Source Tests - Encoding



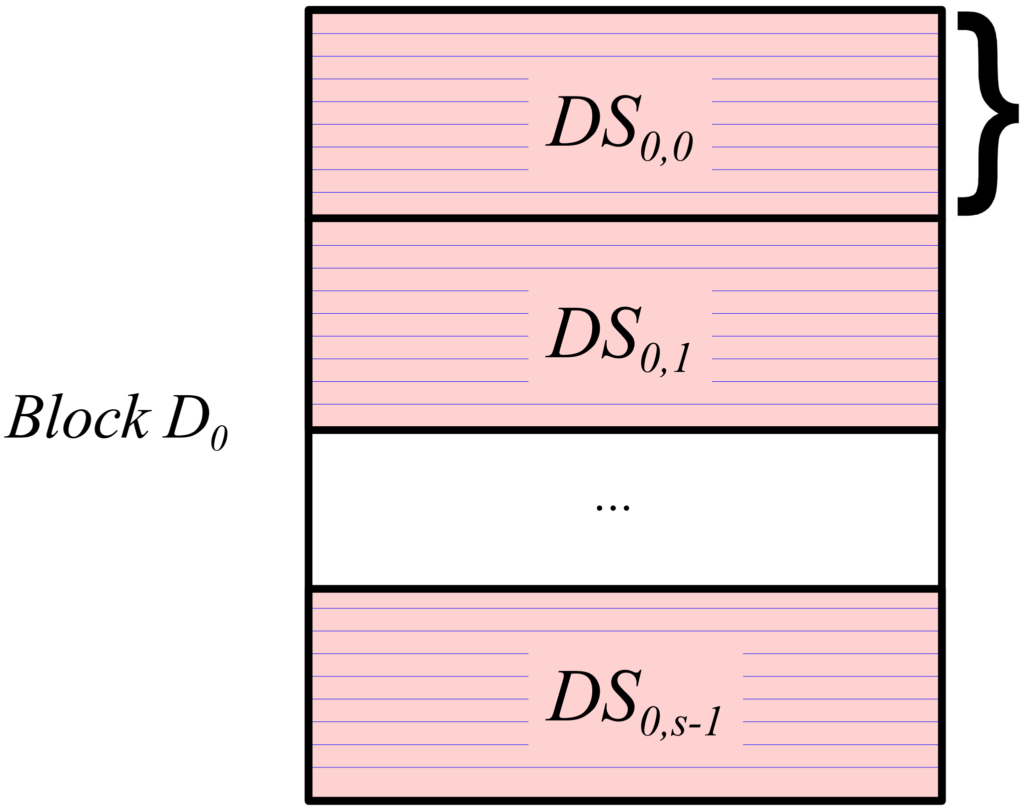
Open Source Tests - Encoding



Open Source Tests - Encoding



Blowing up further.



w packets each of size P.

Each strip is of size wP.

Each block is of size swP.

Data buffer is of size $kswP$.

Parameter Space Explored

- 1GB Video File, ~100 MB data buffer.
- Four configurations: [6,2][14,2][12,4][10,6]
- All implemented codes.
- All legal values of $w \leq 32$.

Machines

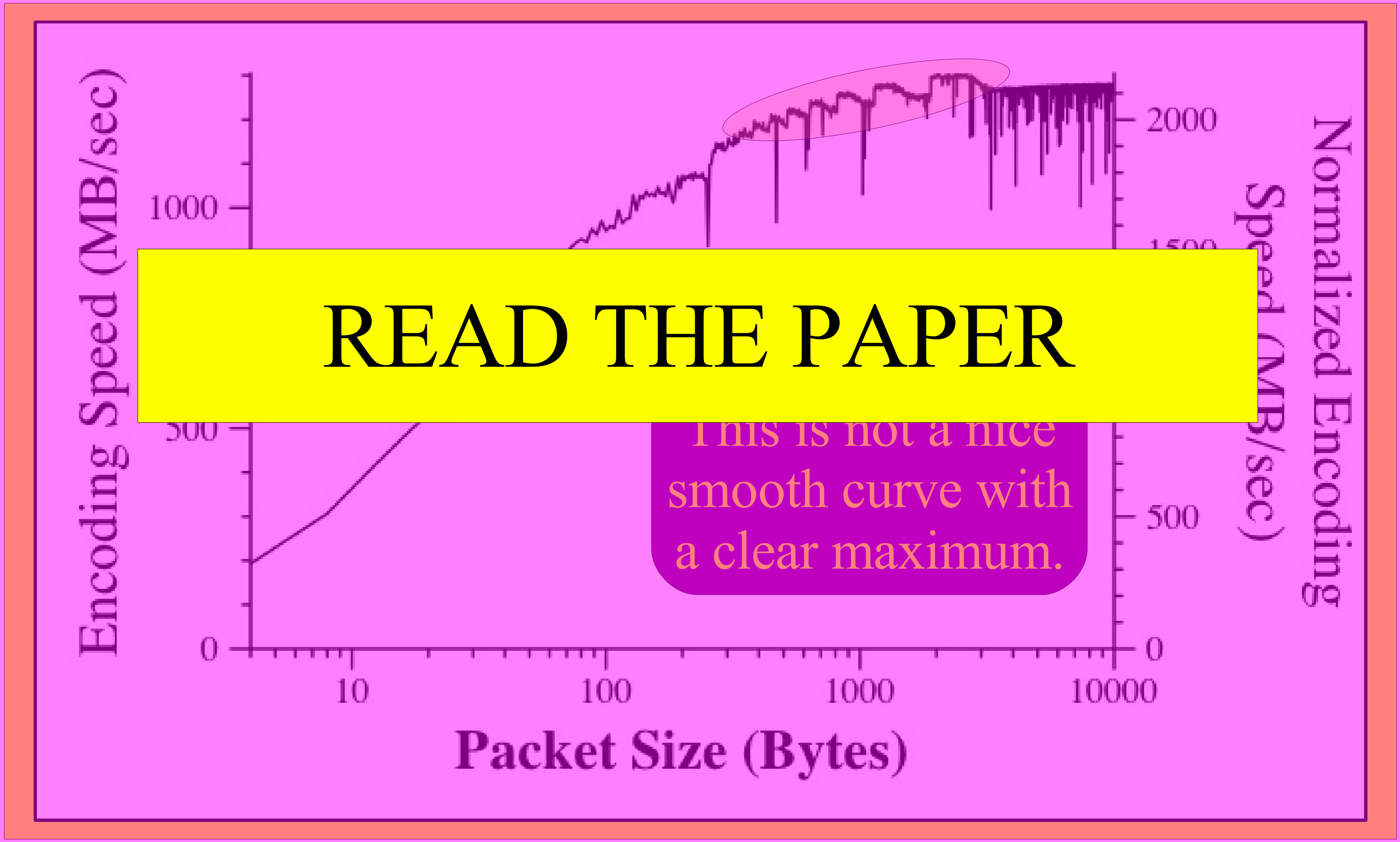
- #1: MacBook (32-bit)
  - 2 GHz Intel Core Duo (only one used).
  - 1 GB RAM, 32KB L1 Cache, 2MB L2 Cache.
  - **memcpy**(): 6.13 GB/s, XOR: 2.43 GB/s.
- #2: Dell (32-bit)
  - 1.5 GHz Intel Pentium 4 .
  - 1 GB RAM, 8KB L1 Cache, 256KB L2 Cache
  - **memcpy**(): 2.92 GB/s, XOR: 1.53 GB/s.

The Measurements that You'll See

- Strip out the disk I/O.
  - You are only seeing encoding/decoding times.
- Averages of 10+ runs, 0.5% variance.
- Show raw speed and “normalized.”

