When Codes for Storage Systems Meet Storage Systems

The 7th Annual Henry Taub TCE Conference, Haifa, Israel

Gala Yadgar
Computer Science
Technion

Based on joint work with Matan Liram, Eitan Yaakobi, Itzhak Tamo and Assaf Schuster
Erasure codes

- An \((n, k)\) erasure code with \(r = n - k\)
- Can recover from at most \(r\) failures
- Recover from 1 failure by reading \(k\) nodes
- Common example: Reed-Solomon
Erasure codes

• An \((n, k)\) erasure code with \(r = n - k\)
• Can recover from at most \(r\) failures
• Recover from 1 failure by reading \(k\) nodes
• Common example: Reed-Solomon
Erasure codes

\( k \) data nodes

\( r \) parity nodes

Goal: minimize recovery costs

Approach: focus on one failure
Practical codes

YOU KEEP USING THAT WORD

I DO NOT THINK IT MEANS WHAT YOU THINK IT MEANS
What makes an erasure code **practical**?

- Fast encoding/decoding

```
  x
 x XOR
 y y ⊕ x
```
What makes an erasure code practical?

What are system designers looking for?

- Fast encoding/decoding

- Reasonable storage overhead

- Flexible parameters

- Reasonable I/O behavior
Example: Butterfly codes (theory)

**Optimal**
- ✓ MDS code
- ✓ Over GF(2) → binary

En Gad, Mateescu, Blagojevic, Guyot, Bandic. *Repair-Optimal MDS Array Codes Over GF (2).* ISIT 2013
Example: Butterfly codes (theory)

**Optimal**

- ✓ MDS code
- ✓ Over GF(2) → binary
- ✓ Rebuilding ratio $= \frac{1}{r}$

$$\frac{\text{# required elements}}{\text{# surviving elements}} = \frac{3}{6} = \frac{1}{2}$$

En Gad, Mateescu, Blagojevic, Guyot, Bandic. *Repair-Optimal MDS Array Codes Over GF (2). ISIT 2013*
Example: Butterfly codes (theory)

### Rows:
\[ r^{k-1} = 8 \]

### Elements:
\[ k \times r^{k-1} = 32 \]

\( k = 7 \rightarrow 448 \text{ elements} \)

En Gad, Mateescu, Blagojevic, Guyot, Bandic. Repair-Optimal MDS Array Codes Over GF (2). ISIT 2013
Example: Butterfly codes (practice)

HDFS → \( |element| = 1MB \)

Time to recover a single node \((k = 7)\)

Reed- Solomon Butterfly

Pamies-Juarez, Blagojević, Mateescu, Cyril Gyuot, En Gad, Bandic.

Opening the Chrysalis: On the Real Repair Performance of MSR Codes. FAST 2016
Example: Butterfly codes (practice)

HDFS $\rightarrow |element| = 1MB$

Ceph $\rightarrow |element| \approx 9KB$

Time to recover a single node ($k = 7$)

Pamies-Juarez, Blagojević, Mateescu, Cyril Gyuot, En Gad, Bandic.

Opening the Chrysalis: On the Real Repair Performance of MSR Codes. FAST 2016
Zigzag codes

Optimal
✓ Non-binary
✓ MDS code
✓ Rebuilding ratio $= \frac{1}{r}$

Zigzag codes

Optimal
✓ Non-binary
✓ MDS code
✓ Rebuilding ratio $= \frac{1}{r}$
✓ Arbitrary $r$

Code “duplication”

\[ s \times k = 2 \times k' \]

Code “duplication”

\[ s \times k \]

\[ k = 2 \times k' \]

What is the optimal construction for $k = 6$?

3 × ▼

2 × ▼
What is the optimal construction for $k = 6$?

- Rebuilding ratio $= \frac{9}{14} = 0.64$

- Rebuilding ratio $= \frac{16}{28} = 0.57$
Make I/O more sequential: constructions

• “Virtual nodes” (puncturing)
• Optimal constructions and dependency sets
• Alignment and padding
Make I/O more sequential: constructions

• “Virtual nodes” (puncturing)

• Optimal constructions and dependency sets

• Alignment and padding
Make I/O more sequential: request coalescing

How many I/O requests read 5 sectors?

Naïve approach (5)  Conservative approach (2)  Aggressive approach (1*)
Evaluation

• 10 servers
  • 16 cores
  • 64-128GB RAM
  • 2 X 500GB HDDs

• 10Gib Ethernet switch

• 19 nodes X 10GB data + redundancy
  • $k = 6, 8, 10$
  • $r = 2, 3, 4$
  • $s = 1, 2, 3, 4, 5$
Recovery reads (normalized to Reed-Solomon)

Zigzag reads up to 34% less data
Within 32% of theoretical bound

Zigzag reads 18%-40% less data
Within 16% of theoretical bound
Recovery time (normalized to Reed-Solomon)

Optimization tradeoff: reads vs. time

Zigzag recovers up to 28% faster (but sometimes slower)
Request coalescing

Stripe size = 4MB

Normalized reads
\( r = 4 \)

Normalized recovery time
\( r = 4 \)
Recovery reads (normalized to Reed-Solomon)

LRC
✓ No fragmentation
✓ Reads less when \( r \) is small
✓ Fast recovery
✗ Extra storage

Zigzag
✗ Fragmentation
✓ MDS - lower overhead
✓ Reads less when \( r \) is large
✓ Fast recovery with large objects

Stripe size = 64MB

\[ k = 6 \]
\[ k = 8 \]
\[ k = 10 \]
Takeaways

✓ Zigzag works for real systems
  • Implementation is only the first step
  • Optimal is not always the best
  • Practical is not always the same

→ Code designers: know your systems!
→ System designers: know your codes!