Effective method for coding RS codes using SIMD instructions

Marov Aleksei, Platonov Sergei

Raidix LLC, Saint Petersburg, Russia.
Outline

• Problem: Errors in data storage
• Method: Reed-Solomon codes
• Goal: Fast coding and Decoding
1. Background
   • Error types
   • Galois Fields arithmetic
   • Known solutions
2. Problem definition
   • Reed-Solomon Codes
3. Coding matrix
4. Recovery matrix
5. SDC detection
   • Main steps
6. Performance comparison
1. **Failure** – error at known positions

2. **Silent Data Corruption** (SDC) – error at unknown positions
\( GF(2^q) \) Galois Fields Arithmetic

Operation with data blocks:

1. Addition
2. Multiplication
3. Inversion
4. Logarithm computation
5. etc.
Known Solutions

• RAID
• Local Reconstruction Codes (LRC) – Microsoft Azure
• Hadoop Distributed Raid File System
• Jerrasure
• Intel Storage Acceleration Library
• etc

Don’t solve the problem of several SDCs or SDCs + failures
1. Background
   • Error types
   • Galois Fields arithmetic
   • Known solutions
2. Problem statement
   • Reed-Solomon Codes
3. Coding matrix
4. Recovery matrix
5. SDC detection
   • Main steps
6. Performance comparison
Problem statement

$n$ data blocks $D_0, D_1, ..., D_{n-1}$

$(D_0, D_1, ..., D_{n-1}, C_0, C_1, ..., C_{m-1})$ are stored to drives

- Calculation of $m$ checksums $C_0, C_1, ..., C_{m-1}$
- Recovery of several failed drives
- Detection and recovery of several SDCs

Desire: High speed of coding and decoding
Reed-Solomon codes

\((D_0, D_1, \ldots, D_{n-1}, C_0, C_1 \ldots, C_{m-1}) – RS code\)

\(l = \) number of failures

\(p = \) number of SDCs

If \(p + 2l \leq m\) then all the errors can be corrected
1. Background
   • Error types
   • Galois Fields arithmetic
   • Known solutions

2. Problem definition
   • Reed-Solomon Codes

3. Coding matrix

4. Recovery matrix

5. SDC detection
   • Main steps

6. Performance comparison
Coding via Coding Matrix

\[
\begin{pmatrix}
C_0 \\
C_1 \\
C_2 \\
\vdots \\
C_{m-1}
\end{pmatrix}
= 
\begin{pmatrix}
\tilde{W}_1(a^{N-1}) & \tilde{W}_1(a^{N-2}) & \cdots & \tilde{W}_1(a^m) \\
\tilde{W}_2(a^{N-1}) & \tilde{W}_2(a^{N-2}) & \cdots & \tilde{W}_2(a^m) \\
\vdots & \vdots & \ddots & \vdots \\
\tilde{W}_m(a^{N-1}) & \tilde{W}_m(a^{N-2}) & \cdots & \tilde{W}_m(a^m)
\end{pmatrix}
\begin{pmatrix}
D_0 \\
D_1 \\
D_2 \\
\vdots \\
D_{n-1}
\end{pmatrix}
\]

- \(D_i\) - data blocks
- \(C_j\) - checksums blocks
- \(\tilde{W}_k\) - basic interpolation polynomials
- \(N = n + m\)
- \(a\) – primitive element of the field
1. Background
   • Error types
   • Galois Fields arithmetic
   • Known solutions
2. Problem definition
   • Reed-Solomon Codes
3. Coding matrix
4. Recovery matrix
5. SDC detection
   • Main steps
6. Performance comparison
Notation for data recovery

\[(Y_0, ..., Y_{N-1}) = (D_0, D_1, ..., D_{n-1}, C_0, ..., C_{m-1})\] - data and checksums blocks

\[l = \text{number of failed blocks} \ (l \leq m)\]