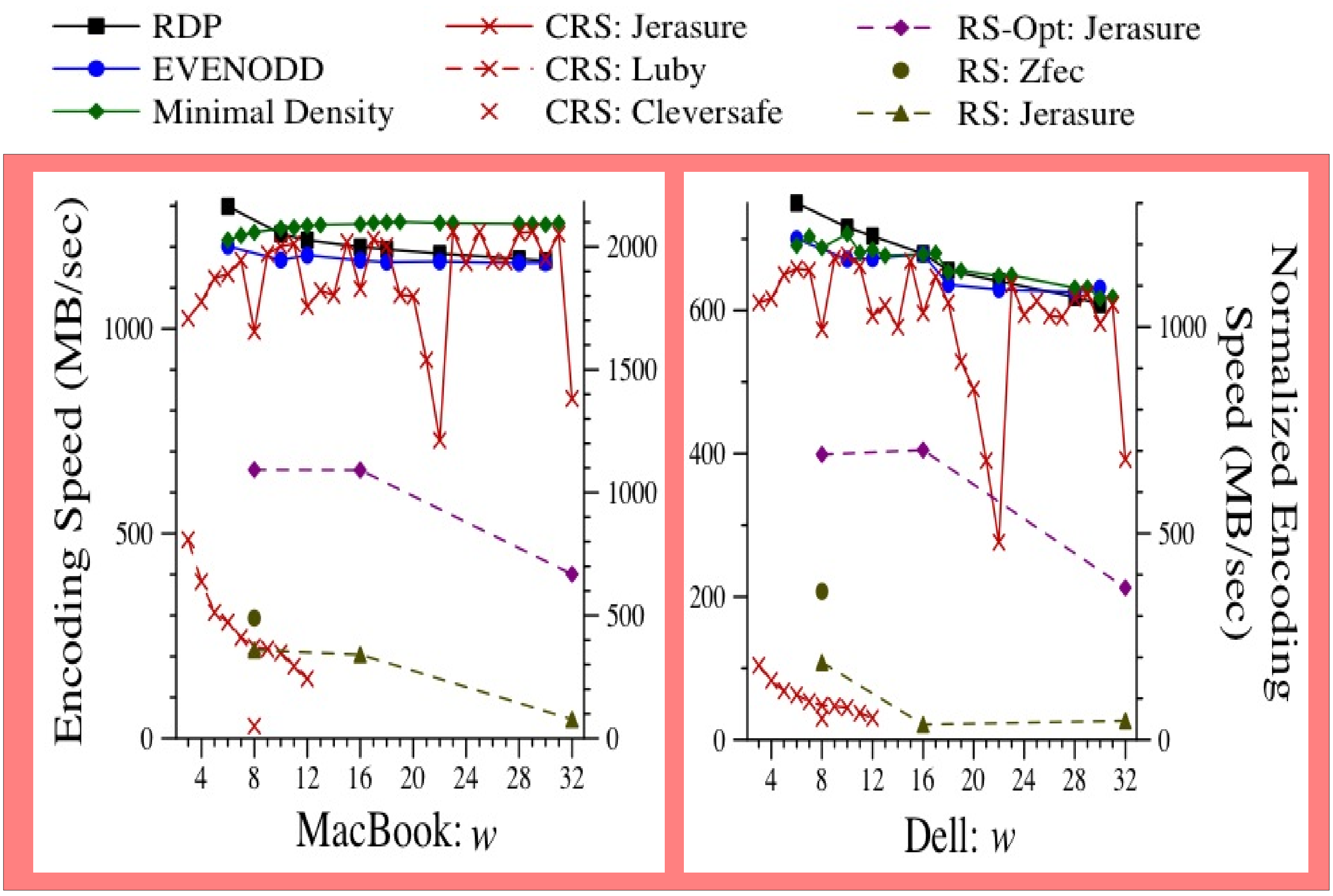
Cache Effects: The packet size.

RDP - [6,2]. w = 6 on MacBook.



READ THE PAPER

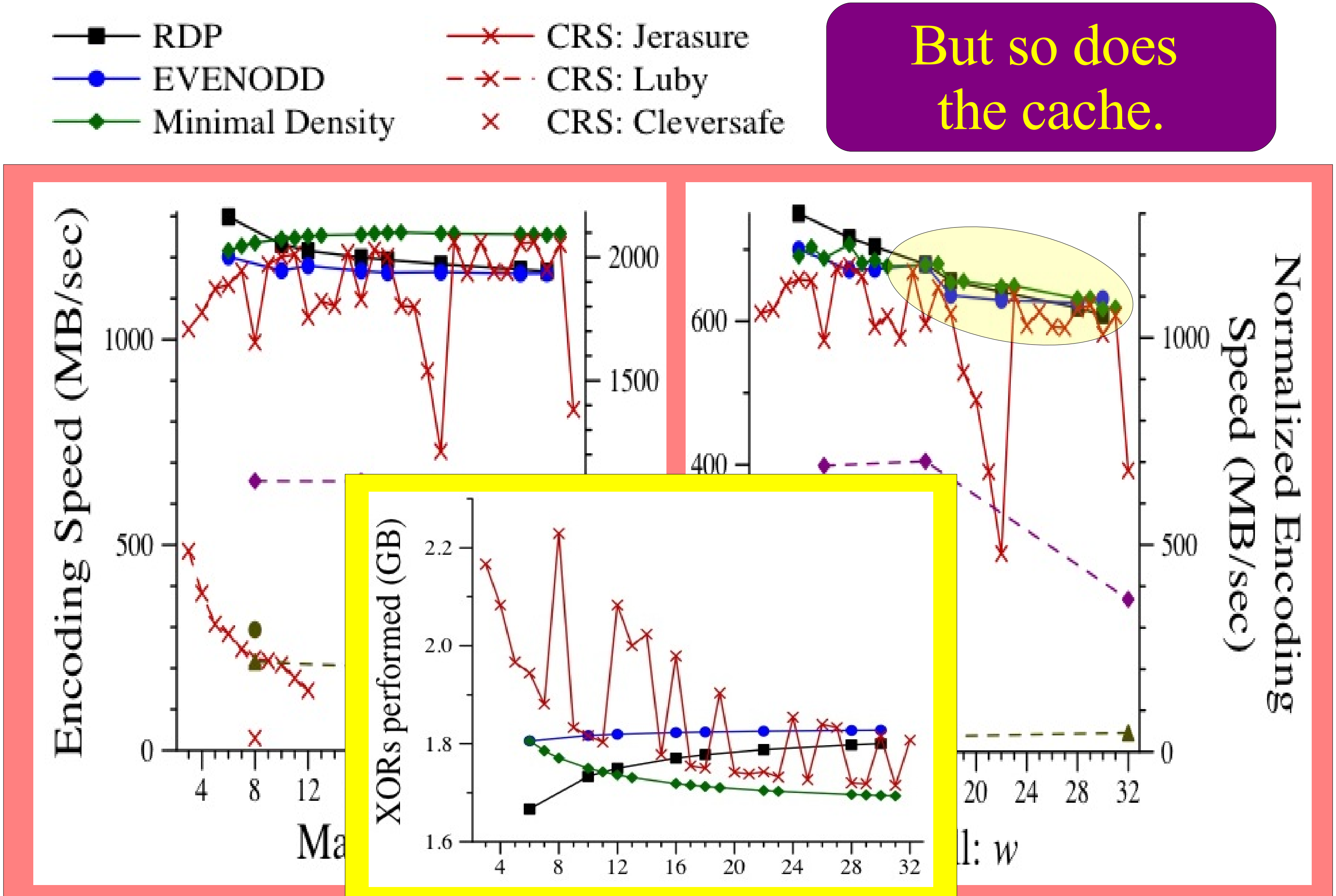
Encoding Performance: [6,2]



