THE EFFICIENT IMPLEMENTATION OF REED-SOLOMON HIGH RATE ERASURE RESILIENT CODES

Jin Li

Microsoft Research, Communication and Collaboration
One Microsoft Way, Bld. 113, Redmond, WA, 98052.
Email: jinl@microsoft.com

ABSTRACT

In this paper, we investigate the efficient implementation of Reed-Solomon high rate erasure resilient codes. Though the implementation of Reed-Solomon codes for error correction coding has been extensively investigated in the past, there is little work on the efficient implementation of Reed-Solomon codes for high rate erasure resilient coding application. In this paper, we investigate a number of technologies, including the direct inverse of the Reed-Solomon sub-generator matrix, and the scalar vector multiplication and addition on the Galois Field to speed up the Reed-Solomon erasure encoding/decoding operations. Our implementation of the Reed-Solomon erasure code achieves an encoding/decoding throughput of 25MB per second, (i.e., 200Mbps), on a Pentium 2.8Ghz computer.

1. INTRODUCTION

A high rate erasure resilient code is a block error correction code with a large coded message space. It is useful in a number of content distribution/backup applications. One application is the digital fountain[1], where a server multicasts/broadcasts erasure coded messages non-stop to a multiple of clients. Each client may tune in from time to time, and pick up the coded messages that are on-air at the moment. Another application is the distributed backup, where a file is erasure encoded and stored in a large number of storage units, either locally or in a distributed fashion. During the recovery, the client attempts to restore the file from the accessible storage units. The third sample application is the distributed content hosting and/or streaming[7], where a file/media is distributed to a number of hosting servers, each of which may elect to host a portion of the file/media in the erasure coded form. During the retrieval, the client locates the hosting servers that are willing to serve, and retrieves the erasure coded file/media simultaneously from those servers. In all such applications, a file and/or media is encoded into a large number of distinctive coded messages. During the retrieval, the client attempts to use a minimum number of the coded messages that are equal to or slightly larger than the original to recover the file/media. The client usually does not have control over what coded messages are available. As a result, the process of distributed content broadcast/backup/hosting/streaming can be considered as passing the coded messages through an erasure channel with heavy loss, and recovering the messages afterwards.

A number of error correction codes can serve as the high rate erasure resilient codes. These codes include: the random generation of linear codes (RLC)[2], the low density parity check (LDPC) codes[3], turbo codes[4], LT codes[5], and Reed-Solomon codes[6]. Among the error correction codes, the Reed-Solomon codes stand out with a number of unique properties. Reed-Solomon codes are maximum distance separable (MDS) codes. They achieve the maximum channel coding efficiency, and are able to decode the original messages with the exact number of received coded messages. Because the generator matrix of the Reed-Solomon codes are structured, the Reed-Solomon coded message can be identified by the row index, which reduces the overhead needed to identify the coded message. Reed-Solomon codes can be applied to messages with small access granularity (short message and small number of original messages), and are suitable for on demand distributed content hosting/streaming applications[7].

Though the implementation of the Reed-Solomon error correction codes has been extensively investigated, there are relatively few works on the efficient implementation of the Reed-Solomon codes for high rate error resilient coding application, which bears unique characteristics. For example, in high rate erasure resilient coding scenario, the coded messages are generated on demand, just prior to message distribution. This is different from the error correction coding application, where all coded messages are generated at the same time. Another difference is that much more parity messages are generated and received in high rate erasure resilient coding. Existing implementations of Reed-Solomon error correction codes, e.g., [8], do not apply well for high rate erasure resilient encoding/decoding operations.

In this work, we investigate a number of technologies, including the direct inverse of the Reed-Solomon sub-generator matrix, and scalar vector multiplication and addition, to speed up the Reed-Solomon high rate erasure encoding/decoding operations. The rest of the paper is organized as follows. We give a brief review of the Reed-Solomon high rate erasure resilient codes in Section 2. The efficient implementation technologies are examined in Section 3. Experimental results are presented in Section 4.

