Storm: Rateless MDS Erasure Codes

Pedro Moreira da Silva(✉), Jaime Dias, and Manuel Ricardo

INESC TEC, Faculdade de Engenharia, Universidade do Porto,
Rua Dr. Roberto Frias, 378, 4200-465 Porto, Portugal
{pmms,jdias,mricardo}@inesctec.pt

Abstract. Erasure codes have been employed in a wide range of applications to increase content availability, improve channel reliability, or to reduce downloading time. For several applications, such as P2P file sharing, MDS erasure codes are more suitable as the network is typically the most constrained resource, not the CPU. Rateless MDS erasure codes also enable to adjust encoding and decoding algorithms as function of dynamic variables to maximize erasure coding gains. State-of-the-art MDS erasure codes are either fixed-rate or have practical limitations. We propose Storm erasure codes, a rateless MDS construction of Reed-Solomon codes over the finite field $\mathbb{F}_{p^2}$, where p is a Mersenne prime. To the best of our knowledge, we are the first to propose a rateless construction (n can be increased in steps of k) with $\Theta(n \log k)$ encoding time complexity and $\min\left\{\Theta(n \log n), \Theta\left(k \log^2 k\right)\right\}$ upper bound for decoding time complexity. We provide the complexity analysis of encoding and decoding algorithms and evaluate Storm's performance.

1 Introduction

Erasure codes have been employed in a wide range of applications to increase content availability, improve channel reliability, or to reduce downloading time. An erasure code generates a set of n symbols, from a set of k symbols at a rate given by k/n, so that any subset of $k(1+\epsilon(k))$ is enough to reconstruct the original information, where $\epsilon(k)$ is the erasure coding overhead. Erasure codes are usually classified according to three orthogonal properties: (1) systematicity, (2) rate fixedness, and (3) coding overhead. An erasure code is systematic if the input symbols are embed into output symbols, and non-systematic otherwise. If n is static and need to be known before encoding, the erasure code is fixed-rate. If n can be dynamically increased and the amount of symbols that can be generated does not impose any practical limitation, the erasure code is rateless. Finally, an erasure code is said MDS (Maximum Distance Separable) if any k symbols out of n are enough to reconstruct the original information [$\epsilon(k) = 0$], or non-MDS if additional symbols are required [$\epsilon(k) > 0$]. Non-MDS erasure codes introduce coding overhead for reducing significantly the encoding and decoding time complexities. LT codes [1] and Raptor codes [2] are the most prominent examples of non-MDS erasure codes because they are rateless and asymptotically optimal [$\epsilon(k) \to 0$ as $k \to \infty$], and the latter is able to achieve linear coding and decoding time complexities.

S. Mumtaz et al. (Eds.): WICON 2014, LNICST 146, pp. 153–158, 2015.
DOI: 10.1007/978-3-319-18802-7_22

For several applications, such as P2P file sharing, MDS erasure codes are more suitable as the network is typically the most constrained resource, not the CPU [3]. Rateless MDS erasure codes also enable to set n as a function of dynamic variables, such as peer participation dynamics and content popularity, to maximize erasure coding gains. Classic Reed-Solomon (RS) codes [4], the most well-known class of MDS codes, are systematic, fixed-rate and have $\Theta(nk)$ encoding, and $\Theta(k^2)$ decoding time complexities, which limits their practical application to 255 symbols. ROME [5] is a rateless MDS construction but one with equivalent time complexity and practical limitations. To overcome these limitations, Didier [6] proposed encoding and decoding algorithms for RS codes over the binary finite field $\mathbb{F}_{2^m}$ with, respectively, $\Theta(n \log n)$ and $\Theta(n \log^2 n)$ time complexities, where $n = 2^m$ and is fixed to the size of the binary finite field. Soro [7] presented encoding and decoding algorithms with $\Theta(n \log n)$ time complexity over a finite field $\mathbb{F}_p$, where p is a Fermat prime $(p = 2^{2^m} + 1)$. Lin [8] extended the work of Didier [6] and proposed $\Theta(n \log k)$ encoding and $\Theta(n \log n)$ decoding algorithms over $\mathbb{F}_{2^m}$ with n also fixed to the finite field size.