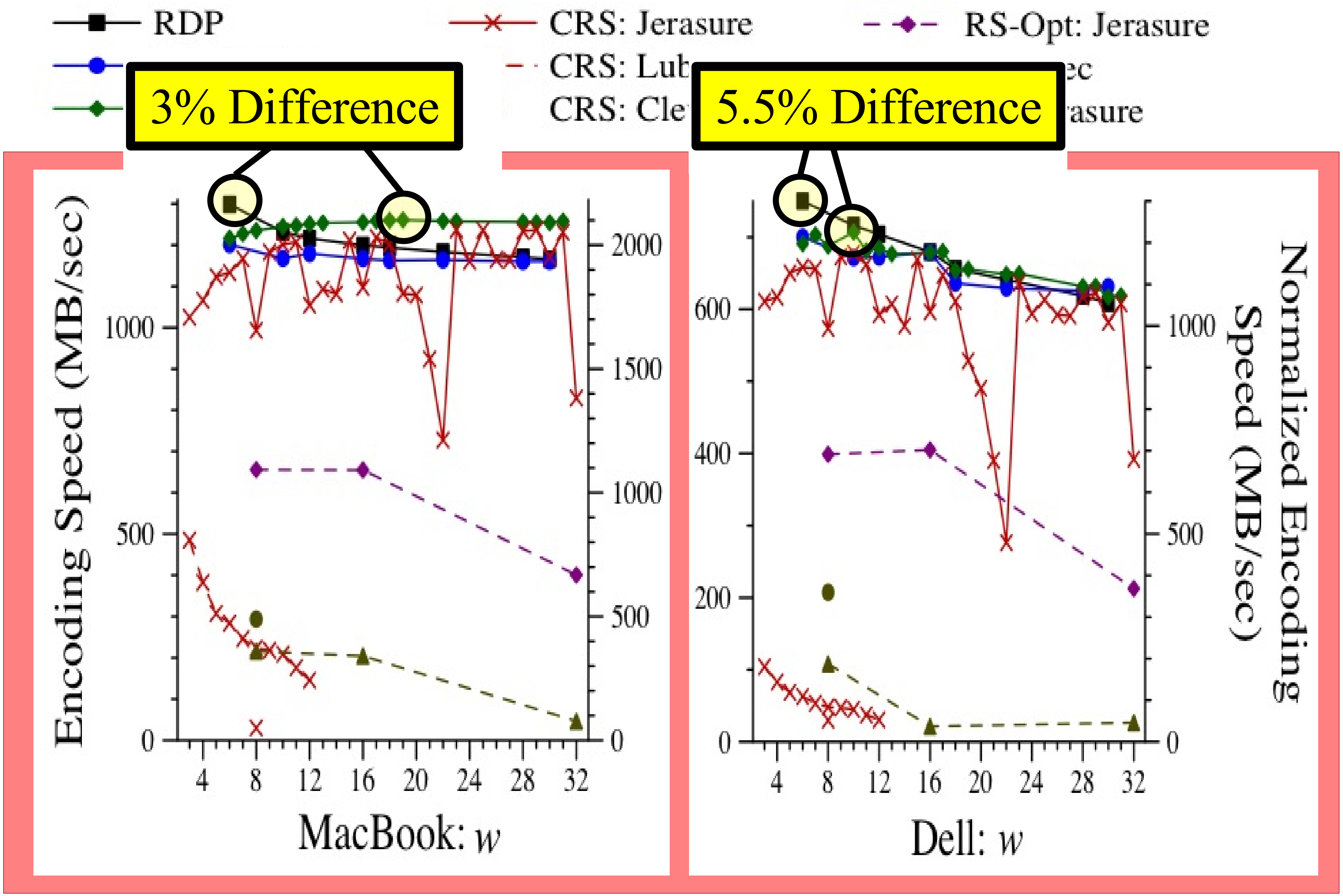
Observation #1
Special purpose codes rock.

Observation #2
XOR count roughly matters.

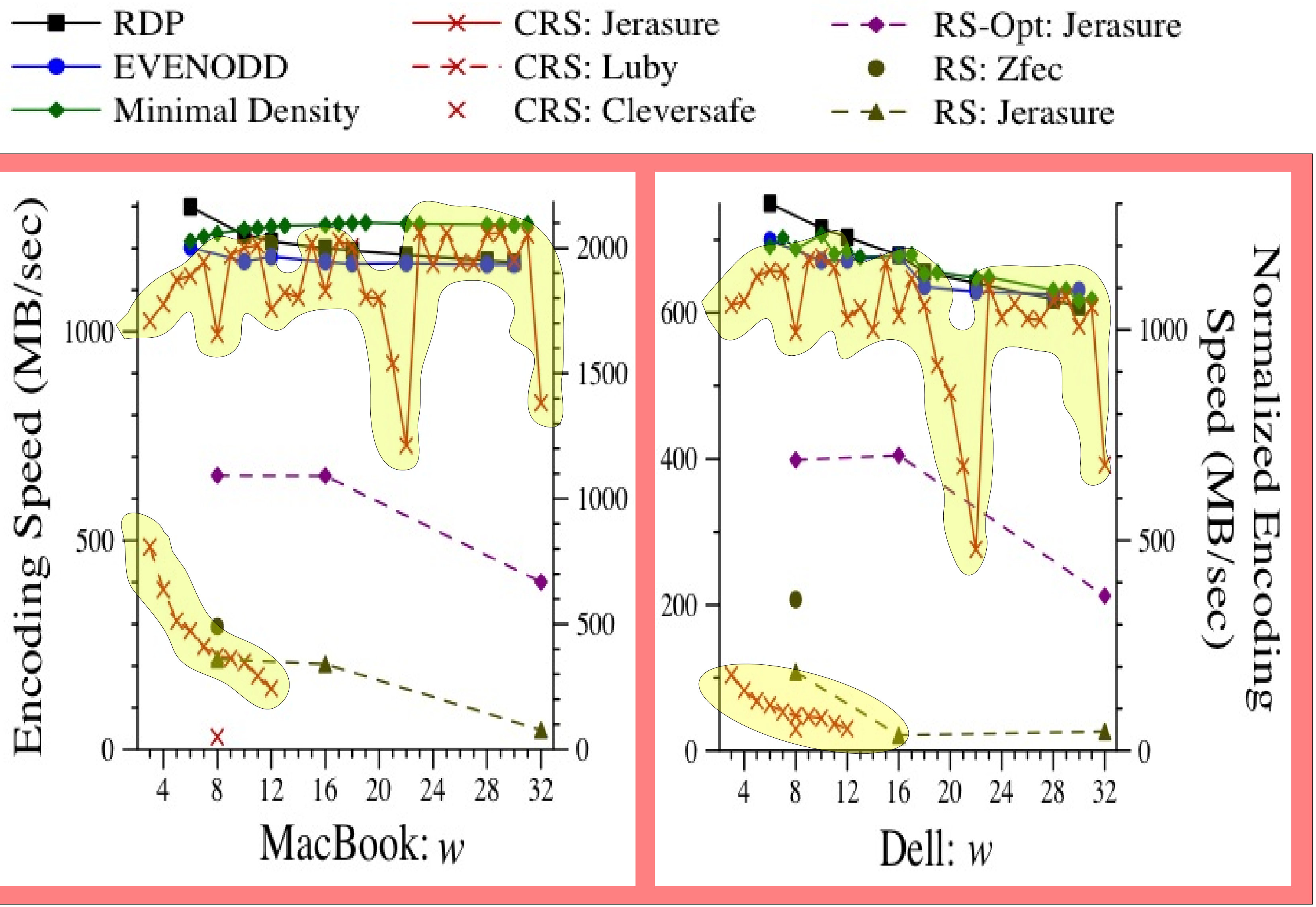
But so does the cache.



