Erasure Coding: Views from 10,000 Feet and Through a Magnifying Glass

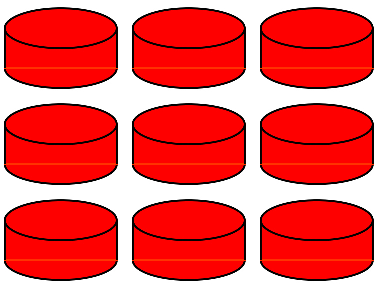
James S. Plank

Professor
Department of EECS
University of Tennessee
plank@cs.utk.edu

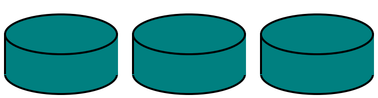
CCGSC
September 16, 2008

What is an Erasure Code?

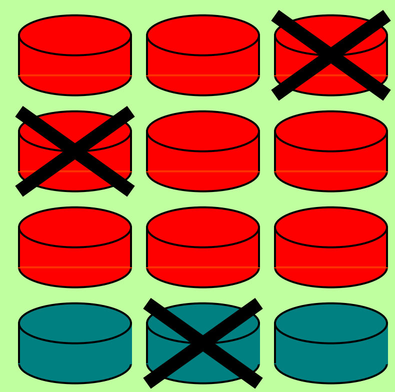
A technique that lets you take *k* pieces of data:



Encode them onto *m* additional pieces of data:



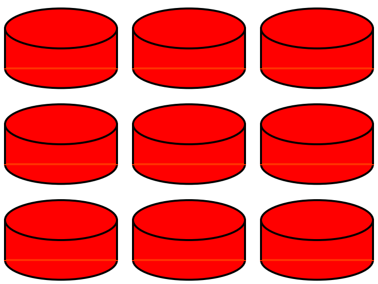
And have the entire system be resilient to up to *m* failures:



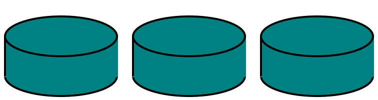
What is an Erasure Code?

Or, alternatively...

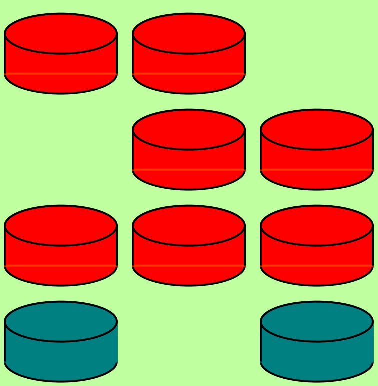
A technique that lets you take *k* pieces of data:



Encode them onto *m* additional pieces of data:



And rebuild the original *k* pieces of data from as few as *k* of the collection:

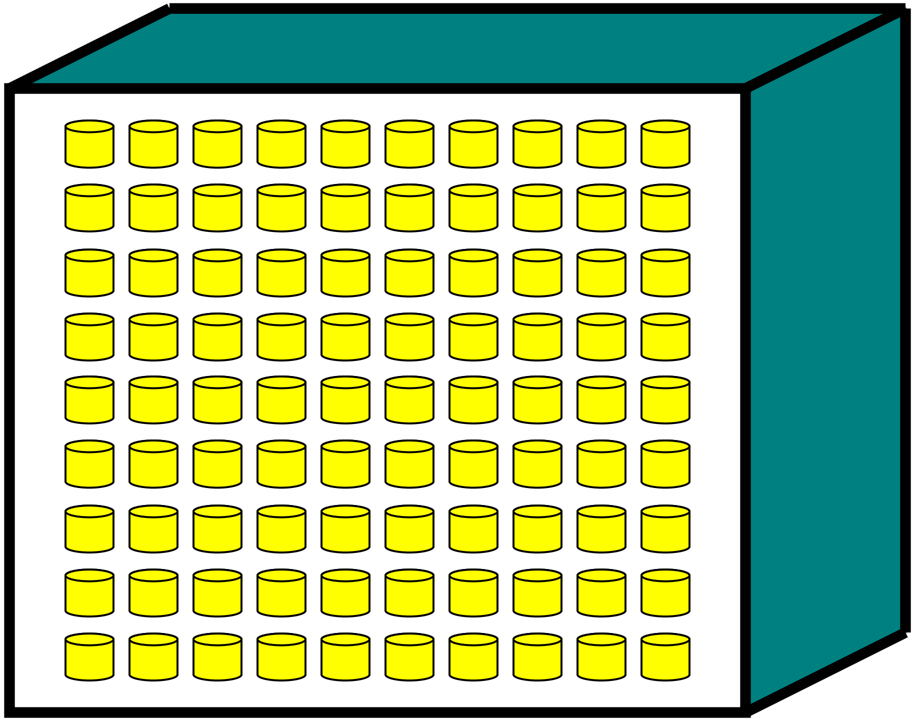


When are they useful?

Anytime you need to tolerate failures.

For example:

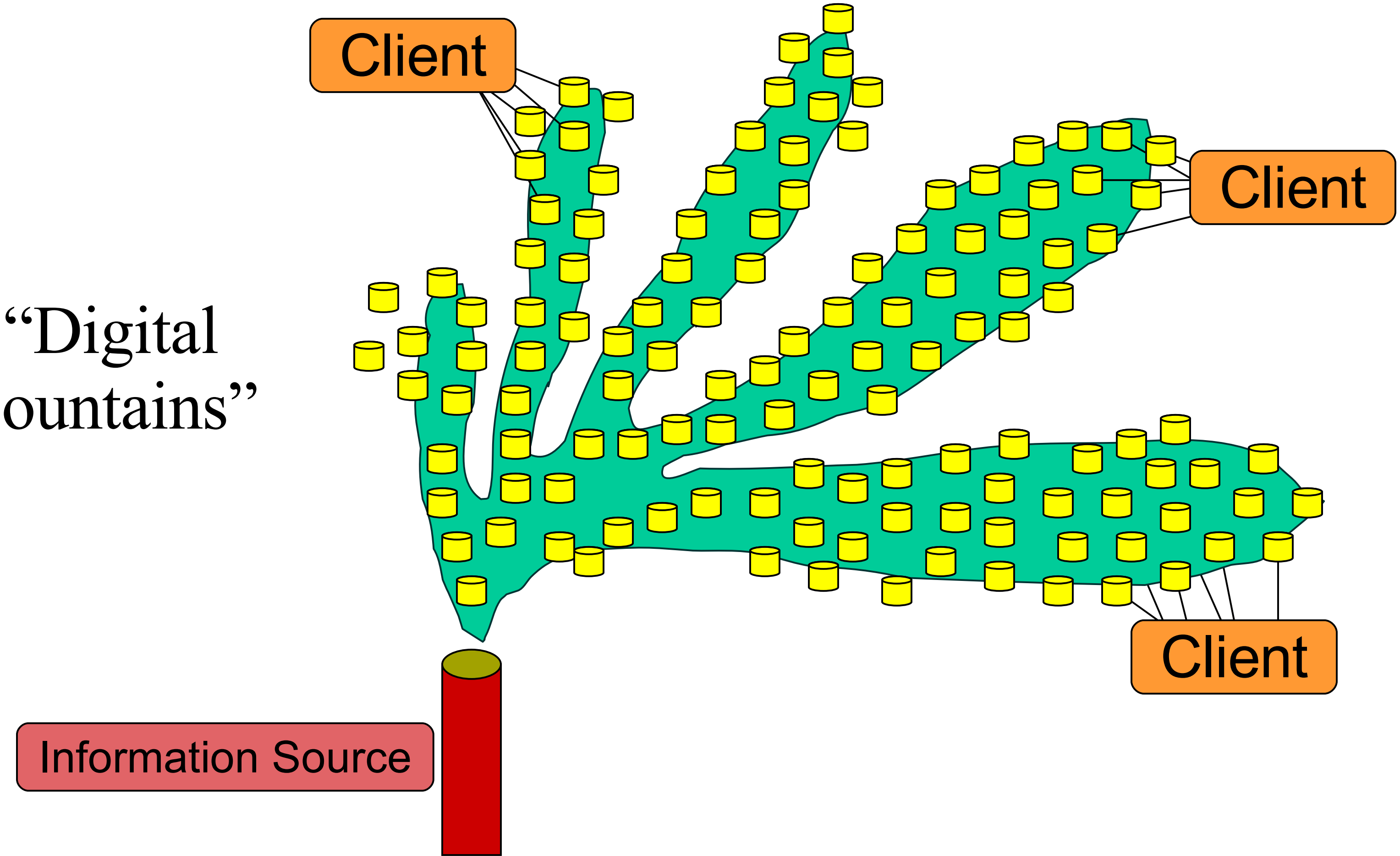
Disk Array Systems
(RAID-5/RAID-6)



When are they useful?

Anytime you need to tolerate failures.

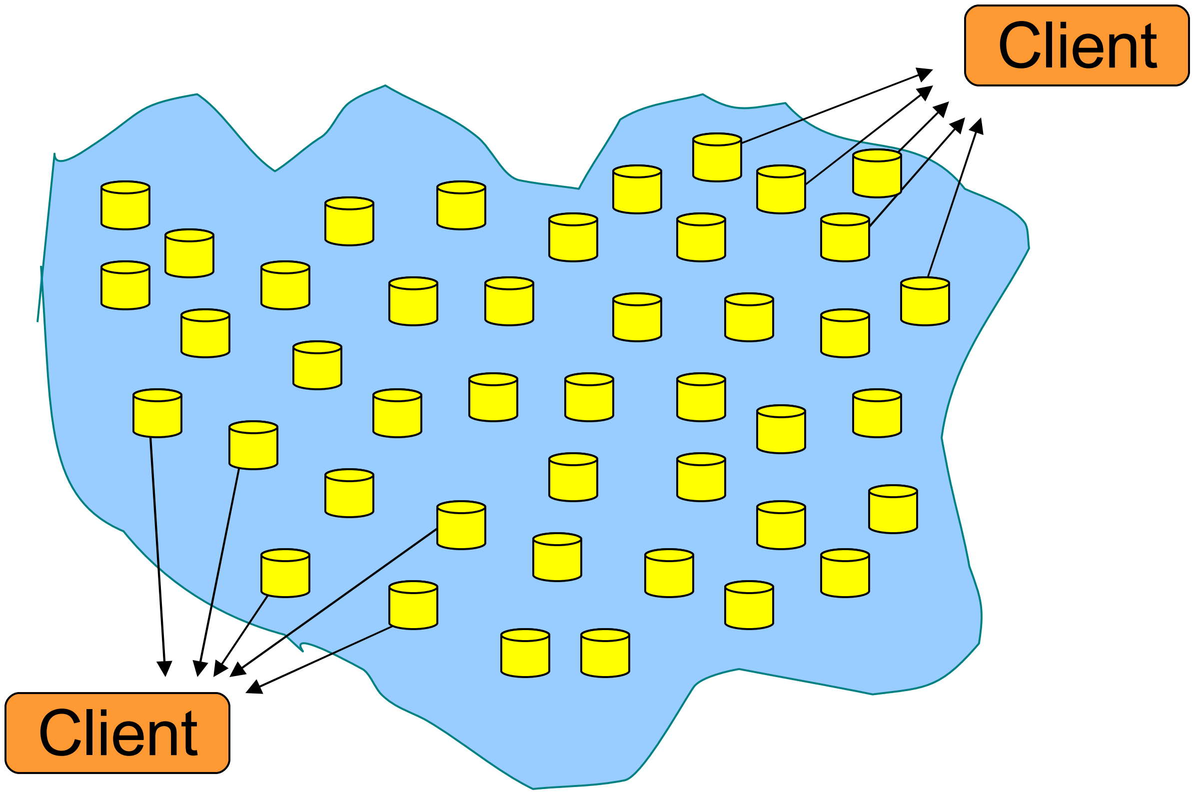
“Digital Fountains”



When are they useful?