2. BACKGROUND: REED-SOLOMON HIGH RATE ERASURE RESILIENT CODES

High rate erasure resilient codes are block error correction codes. They may be described with parameter (n, k), where k is the number of the original messages, and n is the number of the coded messages. The high rate erasure resilient codes satisfy the property that n is much larger than k, thus the k original messages can be expanded into a much larger space of n coded messages. We define the maximum expansion ratio r of the high rate erasure resilient codes as:

$$r = n / k. \tag{1}$$

At the time of decoding, the client receives m messages, where m is a number equal to or slightly larger than k, and attempts to decode the k original messages from the m received coded messages.

The operation of the high rate erasure resilient codes can be described via matrix multiplication over the Galois Field GF(p):

$$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix} = \mathbf{G} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{k-1} \end{bmatrix}, \tag{2}$$

where p is the order of the Galois Field, $\{x_0, x_1, \cdots, x_{k-1}\}$ are the original messages, $\{c_0, c_1, \cdots, c_{n-1}\}$ are the coded messages, and $\mathbf{G}$ is the generator matrix. Each original/coded messages x_i /c_i is usually a long vector, consists of l GF(p) elements. When the client decodes from the m received coded messages, the decoding operation is solving:

$$\begin{bmatrix} c'_0 \\ c'_1 \\ \vdots \\ c'_{m-1} \end{bmatrix} = \mathbf{G}_m \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{k-1} \end{bmatrix}, \tag{3}$$

where $\{c'_0, c'_1, \cdots, c'_{m-1}\}$ are the received messages, $\mathbf{G_m}$ is a sub-generator matrix formed by the m rows of the generator matrix $\mathbf{G}$ that correspond to the coded messages. If the sub-generator matrix $\mathbf{G_m}$ has a full rank k, the original messages can be recovered.

Reed-Solomon codes use a structured generator matrix $\mathbf{G}$ on GF(p) that has the property that any k rows of the generator matrix $\mathbf{G}$ forms a $k \times k$ matrix of full rank. As a result, Reed-Solomon codes are MDS codes with the best error-correction / erasure error resilience property. Traditional, the Reed-Solomon generator matrix $\mathbf{G}$ is the Vandermonde matrix:

$$\mathbf{G} = \begin{bmatrix} 0^0 & 0^1 & \cdots & 0^k \\ 1^0 & 1^1 & \cdots & 1^k \\ \vdots & \vdots & \ddots & \vdots \\ (n-1)^0 & (n-1)^1 & \cdots & (n-1)^k \end{bmatrix}. \tag{4}$$

In Reed-Solomon error correction coding, it is customary to form a message polynomial of:

$$f(y) = x_0 + x_1 y^1 + x_2 y^2 + \cdots + x_{k-1} y^{k-1}, \tag{5}$$

and represent the parity messages as:

$$c_i = f(i), \qquad i = 1, 2, \cdots, n. \tag{6}$$

Most Reed-Solomon error correction coding operations favor the polynomial representation form of equations (5) and (6). However, in high-rate erasure resilient coding, the matrix representation form of (2)-(4) is favored.

An alternative form of Reed-Solomon codes uses the Cauchy matrix as the generator matrix:

$$\mathbf{G} = \begin{bmatrix} \mathbf{I_k} & & & \\ \frac{1}{k+0} & \frac{1}{k+1} & \cdots & \frac{1}{k+(k-1)} \\ \frac{1}{(k+1)+0} & \frac{1}{(k+1)+1} & \cdots & \frac{1}{(k+1)+(k-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{(n-1)+0} & \frac{1}{(n-1)+1} & \cdots & \frac{1}{(n-1)+(k-1)} \end{bmatrix}. \tag{7}$$

With such a Reed-Solomon code, the first k messages are original (systematic) messages (marked by the identity matrix $\mathbf{I_k}$), and the rest n-k messages are parity (coded) messages. The Cauchy matrix has been used in the Galois Field in [11][12], and has been used to define one variant of the Reed-Solomon codes in [13]. In this work, we implement the Cauchy matrix based Reed-Solomon erasure encoding/decoding operation more efficiently with scalar vector multiplication/addition operation described in Section 3.2.