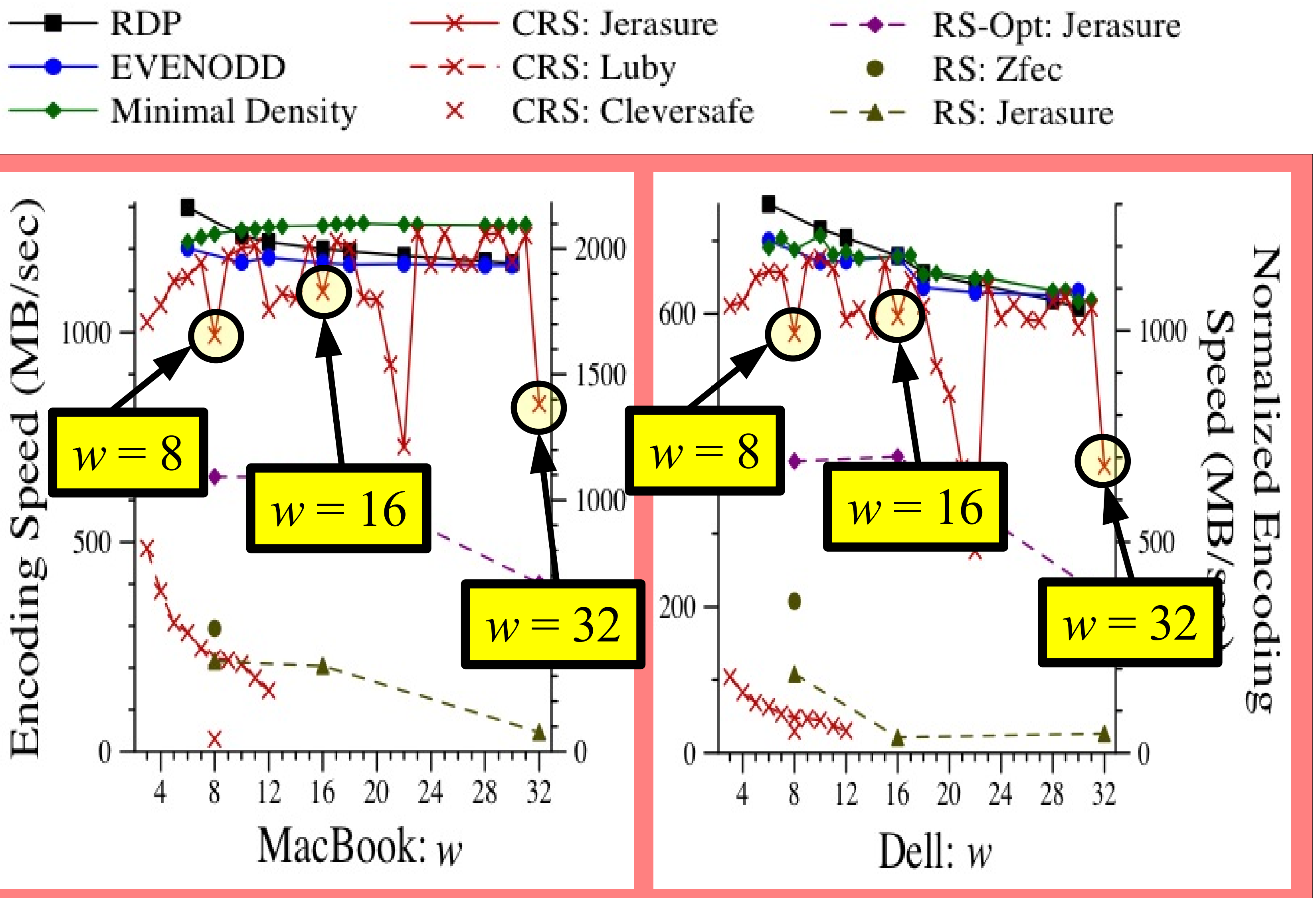
Observation #3. While RDP is a clear winner, others are very close behind.



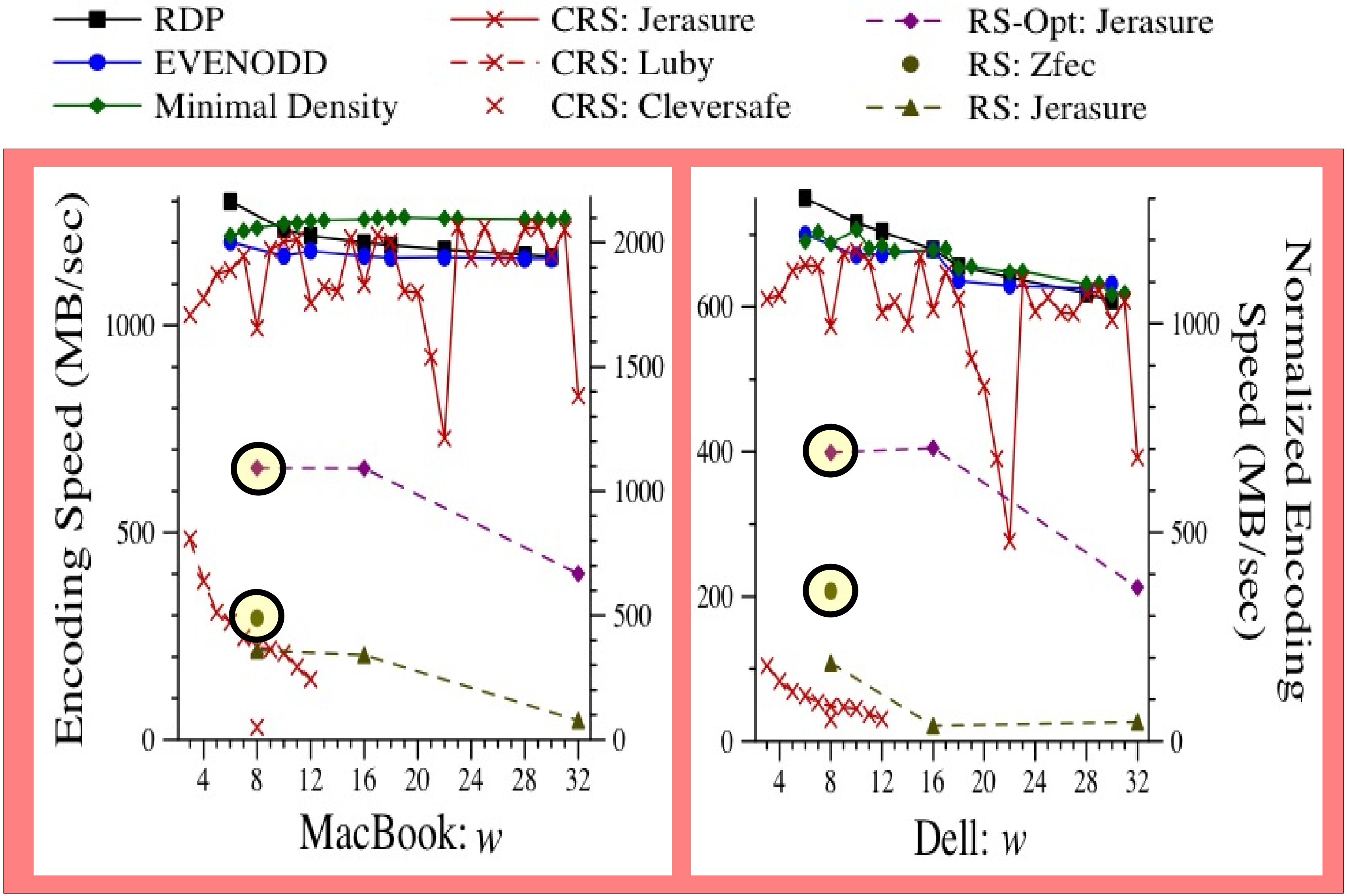
<u>Observation #4.</u> In Cauchy Reed-Solomon Coding, the matrix makes a big difference, as does *w*.



