Pyrit: Polynomial Ring Transforms for Fast Erasure Coding

Some parts of this work have been patented

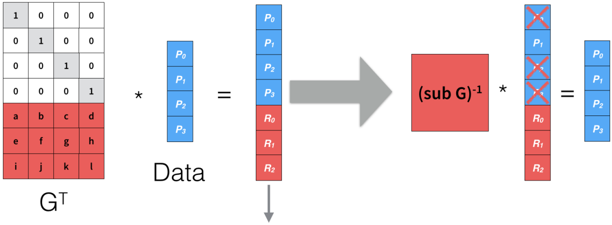
Jonathan Detchart and Jérôme Lacan

July 18, 2017

Outline

1. **Use case and context**
2. Ring transforms
3. An example
4. Results
5. Conclusion

Erasure codes for network coding: a remainder [1]



$R_0 = a * P_0 + b * P_1 + c * P_2 + d * P_3$

$R_1 = e * P_0 + f * P_1 + g * P_2 + h * P_3$

`*` and `+` are done in a **finite field**

Coding operations

erasure codes are matrix vector product over a finite field

[1] J. S. Plank, K. M. Greenan, and E. L. Miller. **A Complete Treatment of Software Implementations of Finite Field Arithmetic for Erasure Coding Applications**. Tech. rep. UT-CS-13-717. University of Tennessee, 2013.

The context

As operations in a finite field are complex, we do operations into a specific **ring**

- By using fast transforms, we move from a finite field structure into a simpler structure called **ring**
- In the ring, operations are much easier.
- We need to go back from the ring structure to the field using the reverse transforms

Outline

Objective : fast encoding/decoding operations

$\mathbb{F}_{2^w}$: $(\alpha_0, \ldots, \alpha_{k-1}) \times \begin{pmatrix} \gamma_{0,0} & \cdots & \gamma_{0,n-1} \\ \vdots & \ddots & \vdots \\ \gamma_{k-1,0} & \cdots & \gamma_{k-1,n-1} \end{pmatrix}$

$R_{2,w+1}$:

Objective : fast encoding/decoding operations

$$\mathbb{F}_{2^w}: \quad (\alpha_0, \ldots, \alpha_{k-1}) \times \begin{pmatrix} \gamma_{0,0} & \cdots & \gamma_{0,n-1} \\ \vdots & \ddots & \vdots \\ \gamma_{k-1,0} & \cdots & \gamma_{k-1,n-1} \end{pmatrix}$$

$$\Downarrow$$

$$R_{2,w+1}: \quad (a_0, \ldots, a_{k-1})$$

Objective : fast encoding/decoding operations

$$\mathbb{F}_{2^w}: \quad (\alpha_0, \ldots, \alpha_{k-1}) \times \begin{pmatrix} \gamma_{0,0} & \cdots & \gamma_{0,n-1} \\ \vdots & \ddots & \vdots \\ \gamma_{k-1,0} & \cdots & \gamma_{k-1,n-1} \end{pmatrix}$$

$$\Downarrow \qquad\qquad \Downarrow$$

$$R_{2,w+1}: \quad (a_0, \ldots, a_{k-1}) \times \begin{pmatrix} g_{0,0} & \cdots & g_{0,n-1} \\ \vdots & \ddots & \vdots \\ g_{k-1,0} & \cdots & g_{k-1,n-1} \end{pmatrix}$$

Objective : fast encoding/decoding operations

$$\mathbb{F}_{2^w}: \quad (\alpha_0, \ldots, \alpha_{k-1}) \times \begin{pmatrix} \gamma_{0,0} & \cdots & \gamma_{0,n-1} \\ \vdots & \ddots & \vdots \\ \gamma_{k-1,0} & \cdots & \gamma_{k-1,n-1} \end{pmatrix}$$

$$\Downarrow \qquad\qquad \Downarrow$$

$$R_{2,w+1}: \quad (a_0, \ldots, a_{k-1}) \times \begin{pmatrix} g_{0,0} & \cdots & g_{0,n-1} \\ \vdots & \ddots & \vdots \\ g_{k-1,0} & \cdots & g_{k-1,n-1} \end{pmatrix} = (b_0, \ldots, b_{n-1})$$

Objective : fast encoding/decoding operations

$$\mathbb{F}_{2^w}: \quad (\alpha_0, \ldots, \alpha_{k-1}) \times \begin{pmatrix} \gamma_{0,0} & \cdots & \gamma_{0,n-1} \\ \vdots & \ddots & \vdots \\ \gamma_{k-1,0} & \cdots & \gamma_{k-1,n-1} \end{pmatrix} = (\beta_0, \ldots, \beta_{n-1})$$

$$\Downarrow \qquad\qquad \Downarrow \qquad\qquad \Uparrow$$

$$R_{2,w+1}: \quad (a_0, \ldots, a_{k-1}) \times \begin{pmatrix} g_{0,0} & \cdots & g_{0,n-1} \\ \vdots & \ddots & \vdots \\ g_{k-1,0} & \cdots & g_{k-1,n-1} \end{pmatrix} = (b_0, \ldots, b_{n-1})$$