It can be shown that for both Vandermonde and Cauchy based Reed-Solomon codes, the maximum expansion ratio r achievable is p/k. Thus, on GF(p), a (p,k) Reed-Solomon code can be constructed that generates at most p coded messages, regardless of the number of original messages k.

In most high rate erasure resilient coding applications, the equation (2) is usually not used to generate all the coded messages; rather, it defines a coded message space. The encoded messages are generated on-demand and one-by-one, for each multicast/broadcast message, each backup storage unit, or each hosting server. In the high rate erasure resilient decoding, k coded messages are collected, with each coded message identifiable through the row index of the generator matrix. The erasure resilient decoding operation then consists of the inversion of the sub-generator matrix $\mathbf{G_k}$, and the multiplication of the inverse with that of the coded messages:

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{k-1} \end{bmatrix} = \mathbf{G_k^{-1}} \begin{bmatrix} c'_0 \\ c'_1 \\ \vdots \\ c'_{k-1} \end{bmatrix}. \tag{8}$$

3. EFFICIENT IMPLEMENTATION OF REED-SOLOMON HIGH RATE ERASURE RESILIENT CODES

In this section, a number of technologies that speed up the Reed-Solomon high rate erasure encoding/decoding operations are examined. The first technology is the calculation of the direct inverse of the sub-generator matrix $\mathbf{G_k}$ using the structure of the Reed-Solomon generator matrix. The second technology speeds up the scalar vector multiplication and addition operations, which are the dominant operations in the erasure encoding/decoding operation.

3.1. Direct inverse of the sub-generator matrix

3.1.1 For Vandermonde matrix based Reed-Solomon codes

Let us assume that the k coded messages received are indexed as $\{t_0, t_1, \cdots, t_{k-1}\}$. The sub-generator matrix $\mathbf{G_k}$ are:

$$\mathbf{G_k} = \begin{bmatrix} t_0^0 & t_0^1 & \cdots & t_0^k \\ t_1^0 & t_1^1 & \cdots & t_1^k \\ \vdots & \vdots & \ddots & \vdots \\ t_{k-1}^0 & t_{k-1}^1 & \cdots & t_{k-1}^k \end{bmatrix}. \tag{9}$$

In [10], the inverse of the Vandermonde matrix can be calculated via:

$$\mathbf{G_k^{-1}} = \mathbf{Q}\mathbf{D}^{-1}, \tag{10}$$

where

$$\mathbf{Q} = \begin{bmatrix} q_{k-1}(t_0) & q_{k-1}(t_1) & \cdots & q_{k-1}(t_{k-1}) \\ q_{k-2}(t_0) & q_{k-2}(t_1) & \cdots & q_{k-2}(t_{k-1}) \\ \vdots & \vdots & & \vdots \\ q_0(t_0) & q_0(t_1) & \cdots & q_0(t_{k-1}) \end{bmatrix}, \tag{11}$$

$$\mathbf{D}^{-1}=\begin{bmatrix} 1/d_0 & & & \\ & 1/d_1 & & \\ & & \ddots & \\ & & & 1/d_{k-1} \end{bmatrix}, \tag{12}$$

$$\text{with} \quad d_i=\prod_{j\neq i}(t_j-t_i), \tag{13}$$

$$\text{and} \quad q_j(x)=\begin{cases} q_{j-1}(x)x+a_j, & j>0 \\ 1, & j=0 \end{cases}, \tag{14}$$

and a_j is the coefficient of the polynomial:

$$f(x)=\prod_{i=0}^{k-1}(x-t_i)=\sum_{j=0}^{k}a_j x^j. \tag{15}$$

The direct inverse computation includes[1] $k^2/2$ operations to calculate the coefficients a_j (equation (15)), k^2 operations to calculate the coefficients d_i (equation (13)), k^2 operations to calculate the elements of the matrix **Q** (equation (14)), and k^2 operations to calculate the elements of the matrix $\mathbf{QD^{-1}}$ (equation (10)). The computation complexity of the direct inverse is thus $4.5k^2$, compared with k^3 if the inverse is calculated via Gaussian elimination.