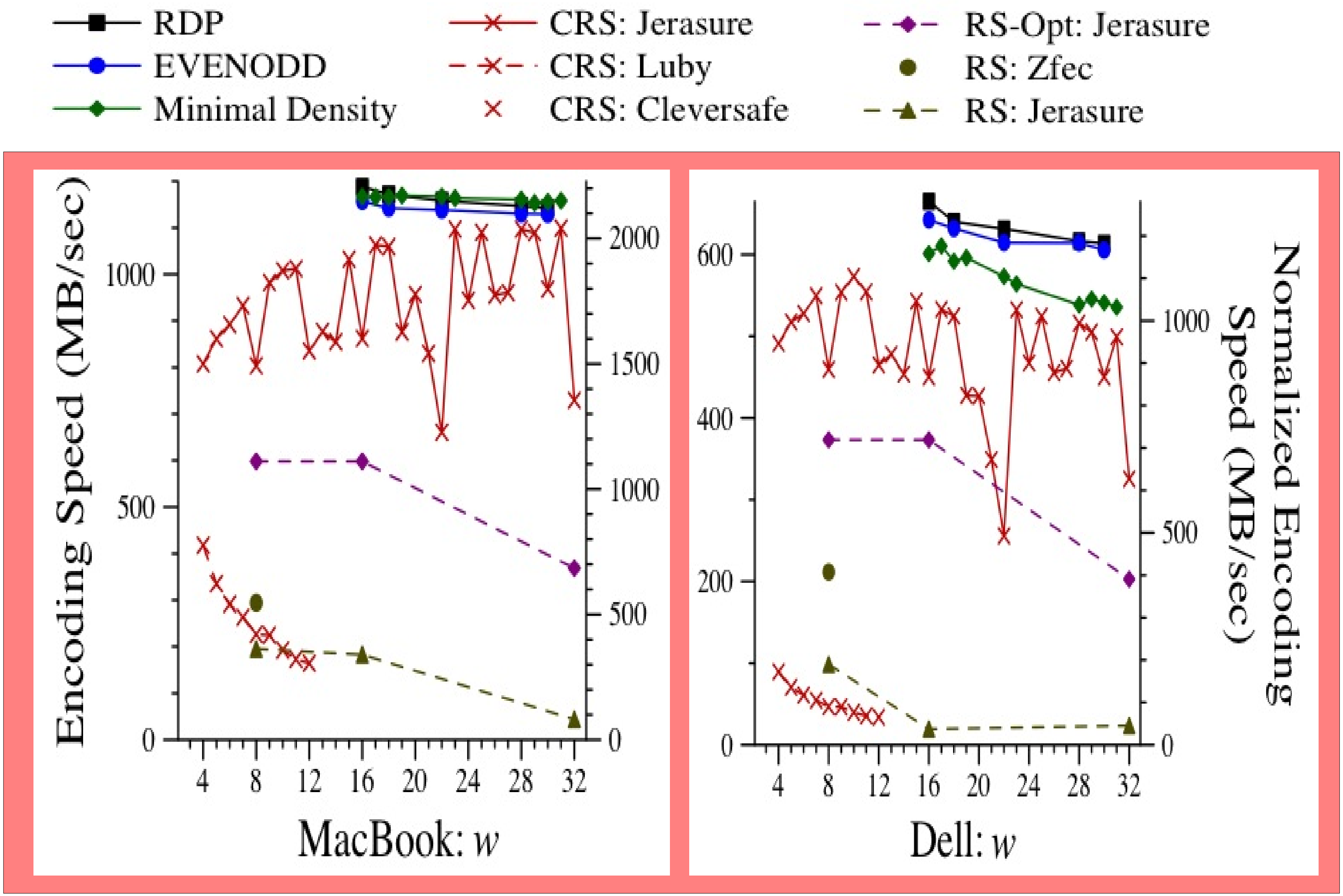
Observation #4. In Cauchy Reed-Solomon Coding, the matrix makes a big difference, as does *w*.



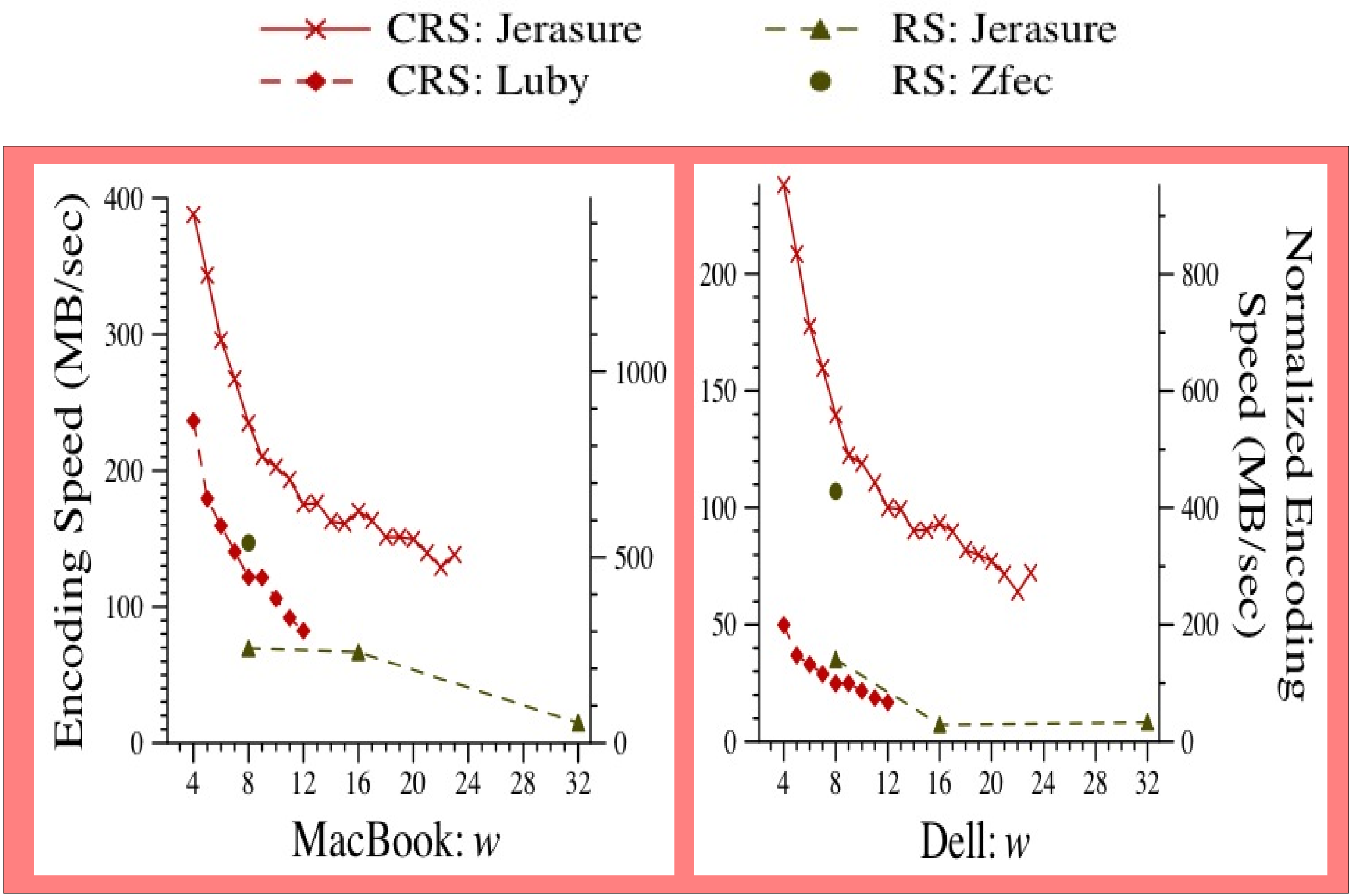
Observation #5. Anvin's optimization is a winner for Reed-Solomon Coding. Zfec has the best performance of the standard Reed-Solomon encoders.