Anytime you need to tolerate failures.

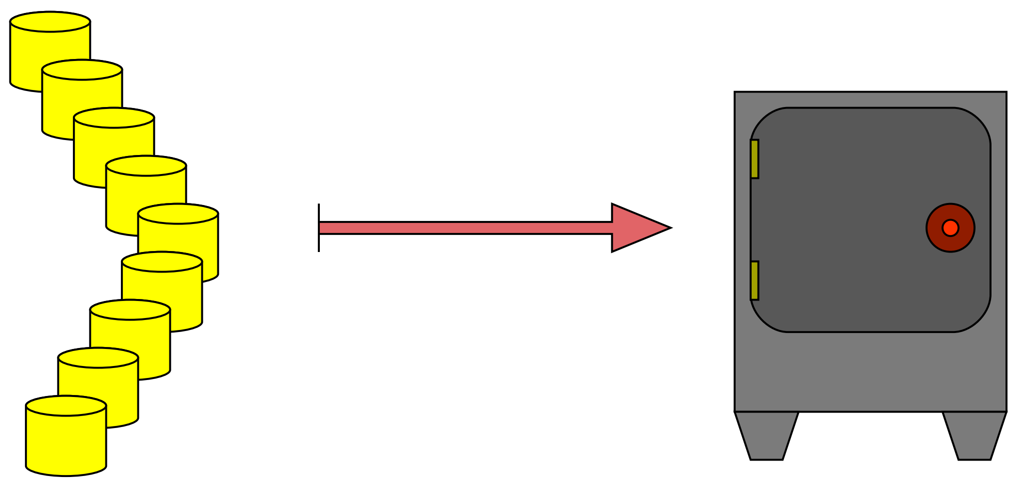
Distributed Data
or
Object Stores:



When are they useful?

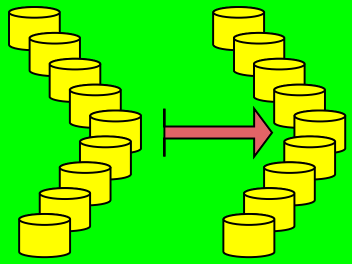
Anytime you need to tolerate failures.

Archival Storage.



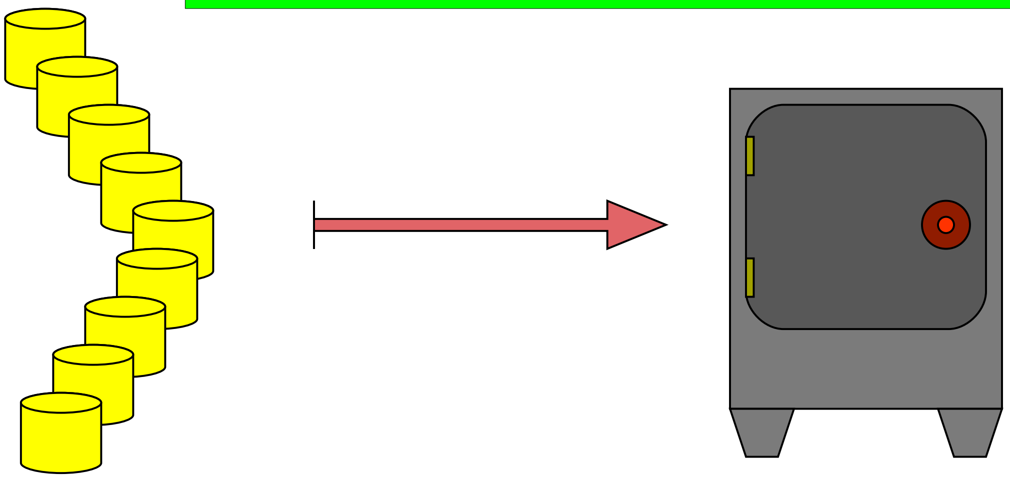
When are they useful?

Anytime you need to tolerate failures.



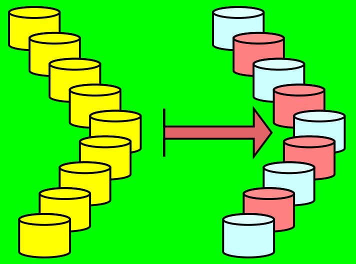
Replicas:
Can lose data
with two failures.

Archival
Storage.



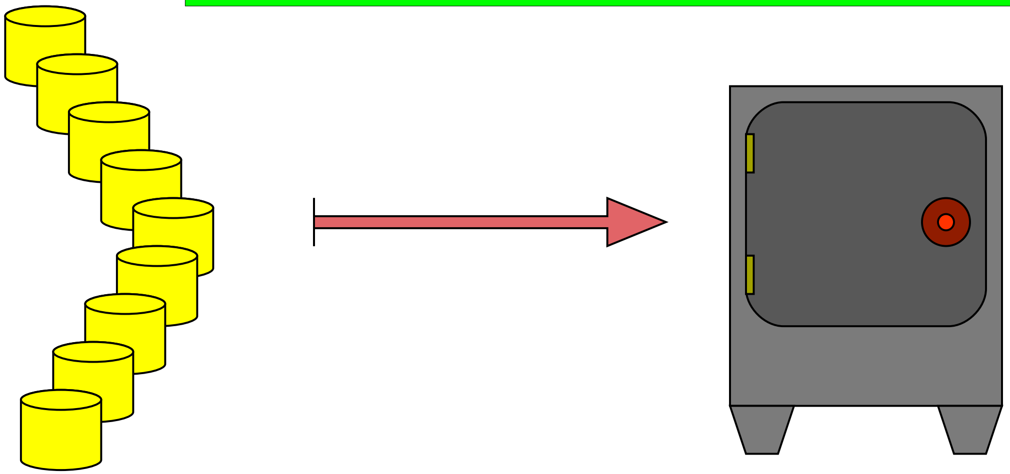
When are they useful?

Anytime you need to tolerate failures.



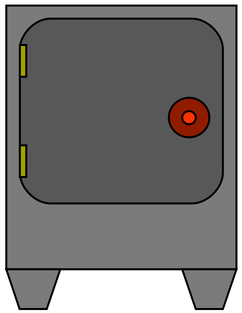
__Erasure Codes__:
Only lose data
with *m* failures.

Archival
Storage.

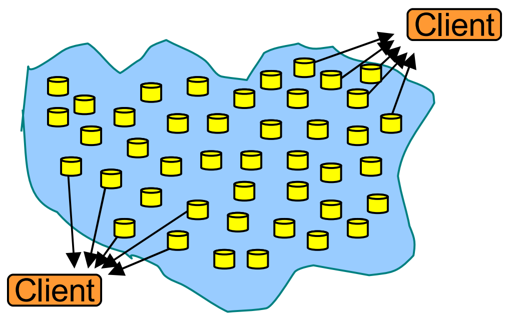


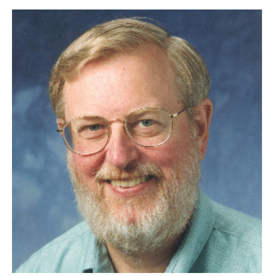
Why should we care at CCGSC?

Archival Storage:

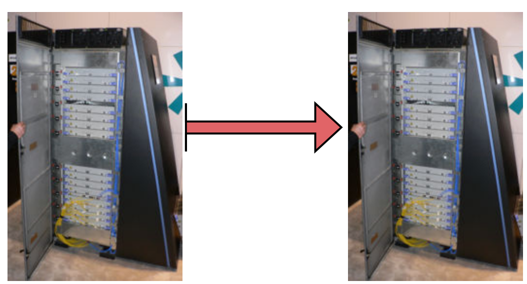


Distributed Object Stores:



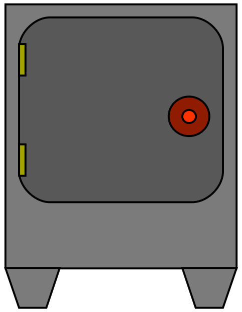


Diskless Checkpointing Systems:



In Fact...

If we've already reached the point where our data to store is larger than our storage capacity...

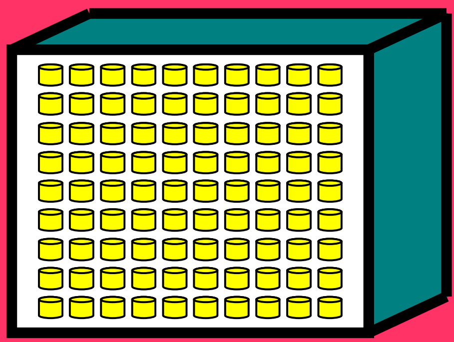


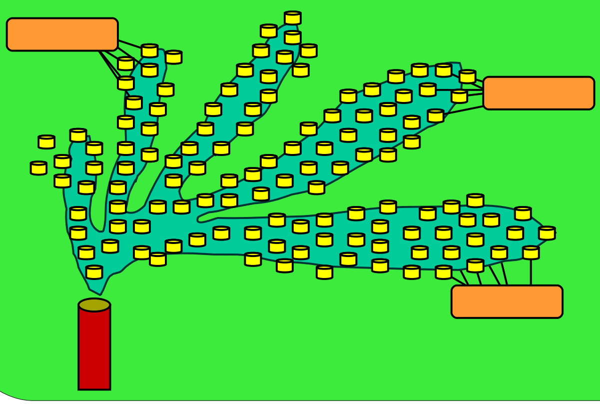
Then we should be living in a world where erasure coding, rather than replication, is the norm.

What is the state of the world with respect to erasure coding?

Noisy Communication Lines: 1960

Disk Arrays 1987





Digital Fountains 1997

A mess!

What is the state of the world with respect to erasure coding?

Noisy Communication Lines: 1960

<u>Reed-Solomon Coding</u>

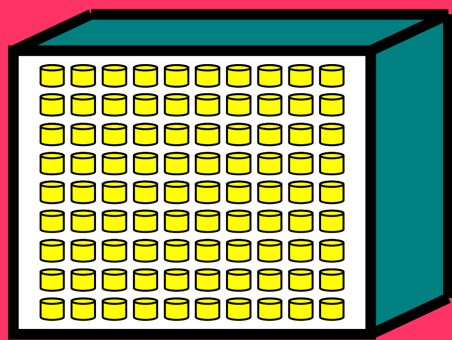
- Any *k*
- Any *m*

Rebuild with any *k* blocks ("MDS").

<u>Expensive</u>: *(k-1)* XORs plus *k* Galois Field Multiplications.

What is the state of the world with respect to erasure coding?