3.1.2 For Cauchy matrix based Reed-Solomon codes

The Cauchy matrix based Reed-Solomon codes consist of systematic and parity codes. Among the k received coded messages, let us assume k-s messages are original. For convenience, we move all the original messages to the front. Let the row index of the received original messages be $\{r_s, r_{s+1}, \cdots, r_{k-1}\}$, and the row index of the received parity messages be $\{t_0, t_1, \cdots, t_{s-1}\}$. Let the row index of the remaining non-received original messages be $\{r_0, r_1, \cdots, r_s\}$. The received coded messages form the sub-generator matrix $\mathbf{G_k}$:

$$\mathbf{G_k}=\begin{bmatrix} \mathbf{I_{k-s}} & \mathbf{0} \\ \mathbf{A} & \mathbf{B} \end{bmatrix}, \tag{16}$$

where **A** is a $s\times(k\text{-}s)$ Cauchy matrix of the form:

$$\mathbf{A}=\begin{bmatrix} \frac{1}{t_0+r_s} & \frac{1}{t_0+r_{s+1}} & \cdots & \frac{1}{t_0+r_{k-1}} \\ \frac{1}{t_1+r_s} & \frac{1}{t_1+r_{s+1}} & \cdots & \frac{1}{t_1+r_{k-1}} \\ \vdots & \vdots & & \vdots \\ \frac{1}{t_{s-1}+r_s} & \frac{1}{t_{s-1}+r_{s+1}} & \cdots & \frac{1}{t_{s-1}+r_{k-1}} \end{bmatrix}, \tag{17}$$

and **B** is a $s\times s$ Cauchy matrix of the form:

$$\mathbf{B}=\begin{bmatrix} \frac{1}{t_0+r_0} & \frac{1}{t_0+r_1} & \cdots & \frac{1}{t_0+r_{s-1}} \\ \frac{1}{t_1+r_0} & \frac{1}{t_1+r_1} & \cdots & \frac{1}{t_1+r_{s-1}} \\ \vdots & \vdots & & \vdots \\ \frac{1}{t_{s-1}+r_0} & \frac{1}{t_{s-1}+r_1} & \cdots & \frac{1}{t_{s-1}+r_{s-1}} \end{bmatrix}. \tag{18}$$

[1] For GF(2^q), the addition is a bitwise xor operation, the multiplication requires three table lookups and one addition. Because the Galois Field multiplication is a dominating factor in computation complexity, we mainly count the number of multiplications in the computation of the complexity.

The inverse of the sub-generator matrix $\mathbf{G_k}$ can be calculated as:

$$\mathbf{G_k^{-1}}=\begin{bmatrix} \mathbf{I_{k-s}} & \mathbf{0} \\ \mathbf{0} & \mathbf{B^{-1}} \end{bmatrix}\begin{bmatrix} \mathbf{I_{k-s}} & \mathbf{0} \\ -\mathbf{A} & \mathbf{I_s} \end{bmatrix}. \tag{19}$$

Cauchy matrix has the property that the determinant of an arbitrary Cauchy matrix **B** can be calculated as:

$$\det(\mathbf{B})=\frac{\prod_{i<j}(t_i-t_j)\prod_{i<j}(r_i-r_j)}{\prod_{i,j=0}^{s-1}(t_i+r_j)}, \tag{20}$$

Using the cofactor, the inverse of the Cauchy matrix can be directly calculated as:

$$\mathbf{B^{-1}}=\left[d_{i,j}\right]_{i,j=0,\cdots,s-1}, \tag{21}$$

where:

$$d_{i,j}=(-1)^{i+j}\frac{e_j f_i}{a_j b_i (t_j+r_i)}, \tag{22}$$

$$a_m=\left\{\prod_{i<m}(t_i-t_m)\right\}\left\{\prod_{i>m}(t_m-t_i)\right\}, \tag{23}$$

$$b_m=\left\{\prod_{i<m}(r_i-r_m)\right\}\left\{\prod_{i>m}(r_m-r_i)\right\}, \tag{24}$$

$$e_m=\prod_i(t_m+r_i), \tag{25}$$

$$f_m=\prod_i(t_i+r_m). \tag{26}$$

The direct inverse computation[2] includes $4s^2$ operations to calculate the coefficient a_m, b_m, e_m, f_m, s^2 operations to calculate the coefficient $d_{i,j}$, the computation complexity of the directly inverse is thus $5s^2$, compared with s^3 if the inverse is implemented via Gaussian elimination.