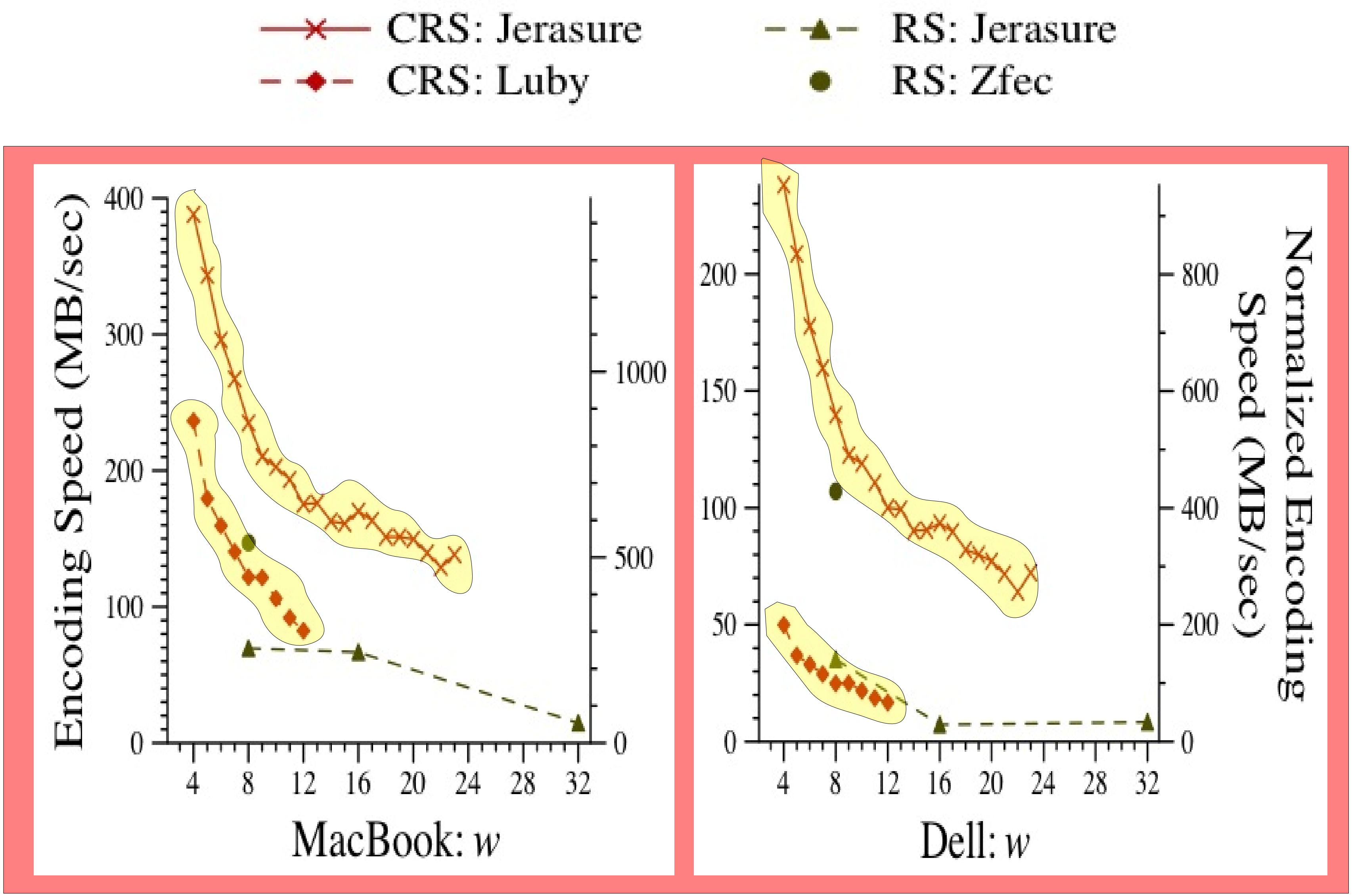
Encoding Performance: [14,2]



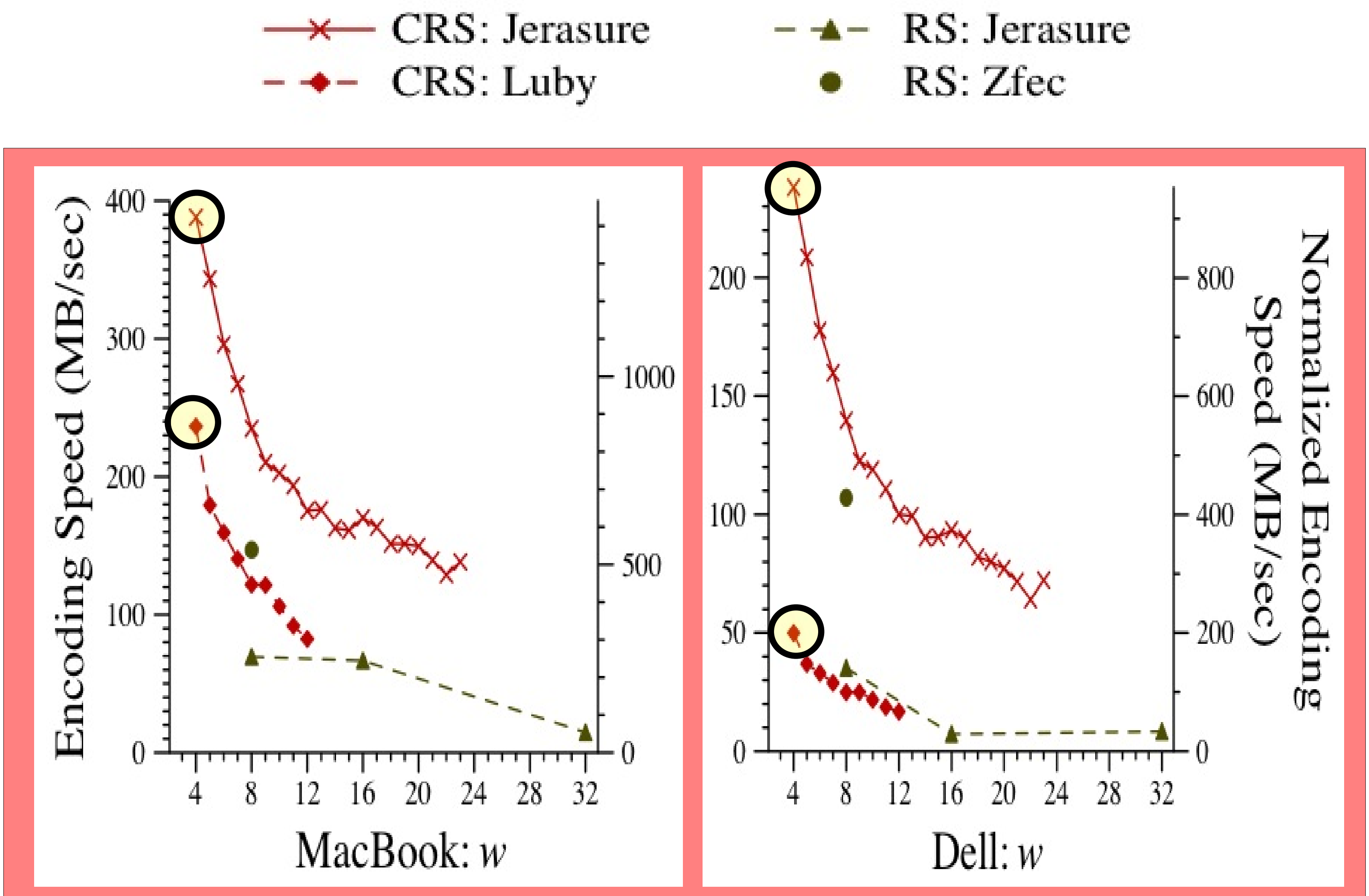
Encoding Performance: [12,4]