Disk Arrays
1987



<u>RAID-5 / RAID-6</u>

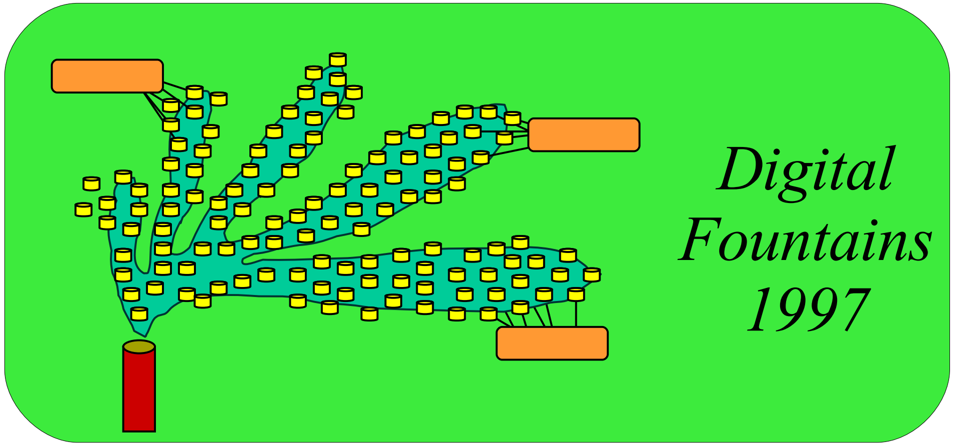
- k typically < *20*
- m = *1* or *2*.

Rebuild with any k blocks ("MDS").

<u>Faster</u>: Approximately *(k-1)* XORs per coding word.

- RAID-5: 1987
- EVENODD: 1996
- RDP: 2004
- Liberation: 2008

What is the state of the world with respect to erasure coding?



<u>LDPC Codes</u>

"Low Density Parity Check"

k, m are very large.

Distinctly non-MDS.

<u>Blazingly Fast</u>: *O(1)* per coding word ("Low Density").

- Tornado Codes: 1997
- LT Codes: 2002
- Raptor Codes: 2003
- Staircase Codes: 2008

Who is doing Erasure Coding?

- Products:
  - *RAID*: Netapp, Panasas, EMC, etc.
  - *Deduplication*: Data Domain.
  - *Archival*: Allmydata, Permabit.
  - *Wide-Area Distribution*: Cleversafe.
- Research:
  - Microsoft (Pyramid Codes).
  - IBM (Weaver, Hover Codes).
  - HP (1-Row Horizontal Codes).

What Have I Been Doing?

- RAID-6 Liberation Codes:
  - FAST 2008, NCA 2008.
  - Excellent performance, non-patented, new codes.
- $A(x) = B$ in GF(2):
  - An NP-Complete Problem?
- Jerasure: Open Source Coding Library:
  - Reed-Solomon, RAID-6, others.

Reed-Solomon Coding Primer

Encoding is a matrix-vector product: All elements are *w*-bit words.

$k+m \leq 2^w$

$$
k+m\left\{\overbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ X_{00} & X_{01} & X_{02} & X_{03} & X_{04} \\ X_{10} & X_{11} & X_{12} & X_{13} & X_{14} \\ X_{20} & X_{21} & X_{22} & X_{23} & X_{24} \end{bmatrix}}^{k}\right. * \left.\begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ D_3 \\ D_4 \end{bmatrix}\right\}k = \begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ D_3 \\ D_4 \\ C_0 \\ C_1 \\ C_2 \end{bmatrix}
$$

[Generator Matrix]^T

Data
(Message)

Data+“Parity”
(Codeword)

Reed-Solomon Coding Primer

- Addition is XOR.
- Multiplication in $GF(2^w)$.
  - Table lookup for $w = 8$.
  - Discrete logs for $w = 16$.
  - More complex for $w = 32$.
- Decoding = Matrix Inversion & Recalculation

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ X_{00} & X_{01} & X_{02} & X_{03} & X_{04} \\ X_{10} & X_{11} & X_{12} & X_{13} & X_{14} \\ X_{20} & X_{21} & X_{22} & X_{23} & X_{24} \end{bmatrix} * \begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ D_3 \\ D_4 \end{bmatrix} = \begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ D_3 \\ D_4 \\ C_0 \\ C_1 \\ C_2 \end{bmatrix}$$

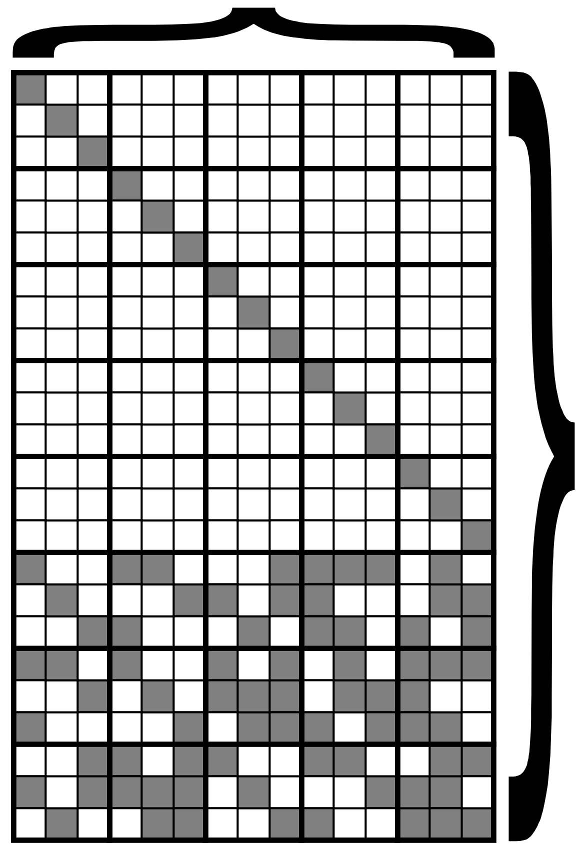
Cauchy Reed-Solomon Coding

Explode matrix by a factor of w in both dimensions:

$k+m \leq 2^w$

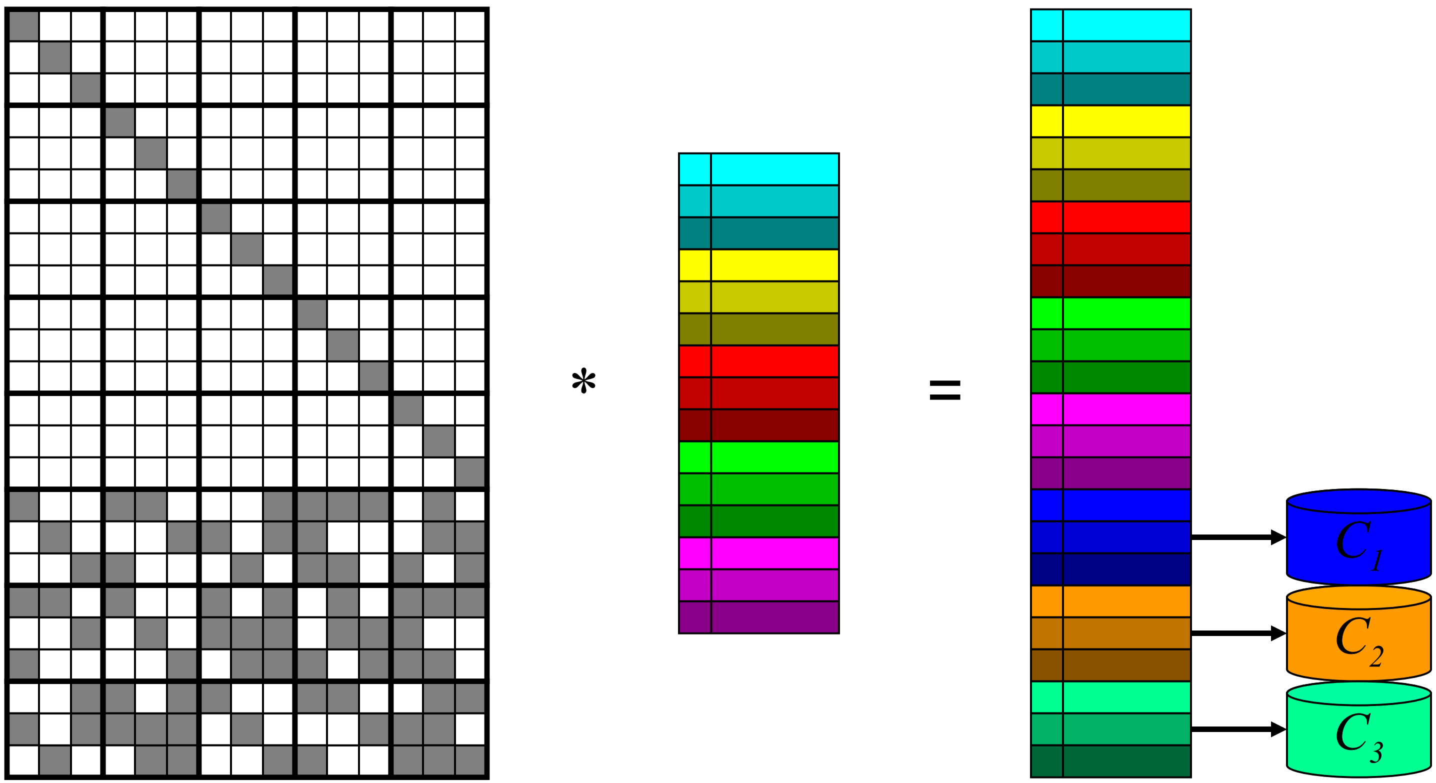
$$k+m\left\{\overbrace{\begin{array}{|c|c|c|c|c|}\hline 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 \\ \hline X_{00} & X_{01} & X_{02} & X_{03} & X_{04} \\ \hline X_{10} & X_{11} & X_{12} & X_{13} & X_{14} \\ \hline X_{20} & X_{21} & X_{22} & X_{23} & X_{24} \\ \hline \end{array}}^{k}\right.$$