Despite their merits, [6–8] still create fixed-rate codes because all encoding symbols must be generated at once for achieving such encoding time complexities. Also, their practical use is limited to about 2^{16} symbols: $2^{16} + 1$ is the largest Fermat number prime up to $2^{2048} + 1$; multiplications over binary finite fields are performed using a lookup table, as carry-less multiplication is not as efficient on current CPUs, and large lookup tables severely degrade performance.

We propose a rateless MDS construction of Reed-Solomon codes over the finite field $\mathbb{F}_{p^2}$, where p is a Mersenne prime $(p = 2^m - 1)$, which we name Storm. Although the construction of RS codes over such field has already been proposed [9], to the best of our knowledge, we are the first to propose a rateless construction (n can be increased in steps of k) with $\Theta(n \log k)$ encoding time complexity and $\min\{\Theta(n \log n), \Theta(k \log^2 k)\}$ upper bound for decoding time complexity. We provide the complexity analysis of encoding and decoding algorithms and evaluate Storm's performance.

The remaining of this paper is structured as follows. Storm erasure codes are presented in Sect. 2. The performance assessment is conducted on Sect. 3. Section 4 concludes this paper and presents the future work.

2 Storm Erasure Codes

Let $s = (s_0, s_1, \ldots, s_{k-1})$ be a source vector of size k, $s(x) = \sum_{i=0}^{k-1} s_i \cdot x^i$ its associated polynomial, and $e = (e_0, e_1, \ldots, e_{n-1})$ an encode vector of size n. The transformation $(s_0, \ldots, s_{k-1}) \xrightarrow{\mathcal{F}} (e_0, \ldots, e_{n-1})$ over $\mathbb{F}_p^n$, with $e_j = \sum_{i=0}^{k-1} s_i \cdot x_j^i$, can be performed as a multipoint polynomial evaluation at the points (code locators) x_j, i.e., $e_j = s(x_j)$. The inverse transformation, $\mathcal{F}^{-1}$, given that a polynomial of degree $<k$ is uniquely determined by any k unique pairs (x_i, e_i), can be performed as a polynomial interpolation. Let the Lagrange basis polynomial be $L(x) = \prod_{i=0}^{k-1} x - x_i$, the barycentric weights be $w_i = \left(\prod_{j=0, j\neq i}^{k-1} x_i - x_j\right)^{-1}$,

and $s(x)$ is defined by

$$s(x) = \sum_{i=0}^{k-1} e_i \cdot \prod_{j=0,\ j\neq i}^{k-1} \frac{x - x_j}{x_i - x_j} = L(x) \cdot \sum_{i=0}^{k-1} \frac{e_i \cdot w_i}{x - x_i} \ . \tag{1}$$

Let $M(k)$ represent the time complexity of multiplying two polynomials of degree $<k$ over a finite field $\mathbb{F}_p$. The encoding and decoding algorithms at arbitrary points takes $M(k) \log k$ time. We refer the reader to [10] for a description of the multipoint evaluation and interpolation algorithms at arbitrary points.

Let r be an n^{th} root of unity of a non-binary finite field $\mathbb{F}_p$, i.e., $n \mid p-1$ so that $r^n \equiv 1 \mod p$ and $r^i \not\equiv 1 \mod p,\ 0 < i < n$. Let r^{z_j} be the power representation of x_j, it follows that $e_j = s(r^{z_j}) = \sum_{i=0}^{k-1} s_i \cdot r^{i \cdot z_j}$. As so, the fast Fourier transform (FFT) is an efficient method for evaluating a polynomial of degree $<n$ at all of the n roots of unity in $\Theta(n \log n)$ time. The FFT can also be used to perform efficient multiplication of polynomials in $\Theta(k \log k)$ time $[M(k) = \Theta(k \log k)]$: multiplying two polynomials of degree $<k$ takes two FFTs of size $2k$ and one inverse FFT (IFFT) of size $2k$.