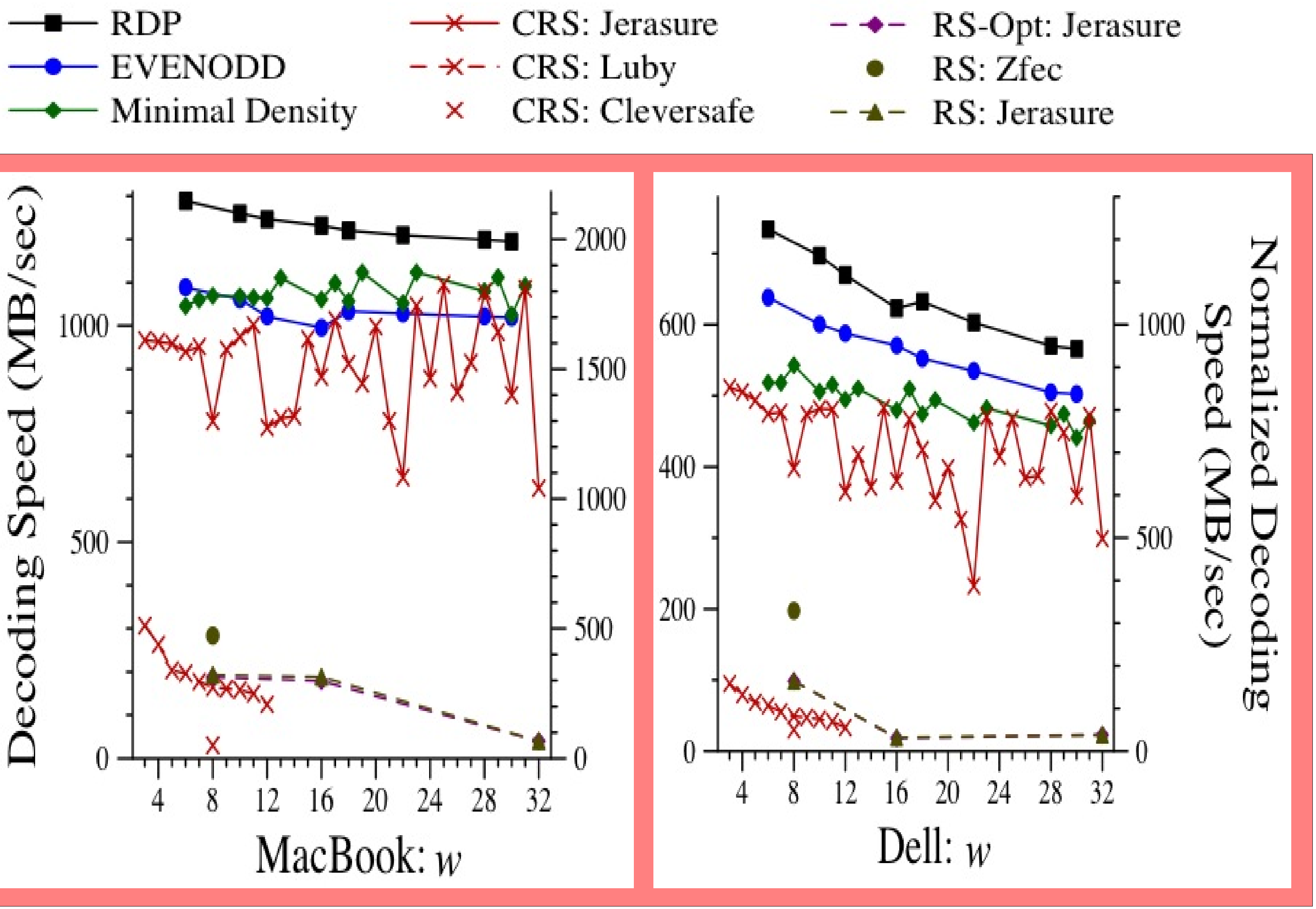
Observation #1: The matrix matters still. [12,4]



Observation #2: Smaller *w* are better. [12,4]



Decoding Performance: [6,2]



Conclusions from the study

Open source erasure code implementations can easily keep up with disks, even on slow CPUs.

Special purpose RAID-6 codes are much better than general-purpose alternatives.

Cauchy Reed-Solomon coding is the better general purpose code.

With Cauchy Reed-Solomon coding, the matrix matters.

With all codes, attention must be paid to *w* and to memory/cache.

Biggest impact of further research:
Beat Reed-Solomon coding beyond RAID-6.

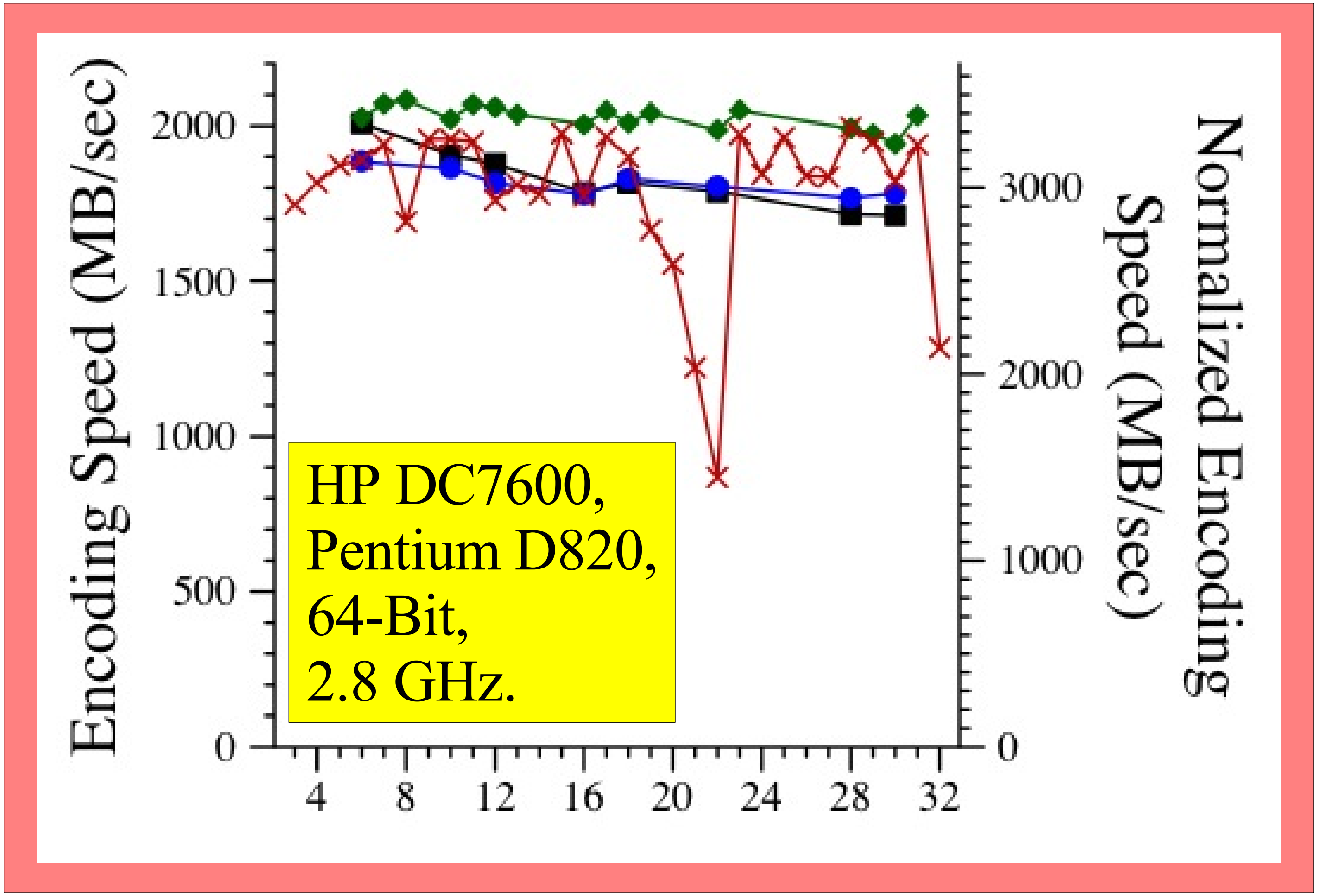
Anticipating Some Questions:

"Your machines suck. "

"Why didn't you use better ones?"

"Why no multicore?"

"Why no use of SSE?"



Anticipating Some Questions:

"My friend has an implementation of Reed-Solomon that blows all of your codes away."

"What do you have to say about that?"

Cool. Post it.

"Why didn't you test the Reed-Solomon codec in the Linux kernel?"

My bad. We should have.