$\longrightarrow$



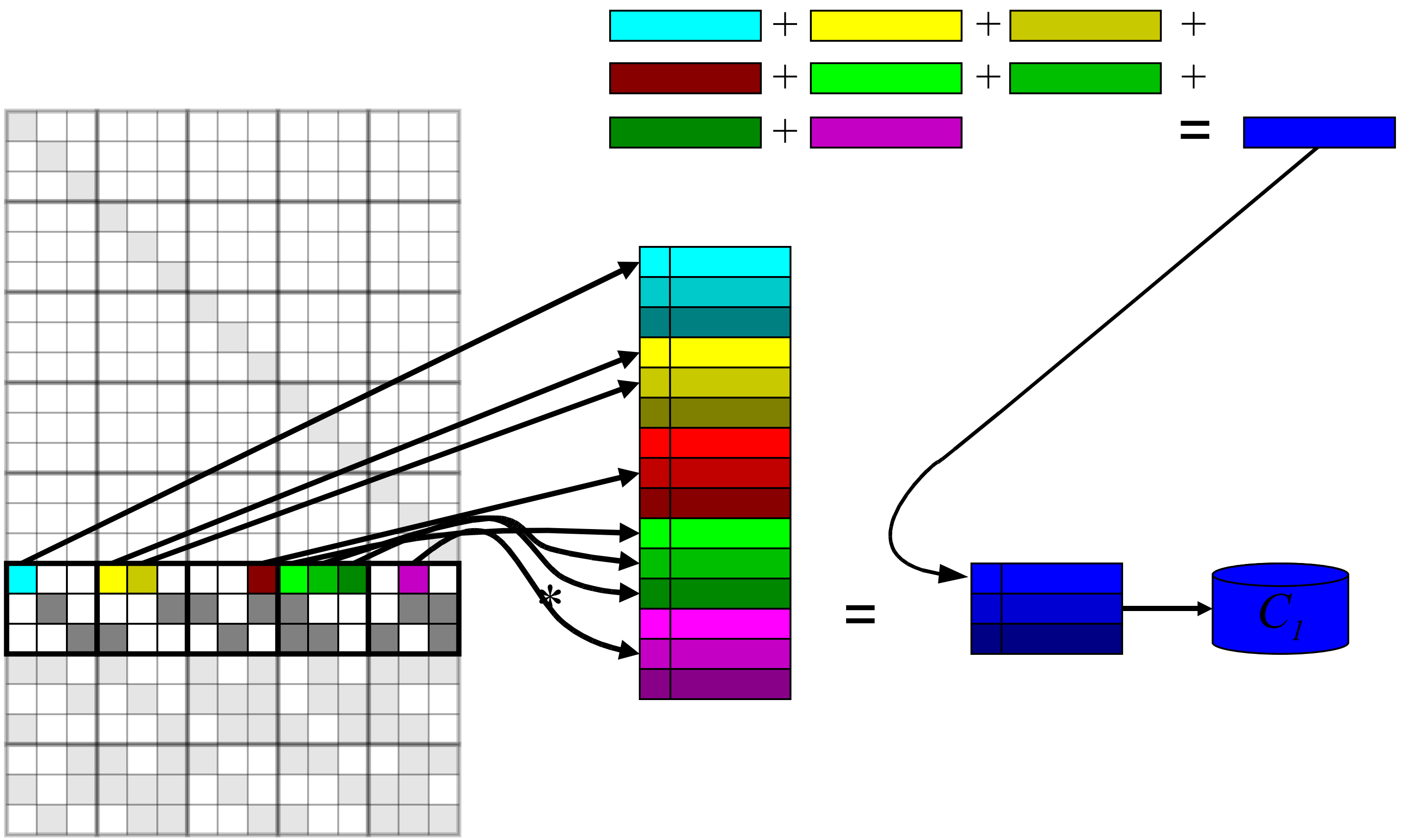
Cauchy Reed-Solomon Coding

Allows you to break data into large *packets*, and encode with XOR.

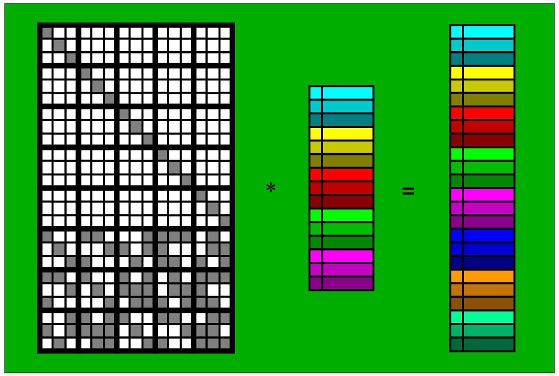


Cauchy Reed-Solomon Coding

Allows you to break data into large *packets*, and encode with XOR.



Cauchy Reed-Solomon Coding



- You want sparse matrices.
  - Small w better than large?
  - Can optimize for RAID-6.
- Do extra XOR's make up for $GF(2^w)$ Multiplications?
- Are large packets a big win?

Jerasure

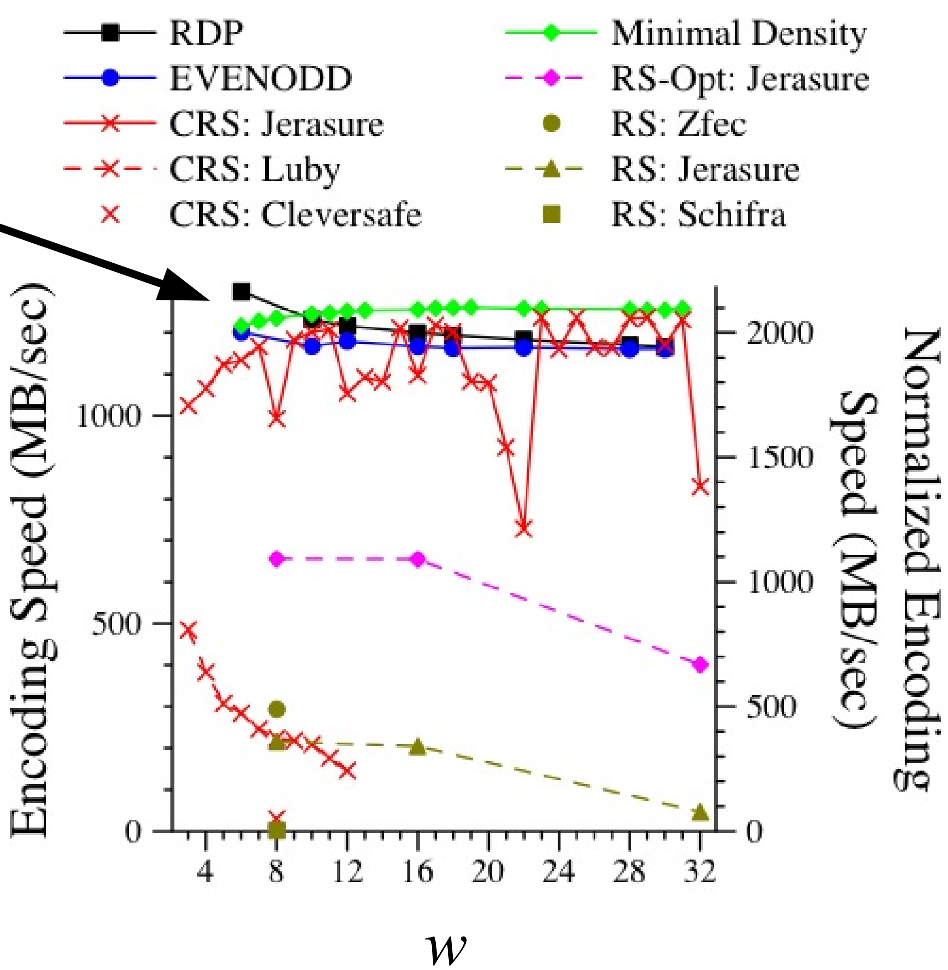
- *Open Source Library for C/C++*
  – Reed-Solomon Codes
  – Cauchy Reed-Solomon Codes
  – General Bit-Matrix Codes
  – Optimized Reed-Solomon for RAID-6
  – Optimized CRS for RAID-6
  – RAID-6 Liberation Codes (Minimal Density)
  – Version 1.2, September, 2008

Encoding Performance:

- Split a 1GB file into *k* pieces & encode into *m*.
- Compare jerasure with open source libraries:
  - *Schifra* (RS: C++*)
  - *Zfec* (RS: C – descendent of Rizzo)
  - *Luby* (CRS: C)
  - *Cleversafe* (CRS: Java)
- Four configurations: [*k*,*m*]
  - RAID-6: [6,2], [14,2]
  - [12,4], [10,6]

[6,2] Encoding Performance

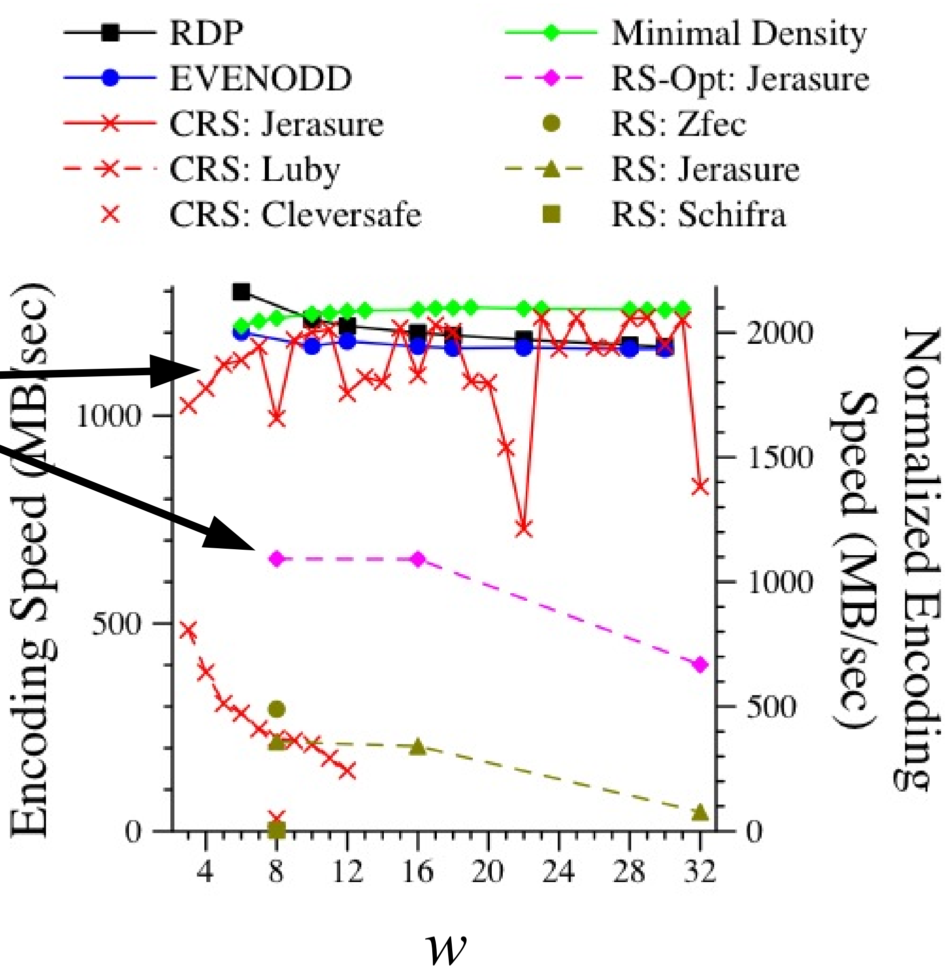
<u>Conclusion #1:</u>
Special-Purpose RAID-6 codes rock.



[6,2] Encoding Performance

<u>Conclusion #1:</u>
Special-Purpose RAID-6 codes rock.

<u>Conclusion #2:</u>
Optimized CRS & RS codes perform better.