2.1 Finite Field

The finite field $\mathbb{F}_{p^2}$, where p is a Mersenne prime ($p = 2^m - 1$), can be constructed as $\mathbb{F}_{p^2} = \{a + b\hat{\imath} \mid a, b \in \mathbb{F}_p\}$, where $\hat{\imath} = \sqrt{-1}$, given that every irreducible quadratic polynomial over $\mathbb{F}_p$ must split over $\mathbb{F}_{p^2}$ [9]. Moreover, in $\mathbb{F}_{p^2}$ there is always a multiplicative group of size 2^{m+1}, as $2^{m+1} \mid p^2 - 1$, whose root, r, is $2^{2^{m-2}} + (-3)^{2^{m-2}}$ [11], and the components of the 8^{th} unity roots are fixed powers of two, only involving additions and circular shifts, enabling efficient radix-8 FFTs. Let $c = 2^{(m-1)/2}$, the set of 8^{th} roots of unity is $\{1, -1, \hat{\imath}, -\hat{\imath}, c(1+\hat{\imath}), c(1-\hat{\imath}), c(-1+\hat{\imath}), c(-1-\hat{\imath})\}$. For improved performance, when performing FFT and IFFT, the unity roots must be pre-calculated. Given that in $\mathbb{F}_{p^2}$ the inverse of a unity root z is its complex conjugate $\left(z \cdot z^{-1} = z \cdot \overline{z} = 1\right)$, the set of unity roots can be shared by the FFT and the IFFT. Finally, there is no known file size limit for $\mathbb{F}_{p^2}$, being $2^{57885161} - 1$ the largest known Mersenne prime.

2.2 Mapping

Elements of $\mathbb{F}_{p^2}$ are pairs of $\mathbb{F}_p$ elements. Therefore, we map each m bits of source data into an $\mathbb{F}_p$ element. Yet, 0 has to be distinguished from $2^m - 1$ as $2^m - 1 \equiv 0 \mod p$. A transformation on $\mathbb{F}_{p^2}^{p/2-1}$ has, at most, $p-2$ elements of $\mathbb{F}_p$ $[2 \cdot (p/2 - 1)]$. Thus, for the set of source elements, s, there is at least one element of $\mathbb{F}_p$, a, that is not in the set: $\forall s \in \mathbb{F}_{p^2}^{p/2-1},\ \exists a \in \mathbb{F}_p : a \notin s$. This element can be used to replace $2^m - 1$ in the source data whenever it occurs, before encoding, and do the reverse after decoding. A transformation on $\mathbb{F}_{p^2}^n$, where $n \geq p/2$, can be treated the same way by dividing it in several $\mathbb{F}_{p^2}^{p/2-1}$ transformations.

2.3 Encoding

Let $s = (s_0, s_1, \ldots, s_{k-1})$ be a source vector of size k, and $e = (e_0, e_1, \ldots, e_{n-1})$ an encoded vector of size n. Extending s with $n - k$ zeros to make it of size n, $s = (s_0, s_1, \ldots, s_{k-1}, 0, \ldots, 0)$, enables n symbols to be generated, at once, using a size n FFT (FFT_n). However, this approach does not enable e to increase as needed, at least not efficiently. To make Storm rateless, we developed an encoding algorithm that enables e to increase in steps of k elements. Let r_n be the n^{th} root of unity in $\mathbb{F}_{p^2}$, i.e., $r_n = r^{\frac{2^{m+1}}{n}}$, $\forall n : n \mid 2^{m+1}$. Considering that,

$$e_{j'=g+({}^n/k)j} = \sum_{i=0}^{k-1} s_i \cdot r_n^{i(g+({}^n/k)j)} = \sum_{i=0}^{k-1} \left(s_i \cdot r_n^{ig}\right) \cdot r_k^{ij} \tag{2}$$

k innovative symbols can be generated using an FFT_k by applying r_n^{ig} factors to each s_i, $0 \leq i < k$, where $0 \leq g < {}^n/k$. The generation of n symbols can be performed in ${}^n/k$ independent steps, and has $\Theta(n \log k)$ time complexity. Let $R_n = \left\{r_n^0, \ldots, r_n^{n-1}\right\}$ be the set of n^{th} roots of unity. Given that $R_{n/2} \subset R_n$, increasing n has no impact on the previously encoded symbols. A transmission data unit of d symbols, such as an IP packet or a P2P chunk, is composed by the evaluation of d source vectors of size k at a given code locator x_i.