3.2. Galois Field scalar vector multiplication and addition

High rate erasure resilient coding usually involves relatively long original and coded messages, e.g., 1 kilo byte(KB) each. During the encoding/decoding operations, the message can be considered as a long vector of GF(p) elements. Therefore, one of the key operations in the erasure encoding/decoding is the following scalar vector multiplication and addition operation:

$$\mathbf{y}=\mathbf{y}+\alpha\cdot\mathbf{x}, \tag{27}$$

where **x** and **y** are vectors of l GF(p) symbols, α is a scalar number in GF(p). In erasure encoding, each erasure coded message generated via equation (2) requires k scalar vector multiplication and addition operations. The erasure decoding operation of (8) can be accomplished by first calculating the elements of the inverse of the sub-generator matrix $\mathbf{G_k^{-1}}$, and then calculating the decoded messages via k^2 scalar vector multiplication and addition operations. In Cauchy matrix based Reed-Solomon decoding, we may first calculate the elements of the matrix $\mathbf{B^{-1}}$ and **A** in equation (19), and then use $k\cdot s$ scalar vector multiplication and addition operations to calculate the decoded messages.

The scalar vector multiplication and addition operation can be efficiently calculated in GF(p) as follows:

[2] Because $\mathbf{G_k^{-1}}$ is to be multiplied with the received messages, which involve a scalar vector multiplication and addition operation, only $\log(a_k)$, $\log(b_k)$, $\log(e_k)$, $\log(f_k)$ and $\log(d_{i,j})$ need to be computed.

Preparation. Establish two lookup tables on GF(p) for the logarithmic operation:

$$\log(x)=\log x, \quad (28)$$

and the exponential operation:

$$\exp(x)=e^x. \quad (29)$$

On GF(p), each lookup tables contains p elements and only needs to be pre-calculated once.

Step 1. Test if α is 0.

If α is 0, the result of the computation (27) is simply **y**. This step is not always necessary, as certain coefficients of the generator matrix and its inverse are guaranteed to be non-zero, and need no testing.

Step 2. Calculate $\log(\alpha)$.

Step 3. Perform multiplication and addition for each symbol x_i and y_i of the message **x** and **y**. First, we test if the symbol x_i is 0. For non-zero symbol x_i, the following operation is performed to update y_i:

$$y_i = y_i + \exp(\log(\alpha)+\log(x_i)). \quad (30)$$

The operation of (30) can be accomplished via a logarithmic table lookup, an addition, an exponential table lookup, and an addition on the Galois Field. Note on GF(2^p), the final operation is a bitwise xor operation.

4. EXPERIMENTAL RESULTS

We implement the high rate Reed-Solomon erasure resilient code based on the Vandermonde and the Cauchy matrix on GF(2^{16}). We select the order of the Galois Field to be 2^{16} because of the following. The order p of a general Galois Field must satisfy:

$$p = z^q, \quad (31)$$

where z is a prime, and q is a positive integer. Nevertheless, GF(z^q) with z other than 2 does not correspond well with information representation in the computer and requires additional bit to represent the coded message, and thus should not be used. In GF(2^q), each Galois Field symbol can be represented as a q-bit binary. The symbol in GF(2^q) with q=8, 16 and 32 can be especially efficiently processed, as it corresponds to a byte (8 bit), a word (16 bit), and a double word (32 bit) in the computer. GF(2^8) can only accommodate a maximum of 2^8=256 coded messages, which is a little bit small for many high rate erasure coding applications. In GF(2^{32}), the logarithmic and exponential lookup table contains 2^{32} entries, and thus cannot be used. In stead, the multiplication operation on GF(2^{32}) has to be factored into 6 multiplication operations on the subfield GF(2^{16}), which increases the computation complexity. Thus, we implement the high rate Reed-Solomon erasure resilient code on GF(2^{16}). The maximum coded message space is 2^{16}=65536. This leads to a maximum expansion ratio of 4096 if the number of original messages is 16, and a maximum expansion ratio of 256 if the number of original messages is 256. Both of which are large enough for most applications. The logarithmic and exponential lookup tables on GF(2^{16}) contain 65536 entries, and are 128KB each.