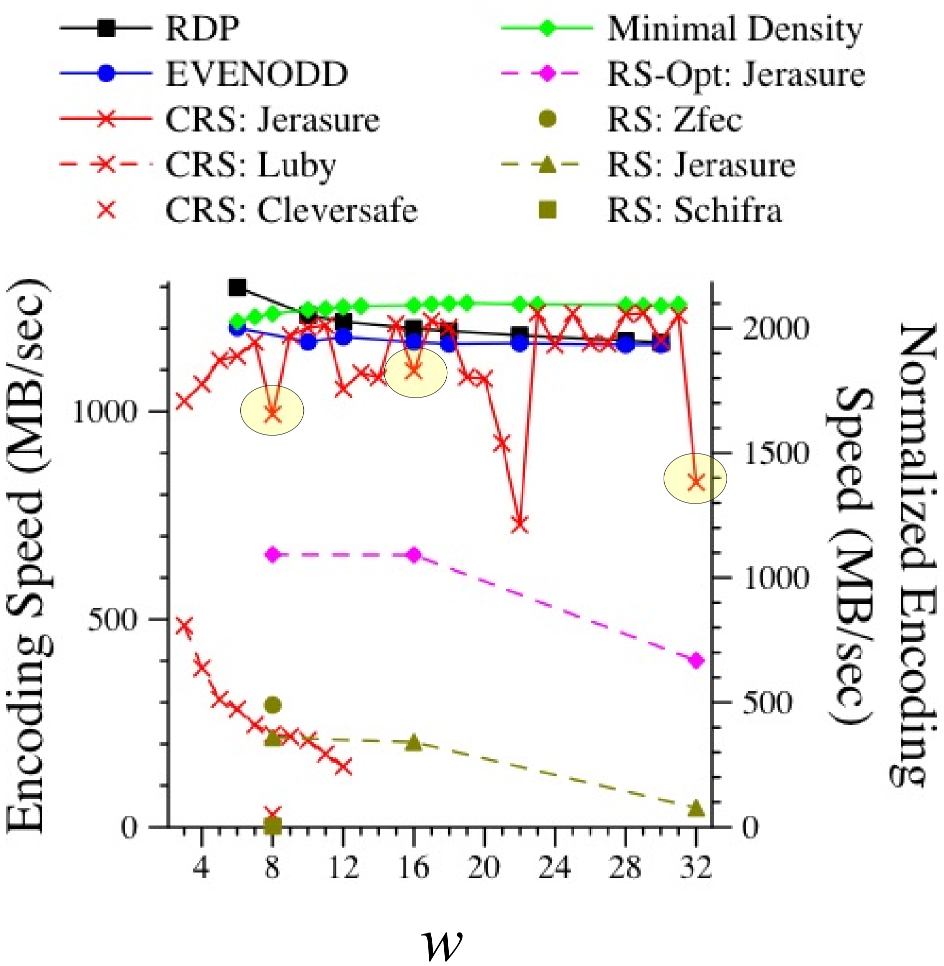
[6,2] Encoding Performance

Conclusion #1:
Special-Purpose RAID-6 codes rock.

Conclusion #2:
Optimized CRS & RS codes perform better.

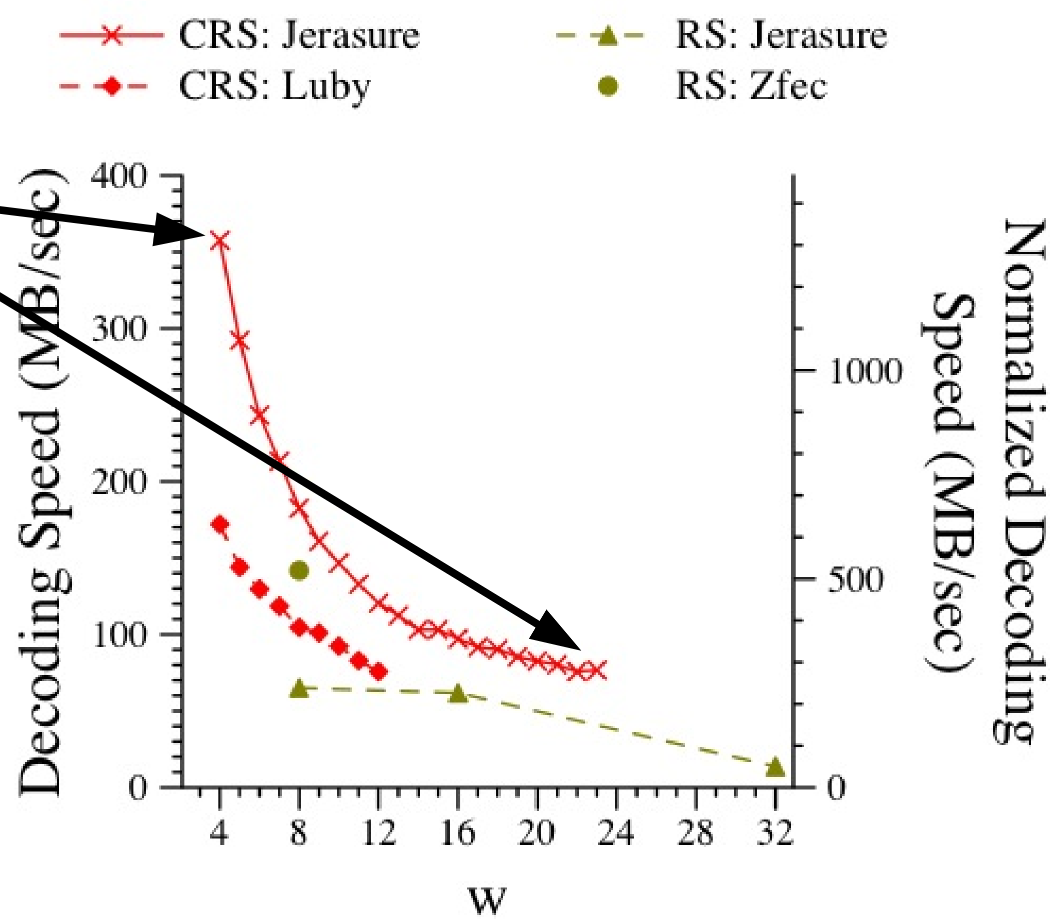
Conclusion #3:
The choice of *w* matters!

(*w* = 8, 16, 32 in CRS...)



[12,4] Encoding Performance

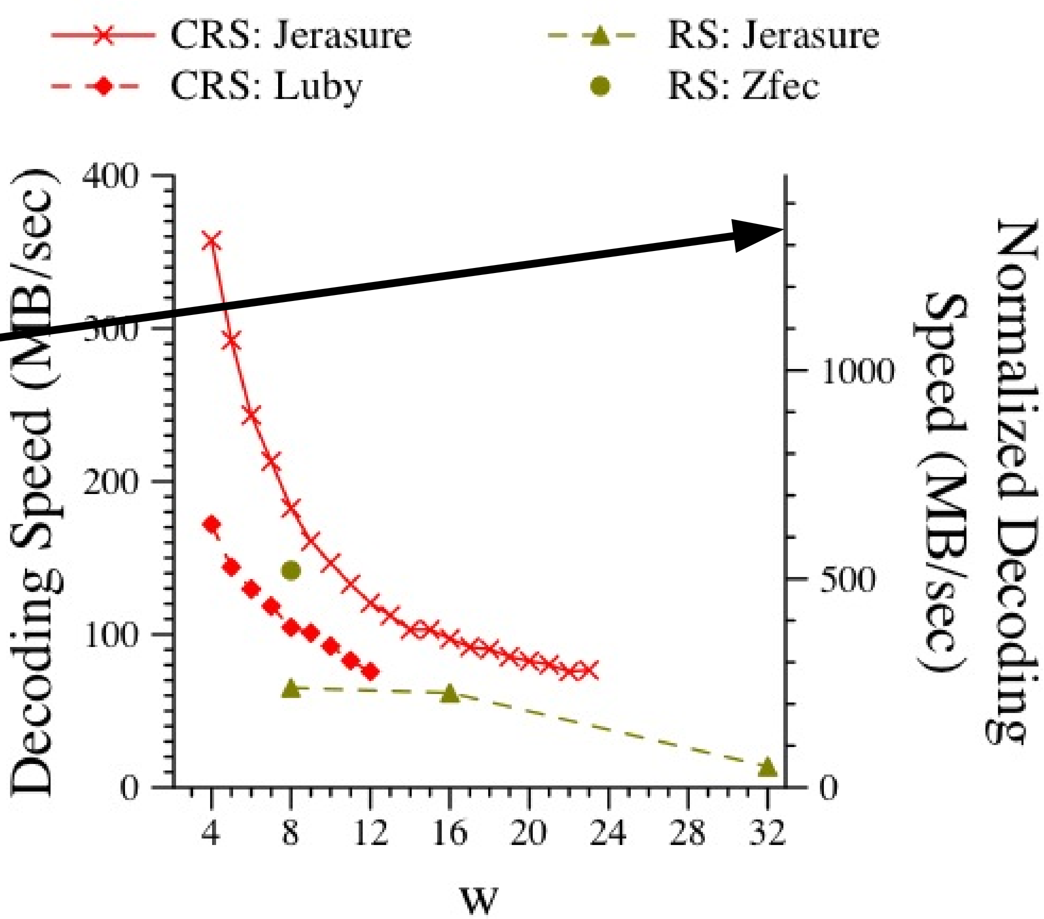
<u>Conclusion #1:</u>
XOR's win, but not if you're sloppy.



[12,4] Encoding Performance

Conclusion #1:
XOR's win, but not if you're sloppy.

Conclusion #2:
Normalized performance bad compared to RAID-6.

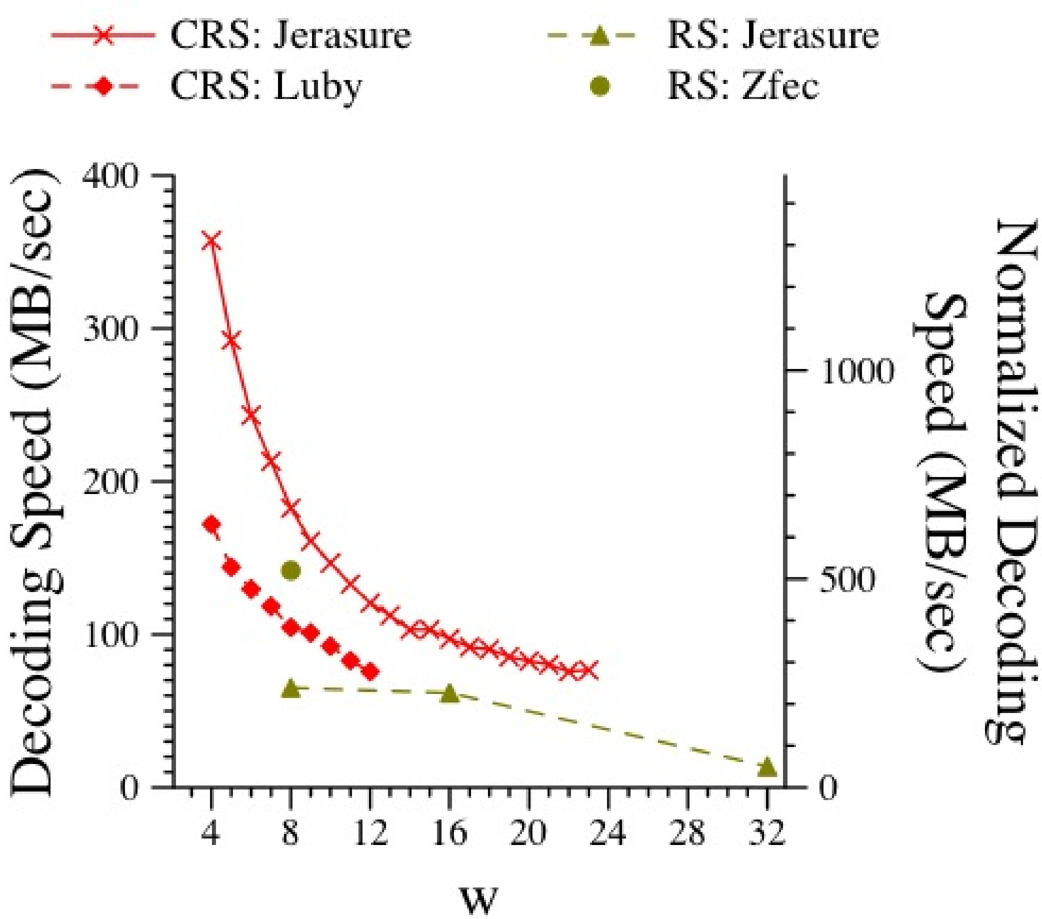


[12,4] Encoding Performance

Conclusion #1:
XOR's win, but not if you're sloppy.