2.4 Decoding

The decoding algorithm consists in five main steps: (1) calculate $L(x)$; (2) compute $L'(x)$; (3) evaluate the barycentric weights as $w_i = L'(x_i)$; (4) compute all $y_i = e_i \cdot w_i$; (5) perform the interpolation. Given that any set of k points is a subset of a set of n roots of unity, the interpolation can be performed either at k arbitrary points or at n unity roots. Let $Y(x) = \sum_{i=0}^{k-1} y_i \cdot x^{z_i}$, and using the Taylor series of ${}^1/(x - r^{z_i}) = -\sum_j r^{z_i(-j-1)} \cdot x^j$, Lagrange's interpolation formula becomes [7]

$$s(x) = -L(x) \cdot \sum_{i=0}^{k-1} \left(\sum_{j=0}^{n-1} y_i \cdot (r^{z_i})^{-j-1} \cdot x^j \right) = -L(x) \cdot \sum_{j=0}^{n-1} Y(r^{-j-1}) \cdot x^j \ . \tag{3}$$

Considering $FFT_{2k} \approx 2FFT_k$, step 1) takes $M(k) \log k$ time: $\log k$ stages each taking $3FFT_{2k} \approx 6FFT_k$. Step 2) takes $\Theta(k)$ time. Step 3) takes $M(n)$ time at n roots of unity – $1FFT_n$ – or $M(k) \log k$ time at arbitrary points – $\log k$ stages of $6FFT_k$. Step 4 has also linear time complexity. Step 5), using Eq. 3, takes $M(n)$ time at n unity roots because evaluating $Y(x)$, evaluating $\sum_{j=0}^{n-1} Y(r^{-j-1}) \cdot x^j$, and multiplying the result by $L(x)$ are all performed in $M(n)$ time: $1FFT_n + 3FFT_{2n} \approx 7FFT_n$. At arbitrary points, step 5) takes $M(k) \log k$ time: $\log k$ stages of $6FFT_k$. Therefore, step 5) has $\min\{M(n), M(k) \log k\}$ time complexity. The overall time complexity is $M(k) \log k + \min\{M(n), M(k) \log k\}$. However, in practice, the overall time complexity is just $\min\{M(n), M(k) \log k\}$ because steps 1–3, since they only depend on x_i, are only performed once, while

steps 4 and 5 are performed several hundreds or thousands of times for an IP packet or a P2P file sharing chunk. $L(x)$ also depends only on x_i and only needs to be computed once per packet or chunk, thus step 5) can be performed in $5FFT_n$ at n roots of unity. Therefore, the decoding has $\min\{M(n), M(k)\log k\}$ practical time complexity: $\min\{5FFT_n, \log k \cdot 6FFT_k\}$.

3 Results

Complexity analysis is important to understand how an algorithm behaves as the input grows; still, it hides constant factors that may alter significantly the algorithms real performance. To assess Storm erasure codes performance, and to compare them with Soro's [7] – the only ones with $\Theta(n \log n)$ time complexity that admit any power of two for n and k, $k \leq n$ –, we implemented them in C++, and ran them on an Intel Core i5-560M under Ubuntu 13.10 64 bits. For evaluation, the Fermat field is $\mathbb{F}_{2^{16}+1}$ and the Mersenne extension field is $\mathbb{F}_{(2^{31}-1)^2}$. The results shown are for a single thread.

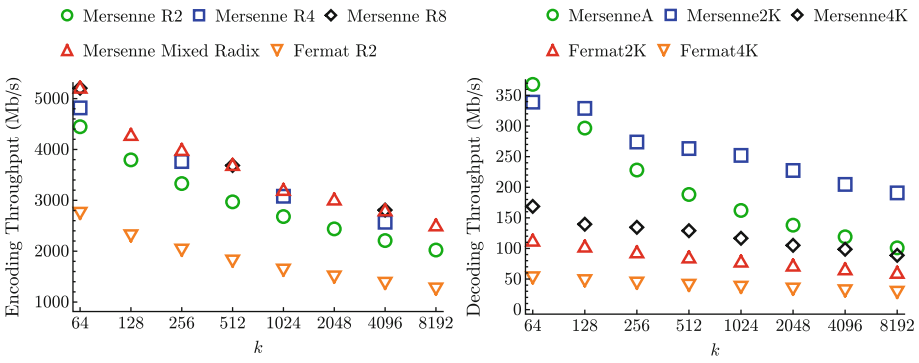

Fig. 1. Encoding throughput for radix-2, radix-4, radix-8, and mixed radix over Mersenne extension field, and for radix-2 over Fermat field [left]. Decoding throughput using interpolation at arbitrary points (A suffix), and at roots of unity with $n = 2k$ and $n = 4k$ over Mersenne and Fermat fields [right].

