Erasure Coding Research for Reliable Distributed and Cluster Computing

James S. Plank
Professor
Department of Computer Science
University of Tennessee
plank@cs.utk.edu
CCGSC History

- In 1998, I talked about checkpointing.
- In 2000, I talked about economic models for scheduling.
- In 2002, I talked about logistical networking.
- In 2004, I was silent.
- In 2006, I’ll talk about erasure codes.
Talk Outline

• What is an erasure code & what are the main issues?
• Who cares about erasure codes?
• Overview of current state of the art
• My research
Talk Outline

• What is an erasure code & what are the main issues?

• Who cares about erasure codes?

• Overview of current state of the art

• My research
What is Erasure Coding?

Encoding:

$k$ data chunks → $m$ coding chunks

Decoding:

$k+m$ data/coding chunks, plus erasures → $k$ data chunks
Specifically

Encoding

$m$ coding chunks

$k$ data chunks

or perhaps

Decoding

$k$ data chunks
Issues with Erasure Coding

- **Performance**
  - **Encoding**
    - Typically \( O(mk) \), but not always.
  - **Update**
    - Typically \( O(m) \), but not always.
  - **Decoding**
    - Typically \( O(mk) \), but not always.
Issues with Erasure Coding

• Space Usage
  – Quantified by two of four:
    • Data Pieces: $k$
    • Coding Pieces: $m$
    • Total Pieces: $n = (k+m)$
    • Rate: $R = k/n$

  – Higher rates are more space efficient, but less fault-tolerant / flexible.
Issues with Erasure Coding

• Failure Coverage - Four ways to specify
  – Specified by a threshold:
    • (e.g. 3 erasures always tolerated).
  – Specified by an average:
    • (e.g. can recover from an average of 11.84 erasures).
  – Specified as MDS (Maximum Distance Separable):
    • MDS: Threshold = average = \( m \).
    • Space optimal.
  – Specified by Overhead Factor \( f \):
    • \( f \) = factor from MDS = \( m \)/average.
    • \( f \) is always \( \geq 1 \)
    • \( f = 1 \) is MDS.
Talk Outline

• What is an erasure code & what are the main issues?

• Who cares about erasure codes?

• Overview of current state of the art

• My research
Who cares about erasure codes?

Anyone who deals with distributed data, where failures are a reality.
Who Cares?

#1: Disk array systems.

- $k$ large, $m$ small ($< 4$)
- Minimum baseline is a requirement.
- Performance is critical.
- Implemented in controllers usually.
- RAID is the norm.
Who Cares?

#2: Peer-to-peer Systems

- $k$ huge, $m$ huge.
- Resources highly faulty, but plentiful (typically).
- Replication the norm.
Who Cares?

#3: Distributed (Logistical) Data/Object Stores

- $k$ huge, $m$ medium.
- Fluid environment.
- Speed of decoding the critical factor.
- MDS not a requirement.
Who Cares?

#4: Digital Fountains

- $k$ is big, $m$ huge
- Speed of decoding the critical factor.
- MDS is not a concern.
Who Cares?

#5: Archival Storage

- $k \leq m$?
- Data availability the only concern.
Who Cares?

#6: Clusters and Grids

Mix & match from the others.
Who cares about erasure codes?

- Fran does *(part of the “Berman pyramid”)*
- Tony does *(access to datasets and metadata)*
- Joel does *(Those sliced up mice)*
- Phil does *(Where the *!!#$’s my data?)*
- Ken does *(Scheduling on data arrival)*
- Laurent does *(Mars and motorcycles)*

They just may not know it yet.
Talk Outline

• What is an erasure code & what are the main issues?
• Who cares about erasure codes?
• Overview of current state of the art
• My research
Trivial Example: Replication

- MDS
- Extremely fast encoding/decoding/update.
- Rate: $R = 1/(m+1)$ - Very space inefficient

One piece of data: $k = 1$

Can tolerate any $m$ erasures.
Less Trivial Example: RAID Parity

- MDS
- Rate: $R = \frac{k}{k+1}$ - Very space efficient
- Optimal encoding/decoding/update:
- Downside: $m = 1$ is limited.
The Classic: Reed-Solomon Codes

- Codes are based on linear algebra over $GF(2^w)$.
- General-purpose MDS codes for all values of $k, m$.
- Slow.

\[
\begin{align*}
\begin{array}{c|c|c|c|c}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 \\
\end{array}
\end{align*}
\]

\[
B \cdot D = C
\]

$B_{ij}$, $D_i$, $C_j$
The RAID Folks: Parity-Array Codes

- Coding words calculated from parity of data words.
- MDS (or near-MDS).
- Optimal or near-optimal performance.
- Small $m$ only ($m=2$, $m=3$, some $m=4$)
- Good names: Even-Odd, X-Code, STAR, HoVer, WEAVER.
The Radicals: LDPC Codes

- Iterative, graph-based encoding and decoding
- Exceptionally fast (factor of $k$)
- Distinctly non-MDS, but asymptotically MDS

\[
\begin{align*}
D_1 + D_3 + D_4 + C_1 &= 0 \\
D_1 + D_2 + D_3 + C_2 &= 0 \\
D_2 + D_3 + D_4 + C_3 &= 0
\end{align*}
\]
Problems with each:

- Reed-Solomon coding is limited.
  - Slow.

- Parity-Array coding is limited.
  - \( m=2, m=3 \) only well understood cases.