Conclusion #2:
Normalized performance bad compared to RAID-6.

Conclusion #2A:
This is the place where research should be focused.



There's a whole lot more...

But what I'm hoping you've gotten out of this:

- Think coding instead of replication.
- There are good open-source tools.
- There is immediate opportunity for research in this area.

Erasure Coding: Views from 10,000 Feet and Through a Magnifying Glass

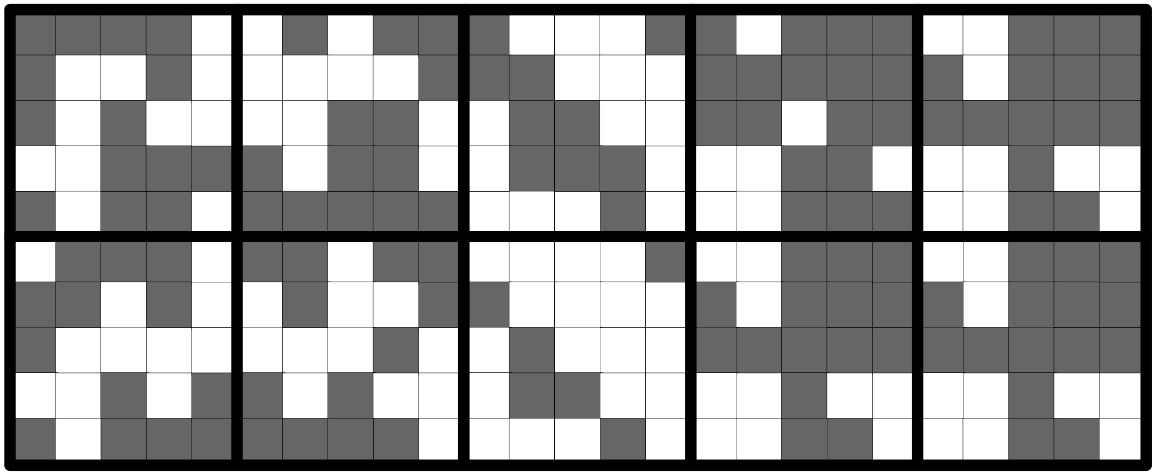
James S. Plank

Professor
Department of EECS
University of Tennessee
plank@cs.utk.edu

CCGSC
September 16, 2008

$A(x) = B$ in $GF(2)$

- Inverted matrices for Liberation decoding are *not* sparse.

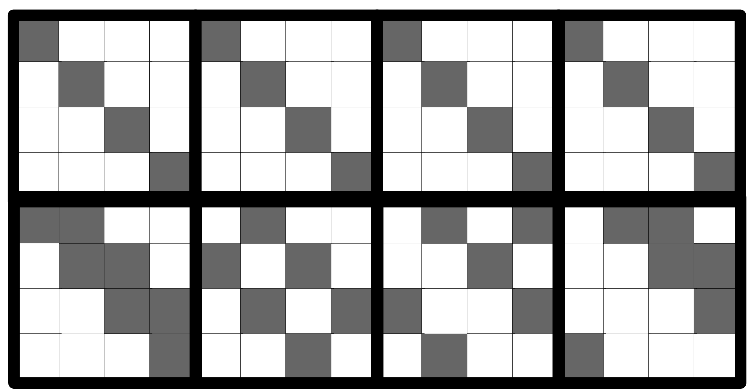


A decoding matrix for $k=5$, $w=5$

This is a problem for efficient decoding:
12.3 XOR's per coding word instead of 4 (optimal)

$A(x) = B$ in $GF(2)$

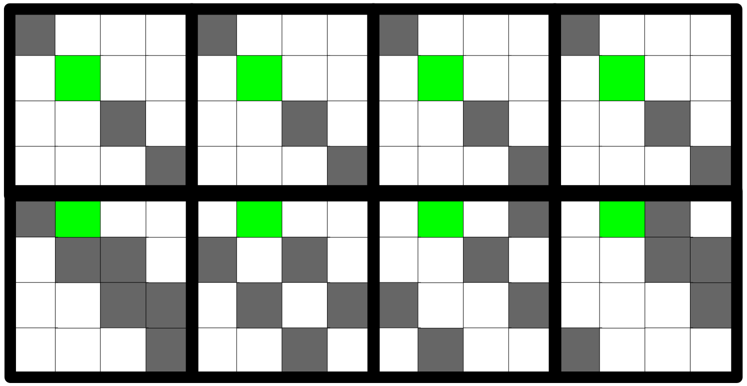
- Take inspiration from RDP – intermediate coding elements may be used as starting points.



RDP matrices for $k = w = 4$.

$A(x) = B$ in $GF(2)$

- Take inspiration from RDP – intermediate coding elements may be used as starting points.

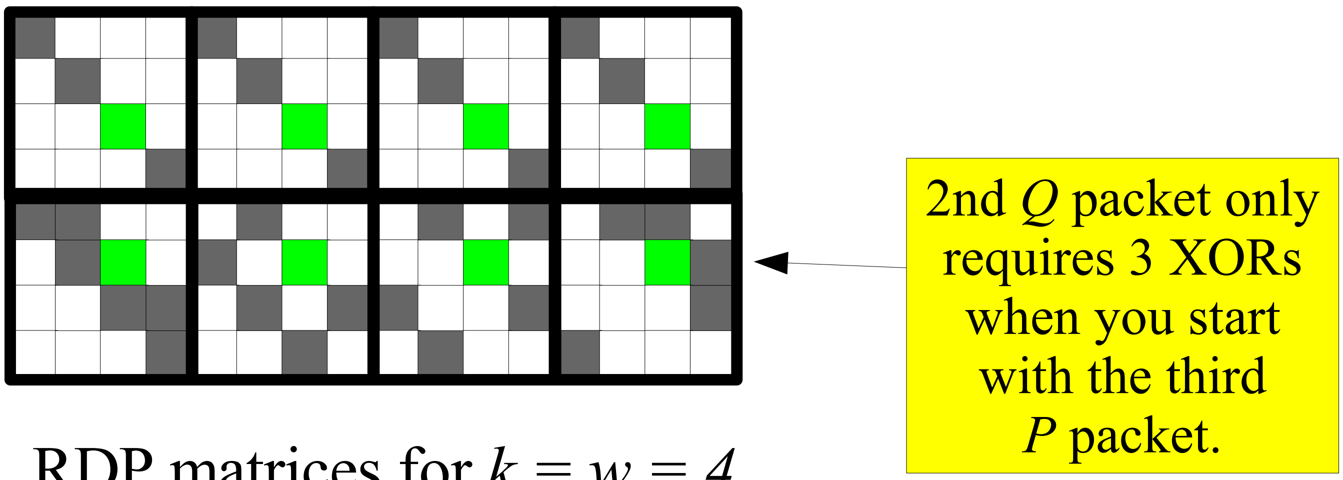


RDP matrices for $k = w = 4$.

First Q packet only requires 3 XORs when you start with the second P packet.

$A(x) = B$ in $GF(2)$

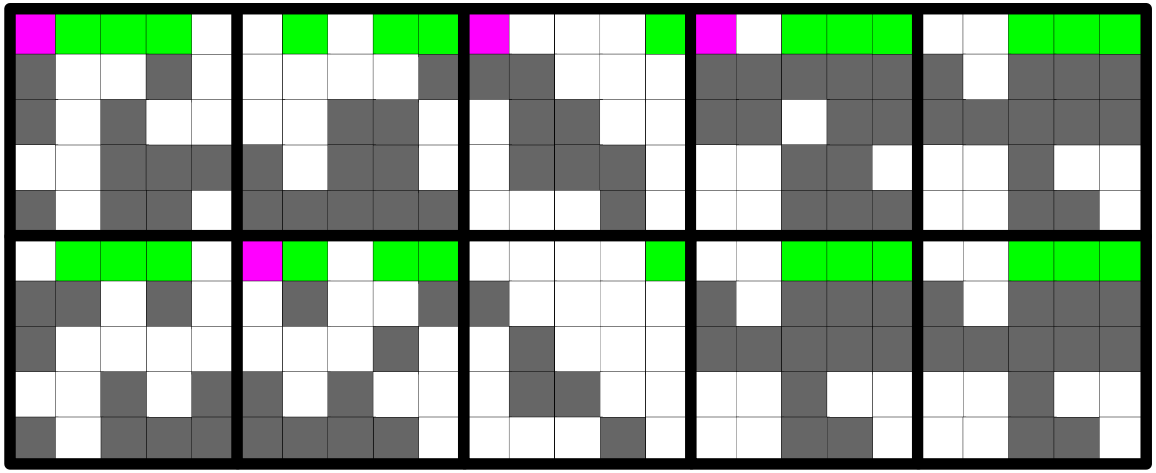
- Take inspiration from RDP – intermediate coding elements may be used as starting points.



RDP matrices for $k = w = 4$.

$A(x) = B$ in $GF(2)$

- The idea: you use intermediate results to perform a bit-matrix vector product with fewer XOR's than the number of ones.

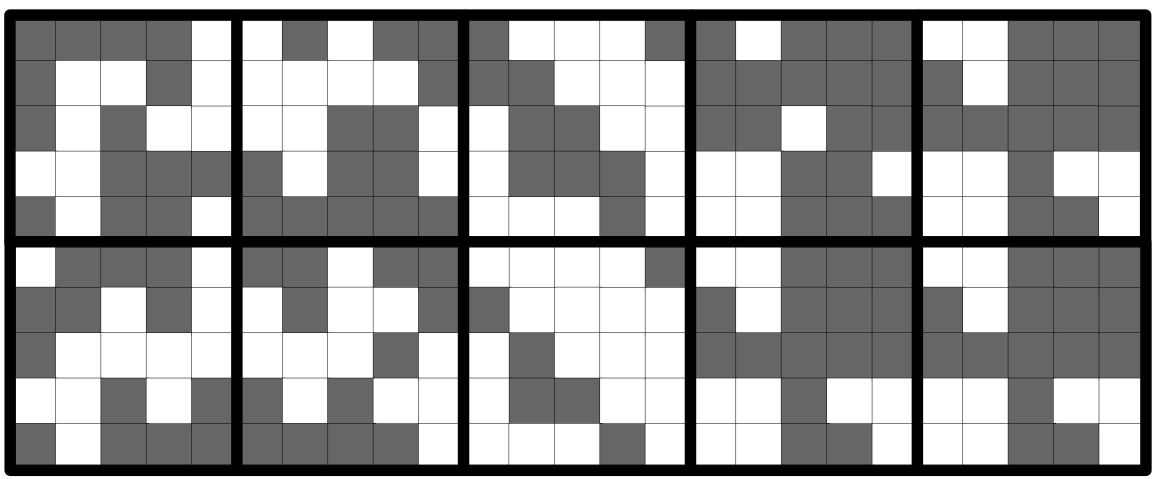


A decoding matrix for $k=5$, $w=5$

Row 0: 4 XORs instead of 15 when you start with row 5.

The Algorithm

- Two arrays:
  - **Start**, initialized to -1
  - **XOR**, initialized to # ones minus one.