It can be seen in Fig. 1 the performance improvement provided by radix-8 FFT in comparison to radix-2 FFT. The mixed radix FFT over $\mathbb{F}_{p^2}$, which uses higher radices whenever possible, nearly doubles the throughput provided by radix-2 FFT over Fermat fields. When comparing only radix-2 FFTs, the larger symbols of $\mathbb{F}_{p^2}$ (62 vs 16 bits) improve performance despite multiplications being slightly more expensive (four integer multiplications and two additions). Identical results were obtained for decoding: the throughput for $n = 2k$ over Mersenne extension field, which is about twice the throughput for $n = 4k$ over that field, is slightly greater than twice the throughput for $n = 2k$ over $\mathbb{F}_{2^{16}+1}$. The decoding algorithm at arbitrary points is more advantageous for small values of k and, for k up to 8192 when $^n/_k > 2$.

4 Conclusions

We presented Storm erasure codes, rateless MDS erasure codes based on RS codes with $\Theta(n \log k)$ encoding time complexity and $\min\{M(n), M(k) \log k\}$ upper bound for decoding time complexity, and assessed their practical performance. These codes are able to saturate a Gigabit interface on a four years old CPU, and are able to provide nearly twice the throughput of equivalent codes defined over Fermat fields. Unlike Fermat fields, there is no known field size limit for $\mathbb{F}_{p^2}$. For evaluation, we only considered a single thread, so we intend to create a parallel multi-core CPU and GPU implementation.

Acknowledgments. This work was supported by Fundação para a Ciência e Tecnologia (FCT) under grant SFRH/BD/69388/2010.

References

1. Luby, M.: LT codes. In: 2002 Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, pp. 271–280 (2002)
2. Shokrollahi, A.: Raptor codes. IEEE Trans. Inf. Theory **52**(6), 2551–2567 (2006)
3. Liben-Nowell, D., Balakrishnan, H., Karger, D.: Analysis of the evolution of peer-to-peer systems. In: Proceedings of the Twenty-First Annual Symposium on Principles of Distributed Computing, PODC 2002, pp. 233–242. ACM (2002)
4. Reed, I., Solomon, G.: Polynomial codes over certain finite fields. J. SIAM **8**(2), 300–304 (1960)
5. He, N., Xu, Y., Cao, J., Li, Z., Chen, H., Ren, Y.: ROME: rateless online MDS code for wireless data broadcasting. In: Global Telecommunications Conference (GLOBECOM 2010), pp. 1–5. IEEE, December 2010
6. Didier, F.: Efficient erasure decoding of Reed-Solomon codes. CoRR, abs/0901.1886 (2009)
7. Soro, A., Lacan, J.: FNT-based Reed-Solomon erasure codes. In: Proceedings of the 7th IEEE Conference on Consumer Communications and Networking Conference, CCNC 2010, Piscataway, NJ, USA, pp. 466–470. IEEE Press (2010)
8. Lin, S.-J., Chung, W.-H., Han, Y.S.: Fast encoding/decoding algorithms for Reed-Solomon erasure codes. CoRR, abs/1404.3458 (2014)
9. Reed, I., Truong, T., Welch, L.: The fast decoding of Reed-Solomon codes using number theoretic transforms. To The Deep Space Network Progress Report, pp. 42–35 (1976)
10. Crandall, R.E., Pomerance, C.: Prime Numbers: A Computational Perspective, 2nd edn. Springer, New York (2005)
11. Creutzburg, R., Tasche, M.: Parameter determination for complex number-theoretic transforms using cyclotomic polynomials. Math. Comput. **52**(185), 189–200 (1989)