We test our implementation by running erasure encoding and decoding operations, and observe the amount of data that can be processed each second. The parameter of the erasure resilient code is the following. The length of the message is 1KB, and the number of the original messages is 16, in accordance with the distributed content hosting application of [7]. Our implementation achieves an encoding/decoding throughput of 25MB per second, (i.e., 200Mbps) on a Pentium 2.8Ghz computer. Alternatively speaking, if the computer network operates at 10Mbps, the erasure encoding/decoding operation only takes 5% of the CPU load. We do not observe significant performance difference between the Reed-Solomon codes based on the Vandermonde matrix and the Cauchy matrix.

In [13], Bloemer *et. al.* implement a Reed-Solomon erasure encoding/decoding operation via xor operation. Their implementation requires on average q xor operation for each multiplication operation on GF(2^q). In comparison, through the scalar vector multiplication and addition operation of (27), the corresponding operation of our implementation takes one comparison to zero, two table lookups, one addition and one xor operation. We have implemented the algorithm of [13], and found that it achieves a throughput of 16MB per second. Our implementation is 50% faster than the implementation of [13].

5. REFERENCES

[1] J. Byers, M. Luby, M. Mitzenmacher and A. Rege, "A digital fountain approach to reliable distribution of bulk data", Proceedings of ACM SIGCOMM '98, Vancouver, Canada, September 1998.

[2] A. Barg and G. D. Forney, "Random codes: minimum distances and error exponents", *IEEE Trans. on Inform Theory,* Vol. 48, no. 9, Sept. 2002, pp.2568-2573.

[3] W. Tan and J. R. Cruz, "Analyzing low-density parity-check codes on partial response channels with erasures using density evolution", *IEEE Trans. on Magnetics,* Vol.40, no. 5, Sept. 2004, pp. 3411–3418.

[4] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, "Effcient Erasure Correcting Codes", *IEEE Trans. on Information Theory*, Vol. 47, No. 2, pp. 569-584, February 2001.

[5] M. Luby, "LT Codes", 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002.

[6] B. W. Stephen and V. K. Bhargava, "Reed-Solomon codes and their applications", Wiley Press, 1999.

[7] C. Zhang and J. Li, "Distributed hosting of web content with erasure coding and unequal weight assignment", accepted by Proc. IEEE International Conf. on Multimedia Expo' 2004, Taipei, Taiwan, Jun. 2004.

[8] S. Gao, "A new algorithm for decoding Reed-Solomon codes", in *Communications, Information and Network Security* (V. Bhargava, H. V. Poor, V. Tarokh and S. Yoon, Eds.), Kluwer Academic Publishers, 2003, pp. 55-68.

[9] Sharpe, *Rings and Factorization*, Cambridge University Press, Cambridge, England, 1987.

[10] I. Kaufman, "The inversion of the Vandermonde matrix and transformation to the Jordan canonical form", *IEEE Trans. On Automatic Control,* Vol. 14, no. 6, Dec. 1969, pp.774-777.

[11] D. Grigoriev, M. Karpinski, M. Singer, "Fast parallel alogorithms for multivariate polynomial over finite fields", SIAM Journal on Computing, Vol. 19, 1990, pp. 1059-1063.

[12] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance", *J. ACM*, Vol. 36, No. 2, April 1989, pp. 335-348.

[13] J. Bloemer, M. Kalfane, M. Karpinski, R. Karp, M. Luby and D. Zuckerman, "An xor-based erasure-resilient coding scheme", ICSI TR-95-048, August 1995, Berkeley, CA.