- LDPC codes are also limited.
  - Asymptotic, probabilistic constructions.
  - Non-MDS in the finite case.
  - Too much theory; too little practice.
So......

• Besides replication and RAID, the rest is gray area, clouded by the fact that:
  
  – Research is fractured.
  
  – 60+ years of additional research is related, but doesn’t address the problem directly.
  
  – Patent issues abound.
  
  – General, optimal solutions are as yet unknown.
The Bottom Line

• The area is a mess:
  – Few people know their options.
  – Misinformation is rampant.
  – The majority of folks use vastly suboptimal techniques (especially replication).
Talk Outline

• What is an erasure code & what are the main issues?

• Who cares about erasure codes?

• Overview of current state of the art

• My research
My Mission:

• To unclutter the area using a 4-point, rhyming plan:
  
  – **Elucidate**: Distill from previous work.
  – **Innovate**: Develop new/better codes.
  – **Educate**: Because this stuff is not easy.
  – **Disseminate**: Get code into people’s hand.
5 Research Projects

• 1. Improved Cauchy Reed-Solomon coding.
• 2. Parity-Scheduling
• 3. Matrix-based decoding of LDPC’s
• 4. Vertical LDPC’s
• 5. Reverting to Galois-Field Arithmetic
1. Improved Cauchy Reed-Solomon Coding.

• Regular Reed-Solomon coding works on words of size $w$, and expensive arithmetic over $GF(2^w)$. 
1. Improved Cauchy Reed-Solomon Coding.

- Cauchy RS-Codes expand the distribution matrix over $GF(2)$ (bit arithmetic):
- Performance proportional to *number of ones per row*.
1. Improved Cauchy Reed-Solomon Coding.

- Different Cauchy matrices have different numbers of ones.
- Use this observation to derive optimal / heuristically good matrices.
1. Improved Cauchy Reed-Solomon Coding.

- E.g. Encoding performance: (NCA 2006 Paper)
2. Parity Scheduling

- Based on the following observation:

\[
\begin{align*}
A &= \sum \text{orange} \\
B &= \sum \text{red} \\
C &= \sum \text{green} + B \\
D &= \sum \text{purple} \\
E &= \sum \text{turquoise}
\end{align*}
\]

\[
\begin{align*}
C_{1,1} &= A + E + \text{black} \\
C_{1,2} &= C + \text{black} + \text{black} \\
C_{1,3} &= D + \text{black} + \text{black} + \text{black} \\
C_{2,1} &= C + E + \text{black} + \text{black} \\
C_{2,2} &= B + D + \text{black} + \text{black} \\
C_{2,3} &= A + \text{black} + \text{black} + \text{black}
\end{align*}
\]

Reduces XORs from 41 to 28 (31.7%).
Optimal = 24.
2. Parity Scheduling

- Relevant for all parity-based coding techniques:
- Start with common subexpression removal.
- Can use the fact that XOR’s cancel.

*Bottom line:* RS coding approaching optimal?
An aside for those who work with linear algebra…. Look familiar?
3. Matrix-Based Decoding for LDPC’s

• **The crux**: Graph-based encoding and decoding are *blisteringly fast*, but codes are *not MDS*, and in fact, *don’t decode perfectly*.

\[
\begin{align*}
D_1 + D_3 + D_4 + C_1 &= 0 \\
D_1 + D_2 + D_3 + C_2 &= 0 \\
D_2 + D_3 + D_4 + C_3 &= 0
\end{align*}
\]

**Add all three equations**: \( C_1 + C_2 + C_3 = D_3 \).
3. Matrix-Based Decoding for LDPC’s

- Solution: Encode with graph, decode with matrix.

**Issues**: incremental decoding, common subex’s, etc.

**Result**: Push the state of the art further.
4. Vertical LDPC’s

- Employ augmented LDPC’s & Distribution matrices to combine benefits of vertical coding/LDPC encoding.

Augmented LDPC

Augmented Binary Distribution Matrix

MDS WEAVER code for $k=2$, $m=2$
5. Reverting to Galois Field Arithmetic

- This is an MDS code for $k=4$, $m=4$ over $GF(2^w)$, $w \geq 3$:

\[
\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 2 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 2 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 2 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 2 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 2 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
2 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\
\end{array}
\]

The kitchen table code

\{ Node 1, Node 2, Node 3, Node 4, Node 5, Node 6, Node 7, Node 8 \}
5. Reverting to Galois Field Arithmetic

- If we use the Cauchy Reed-Solomon coding transformation, we get the following Binary Dist. Matrix:

<table>
<thead>
<tr>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
<th>Node 4</th>
<th>Node 5</th>
<th>Node 6</th>
<th>Node 7</th>
<th>Node 8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.33</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>XORs</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>per</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>coding</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>word.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.33 XORs per coding word.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Best current code is Cauchy RS @ 5.75 XORs per coding word.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>At $GF(2^7)$, it’s 3.14</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>And at $GF(2^\infty)$, it’s 3.00.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What I Hope You Got From This:

• You pretend to care about erasure codes.
• You understand some of their issues, and that we don’t currently live in a perfect world.
• I’m working to push the world more toward perfection.
• Some of this stuff is cool.
• Look for code / papers.
Erasure Coding Research for Reliable Distributed and Cluster Computing

James S. Plank
Professor
Department of Computer Science
University of Tennessee
plank@cs.utk.edu