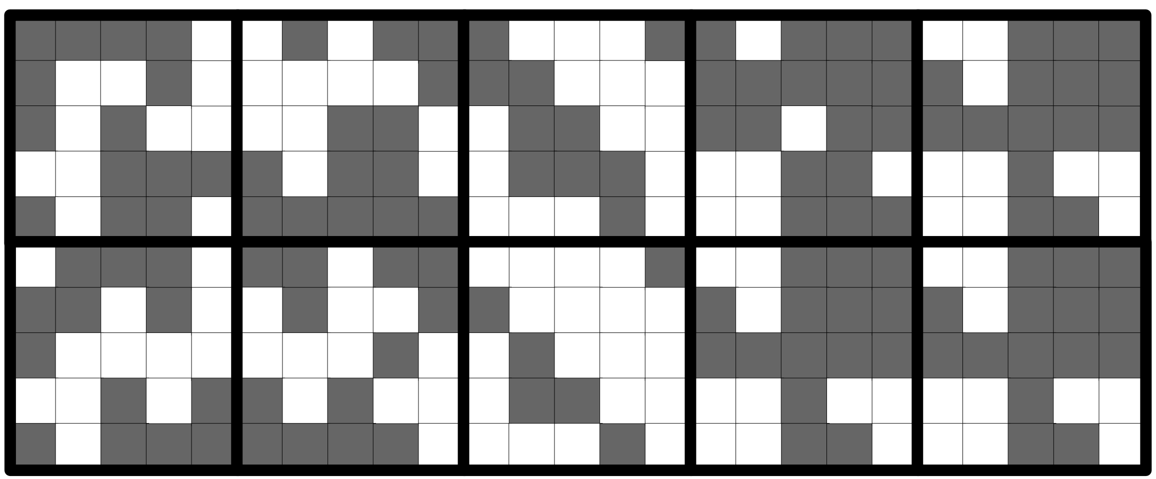


A decoding matrix for *k=5, w=5*

Start	XOR
-1	15
-1	13
-1	14
-1	11
-1	13
-1	13
-1	13
-1	12
-1	7
-1	12

The Algorithm

- Find row with minimal **XOR**.
  - That row will be created from scratch with the given number of XORs.

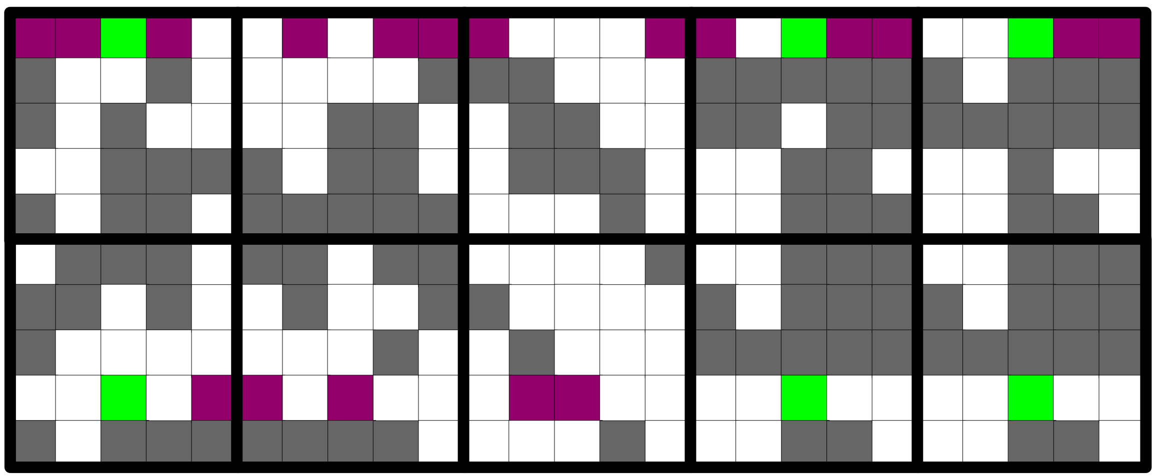


A decoding matrix for $k=5$, $w=5$

Start	XOR
-1	15
-1	13
-1	14
-1	11
-1	13
-1	13
-1	13
-1	12
-1	7
-1	12

The Algorithm

- For every other row:
  – See if fewer XOR's are required if that row is used as a starting point and update the arrays accordingly.



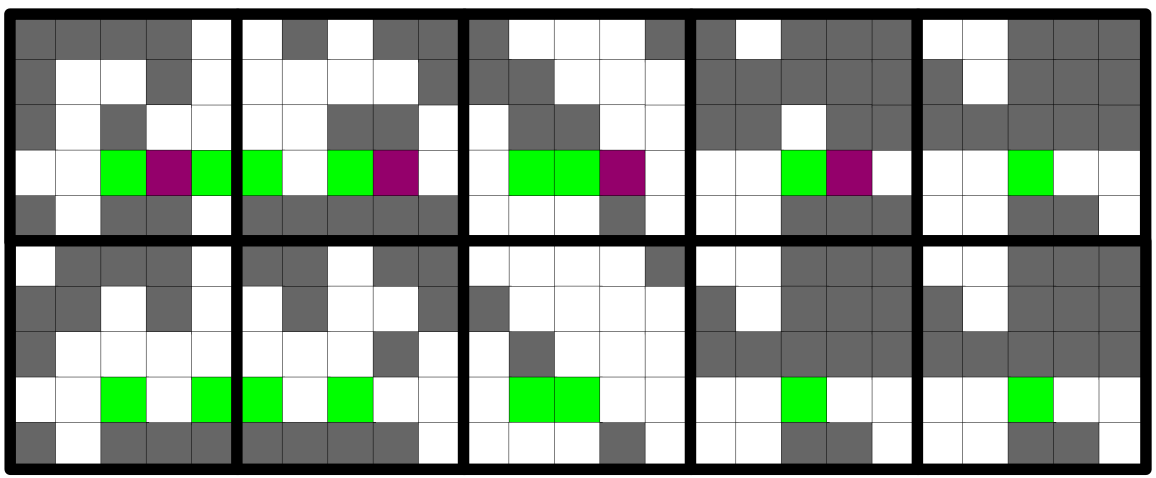
A decoding matrix for *k=5, w=5*

Start	XOR
-1	15
-1	13
-1	14
-1	11
-1	13
-1	13
-1	13
-1	12
-1	7
-1	12

E.g. Creating row 0 from row 8 requires 18 XORs, so no update.

The Algorithm

- For every other row:
  - See if fewer XOR's are required if that row is used as a starting point and update the tables accordingly.



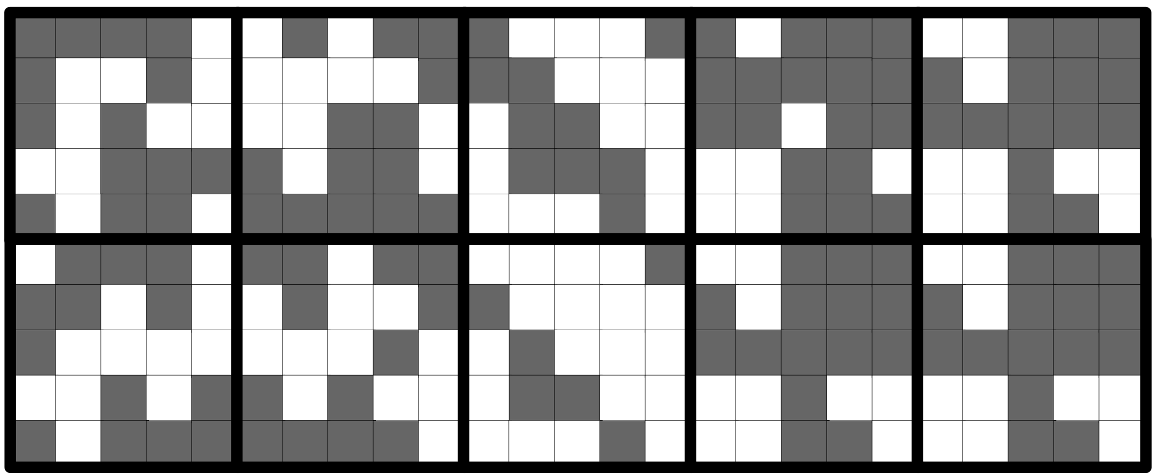
A decoding matrix for $k=5$, $w=5$

Start	XOR
-1	15
-1	13
-1	14
8	4
-1	13
-1	13
-1	13
-1	12
-1	7
-1	12

E.g. However, row 3 only requires 4 XORs: Update the tables.

The Algorithm

- Repeat the process
  - Find row with minimum **XORs**
  - Update **Start/XOR** of other rows.

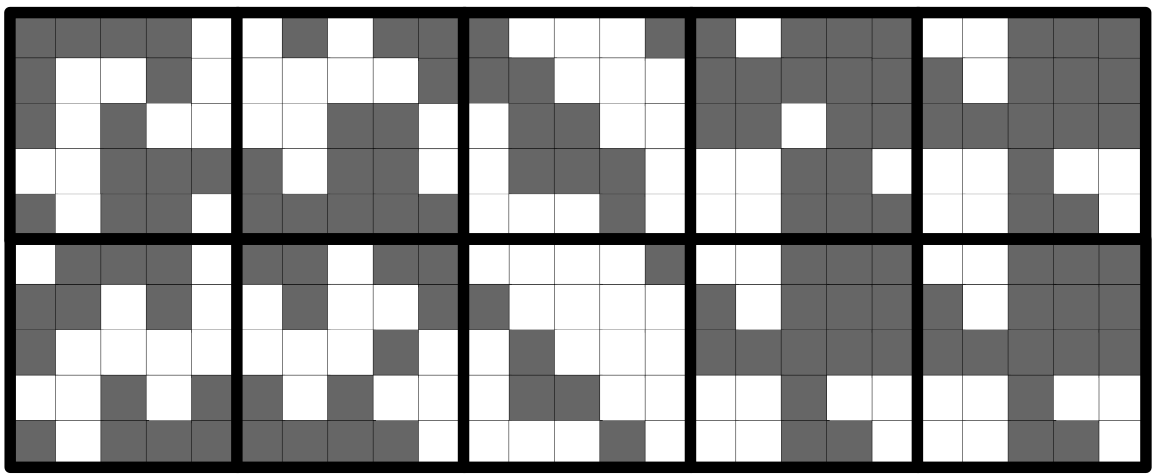


A decoding matrix for k=5, w=5

Start	XOR
-1	15
-1	13
8	13
8	4
8	12
-1	13
-1	13
-1	12
-1	7
8	9

The Algorithm

- The Final Result:
  - 45 XORs instead of 123.
  - Works with RDP too (but not EVENODD).