Transforms between the field and the ring

To optimize the coding operations, we consider several transforms:

- The *embedding transform* [2]: **very fast** to transform **finite field** elements to **ring elements**
- The *parity transform*: **very fast** to transform **finite field** elements to **ring elements**
- The *sparse transform*: **very efficient** to **reduce the complexity** of the operations in the ring. Will choose the sparsest element in the ring corresponding to the field element.

[2] Toshiya Itoh and Shigeo Tsujii. "Structure of parallel multipliers for a class of fields $GF(2^m)$". In: **Information and Computation** 83.1 (1989), pp. 21 –40.

Final scheme [3]

$$\mathbb{F}_{2^w}: \quad (\alpha_0, \ldots, \alpha_{k-1}) \times \begin{pmatrix} \gamma_{0,0} & \cdots & \gamma_{0,n-1} \\ \vdots & \ddots & \vdots \\ \gamma_{k-1,0} & \cdots & \gamma_{k-1,n-1} \end{pmatrix} = (\beta_0, \ldots, \beta_{n-1})$$

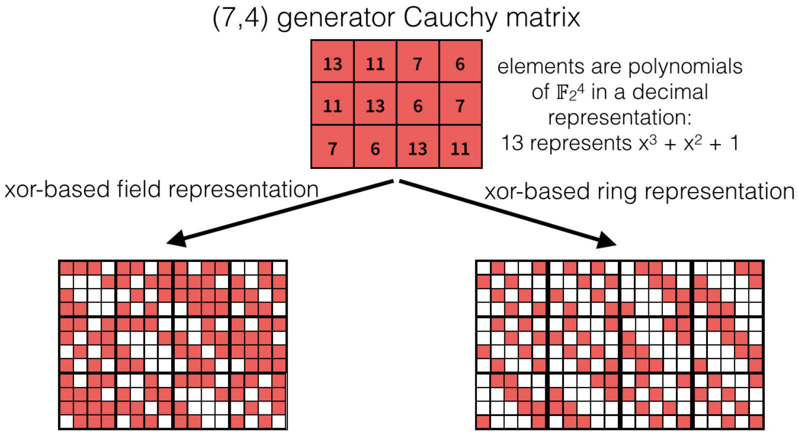
$\Downarrow$ *Emb* or *Par* $\qquad\qquad$ $\Downarrow$ *Sparse* $\qquad\qquad$ $\Uparrow$ Emb^{-1} or Par^{-1}

$$R_{2,w+1}: \quad (a_0, \ldots, a_{k-1}) \times \begin{pmatrix} g_{0,0} & \cdots & g_{0,n-1} \\ \vdots & \ddots & \vdots \\ g_{k-1,0} & \cdots & g_{k-1,n-1} \end{pmatrix} = (b_0, \ldots, b_{n-1})$$

[3] J. Detchart and J. Lacan. “Fast Xor-based Erasure Coding based on Polynomial Ring Transforms”. In: **ISIT17**.

Outline

The sparse transform

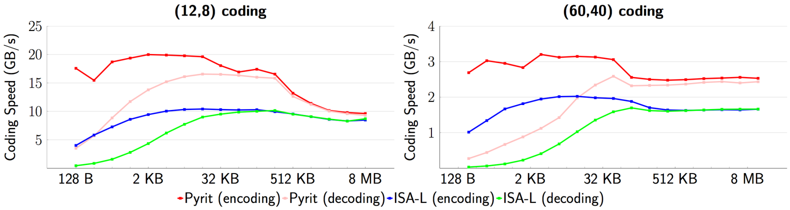


Xor operations in the field or in the ring

Using the *sparse* transform

Outline

Comparison with the fastest known implementation: Intel ISA-L [4]



Coding Speeds (in GB/s) for several data block lengths

CPU: Intel Core i5-6500 (Skylake architecture) @3.20 GHz

- Encoding and decoding: up to 2X faster
- Works well with small codes (GF(16), GF(64))

[4] "ISA-L: Intel Storage Acceleration library". In: **https://github.com/01org/isa-l**.

Outline

Conclusion

- Reduces the coding and decoding complexities
- Easy to implement (just a set of *xor* operations)
- Allows the use of uncommon fields like GF(64) rather than GF(16) or GF(256)
- GF(64) is a good compromise between capacity corrections and coding speed (fits perfectly in Tetrys)

Thank you!