A Performance Evaluation of Open Source Erasure Codes for Storage Applications

James S. Plank
Catherine D. Schuman
(Tennessee)

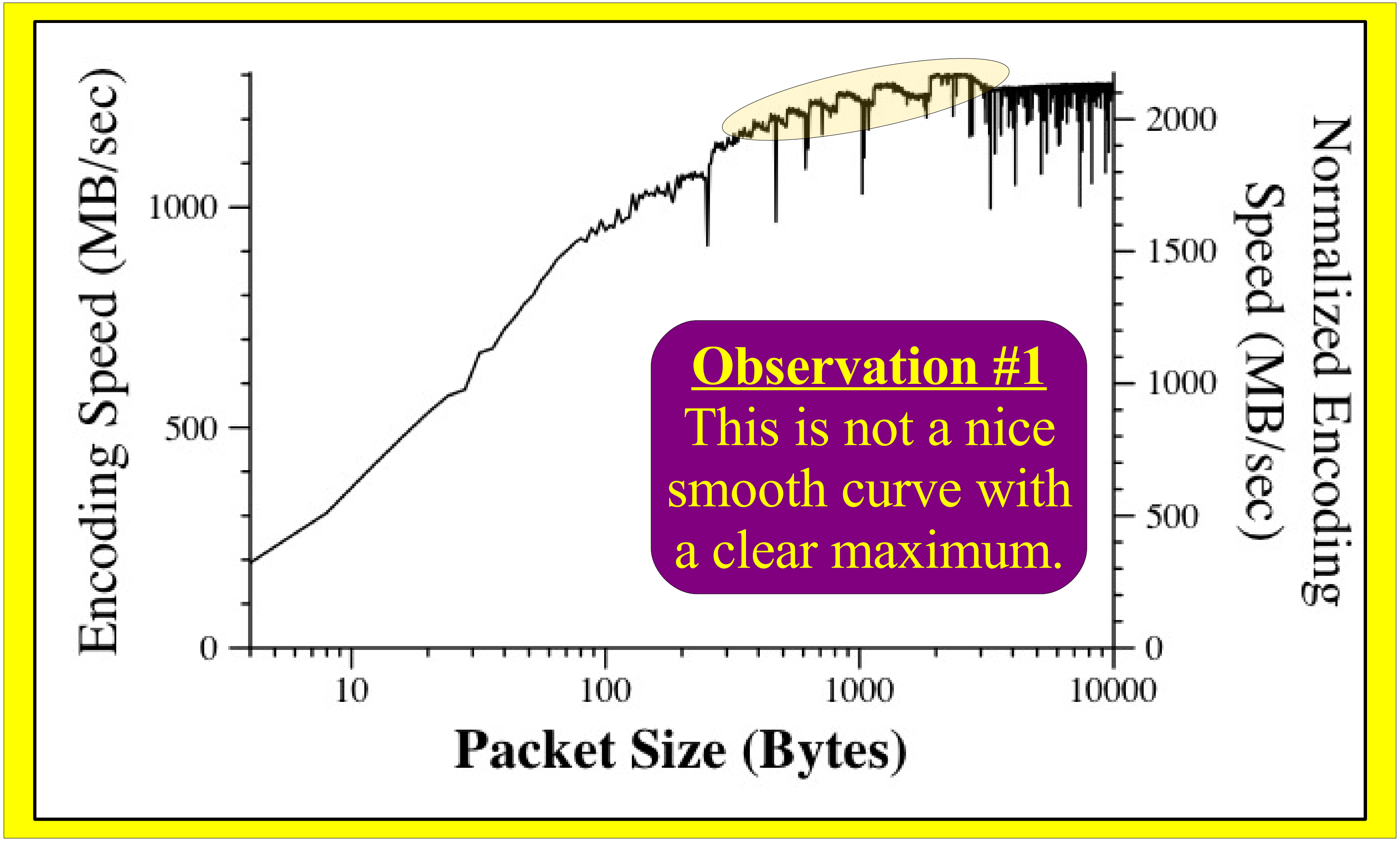
Jianqiang Luo
Lihao Xu
(Wayne State)

Zooko Wilcox-O'Hearn Allmydata

Usenix FAST
February 27, 2009

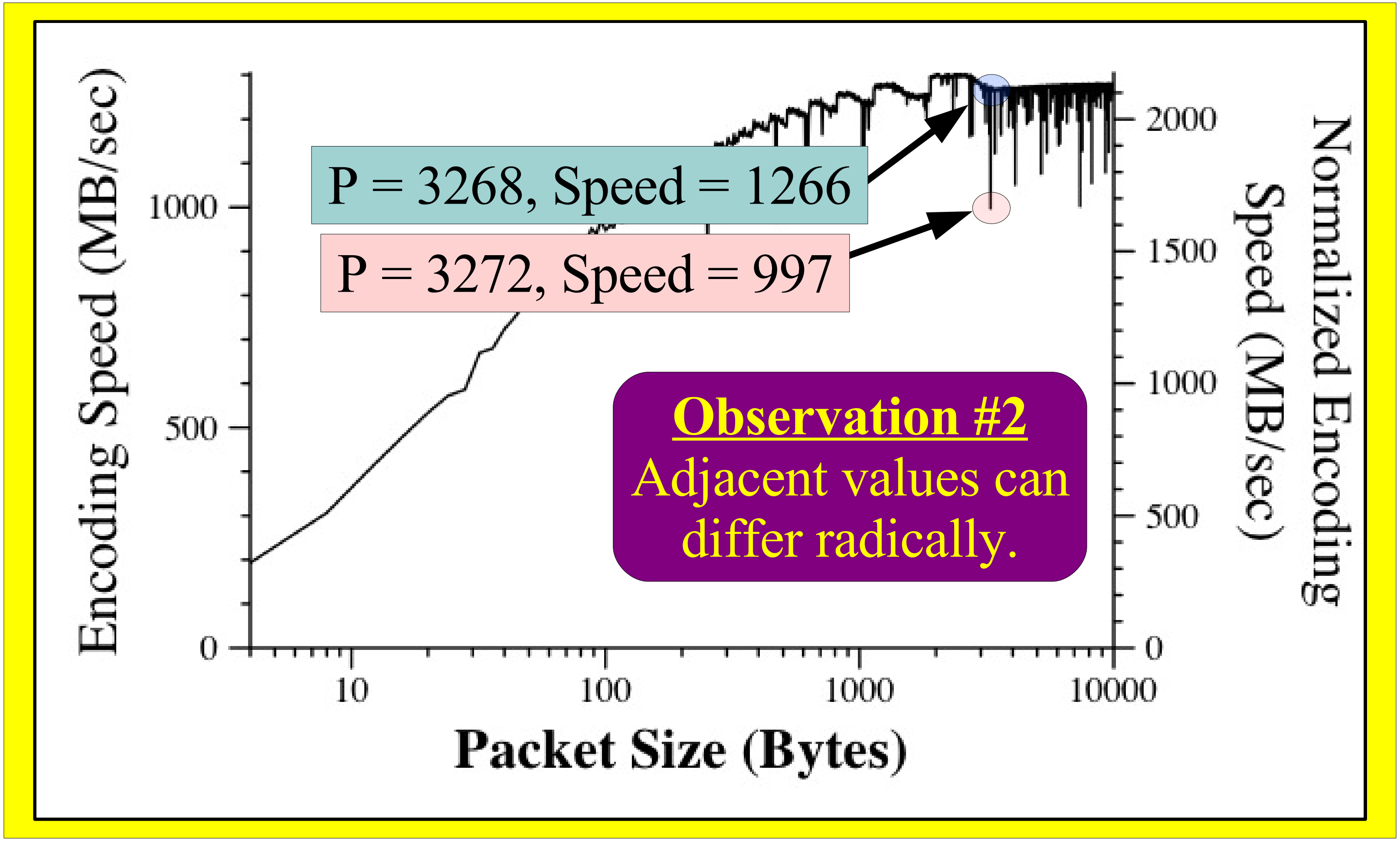
Cache Effects: The packet size.

RDP - [6,2]. w = 6 on MacBook.



Cache Effects: The packet size.

RDP - [6,2]. *w* = 6 on MacBook.



Cache Effects: The packet size.

Result

A heuristic search algorithm to find the "best" packet size.

Remaining graphs always show performance of the best packet size.