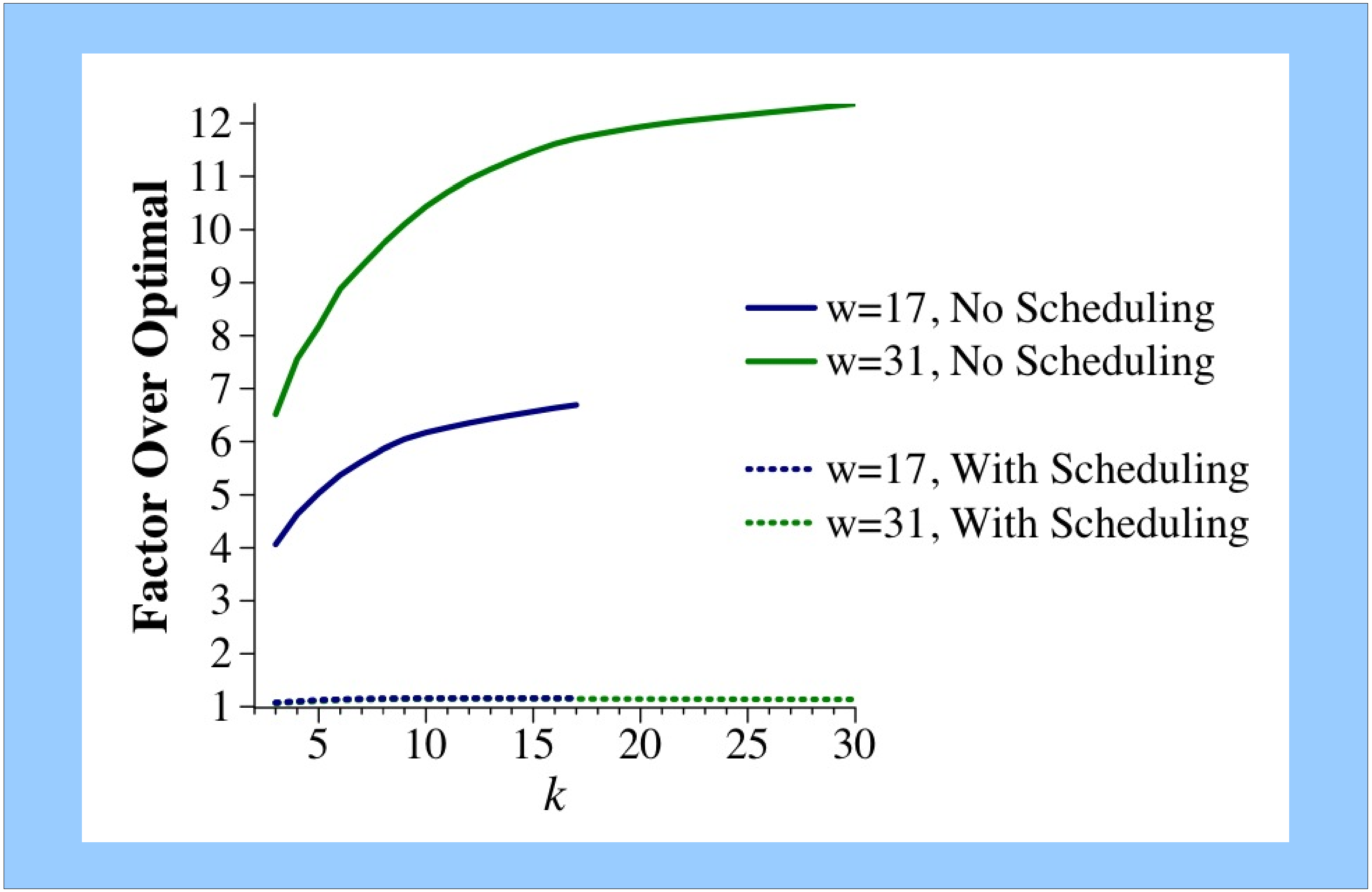


A decoding matrix for $k=5$, $w=5$

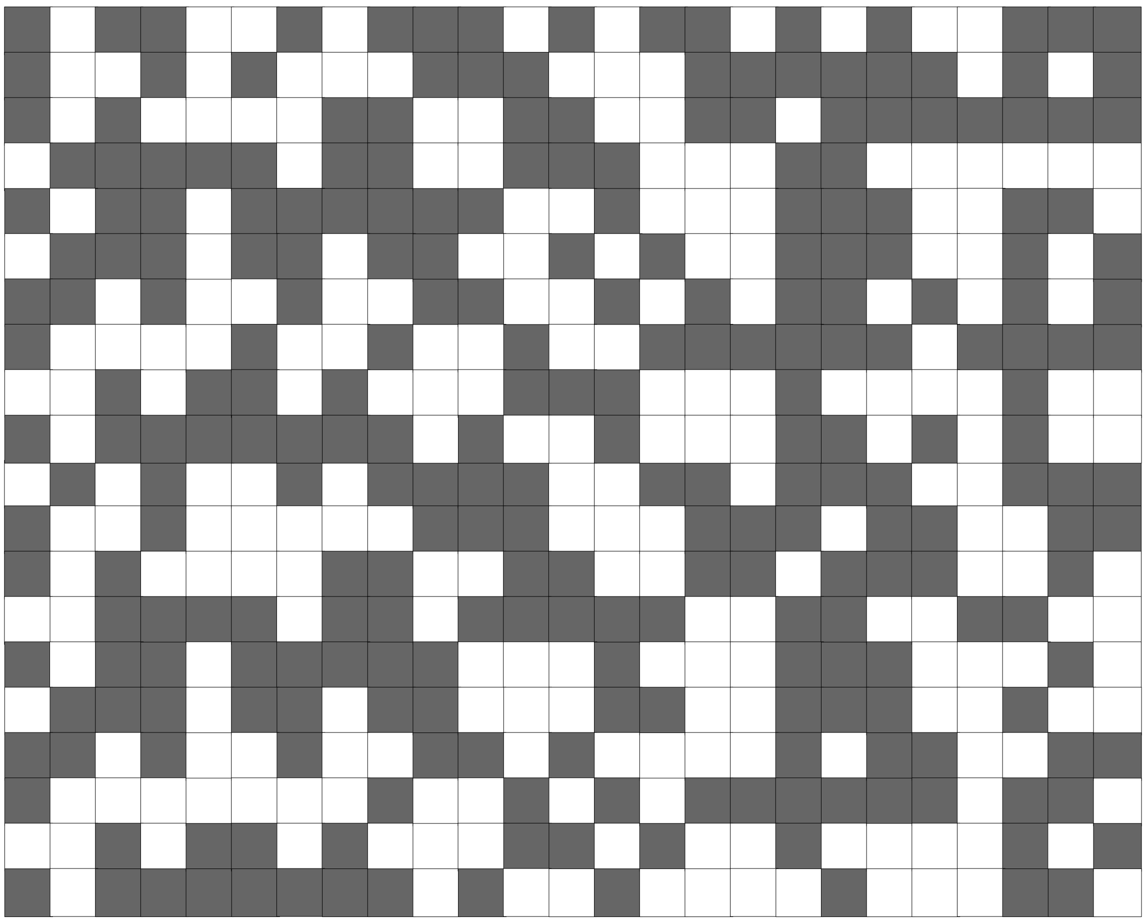
Start	XOR
5	4
6	4
7	4
8	4
9	4
4	5
0	4
1	5
-1	7
3	5

Bit-matrix Scheduling

- Put graphically:



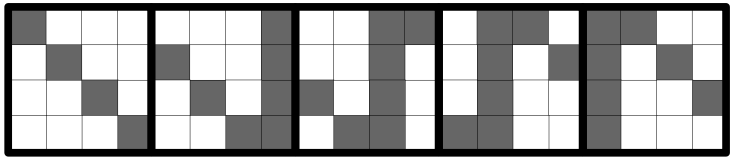
Still, $A(x) = B$ is $A(x) = B$!!!



$$x \quad = \quad B$$

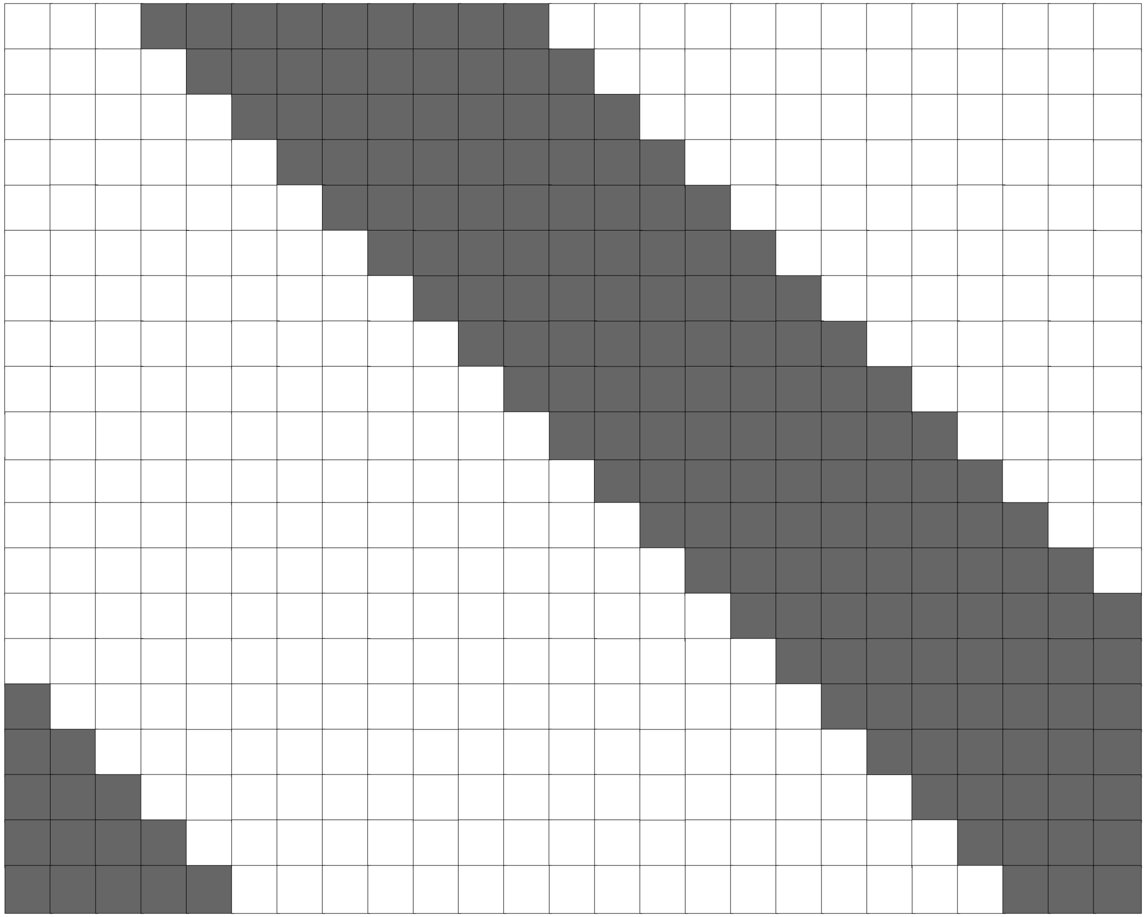
$A(x) = B$ in $GF(2)$: Current Approaches

- The algorithm you just saw:
  - “Code-Specific Hybrid Reconstruction” [Hafner04].
- Common Subexpression Removal.
  - Implemented with matching [Huang07].
  - Problem shown to be NP-Complete.
  - Works well with EVENODD & RDP.



EVENODD, k=5, w=4

Where common subexpression won't work.



$A(x) = B$ in $GF(2)$

- Dynamic Programming? Graph Algorithms?:
  - It doesn't have to be blazingly fast.
    - Hard-wire it in for given $k/m/w$.
  - Doesn't $A(x) = B$ sound like an HPC problem?

There's a whole lot more...

But what I'm hoping you've gotten out of this:

- Think coding instead of replication.
- There are good open-source tools.
- There is immediate opportunity for research in this area.

Erasure Coding: Views from 10,000 Feet and Through a Magnifying Glass

James S. Plank

Professor
Department of EECS
University of Tennessee
plank@cs.utk.edu

CCGSC
September 16, 2008