\[k_1, ..., k_l\] - positions of failures

\[\bar{Y}_{l,1}\] - the column of failed blocks

\[Y'_{(N-l),1}\] - the column of not failed blocks
Failures recovery via Recovery Matrix

\[ \bar{Y} = R Y', \quad \bar{Y} \text{ - failed blocks, } Y' \text{ - not failed blocks} \]

\[
R = \begin{pmatrix}
\tilde{W}_1(a^{N-1}) & \cdots & \tilde{W}_1(a^{N-k_1}) & \tilde{W}_1(a^{N-k_1-2}) & \cdots & \tilde{W}_1(1) \\
\tilde{W}_2(a^{N-1}) & \cdots & \tilde{W}_2(a^{N-k_1}) & \tilde{W}_2(a^{N-k_1-2}) & \cdots & \tilde{W}_2(1) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\tilde{W}_l(a^{N-1}) & \cdots & \tilde{W}_l(a^{N-k_1}) & \tilde{W}_l(a^{N-k_1-2}) & \cdots & \tilde{W}_l(1)
\end{pmatrix}
\]

- No extra memory for intermediate computations
- Simple Recovery matrix calculation
- Good parallelization
1. Background
   • Error types
   • Galois Fields arithmetic
   • Known solutions
2. Problem definition
   • Reed-Solomon Codes
3. Coding matrix
4. Recovery matrix
5. SDC detection
   • Main steps
6. Performance comparison
SDC detection

\[(Y_0, \ldots, Y_{N-1}) = \]
\[= (D_0, D_1, \ldots, D_{n-1}, C_0, C_1 \ldots, C_{m-1}) - \text{RS code}\]

\[p = \text{maximal number of SDCs} \ (\leq \lfloor m/2 \rfloor)\]

\[j_1, j_2, \ldots, j_p - \text{SDCs positions (a priori unknown)}\]

Problem:
1. Detect the SDC presence
2. Find SDC positions
SDC detection

1. Syndrome calculation $S = VY$, where $V$ – Vandermonde matrix

2. $S \rightarrow$ Berlekamp-Massey algorithm $\rightarrow$ error locator polynomial.

3. Error locator polynomial $\rightarrow$ Chien’s Search $\rightarrow j_1, j_2, \ldots, j_p$

4. Recovery via Recovery matrix
1. Background
   • Error types
   • Galois Fields arithmetic
   • Known solutions

2. Problem definition
   • Reed-Solomon Codes

3. Coding matrix

4. Recovery matrix

5. SDC detection
   • Main steps

6. Performance comparison
Recovery performance comparison

- **R** – recovery matrix method
- **S** – matrix inversion method
- **F** – Forney Algorithm

- \( n = 128 \)
- \( m = 2 \ldots 16 \)
- Each block size = 4096 bytes
Recovery performance comparison

R – recovery matrix method
S – matrix inversion method
F – Forney Algorithm

- $n = 24 \ldots 64$
- $m = 4$
- Each block size = 4096 bytes
Conclusion
We suggest method for RS coding via matrix operations. Coding and Decoding up to 25 % faster.

Advantages
• Good parallelization
• Small memory requirements
• Strassen matrix multiplication algorithm
Thanks for your attention!

Questions please.

Marov Aleksei
marov.a@raidixstorage.com
Basic interpolation polynomials

Notation

$$W(x) = \prod_{i=1}^{n} (x - \lambda_i), \quad W_j(x) = \frac{W(x)}{x - \lambda_j}, \quad j = 1, n$$

$$\tilde{W}_j(x) = \frac{W_j(x)}{W_j(\lambda_j)} = \frac{W_j(x)}{W'(\lambda_j)}$$

$$= \frac{(x - \lambda_1)(x - \lambda_2) \cdots (x - \lambda_{j-1})(x - \lambda_{j+1}) \cdots (x - \lambda_n)}{(\lambda_j - \lambda_1)(\lambda_j - \lambda_2) \cdots (\lambda_j - \lambda_{j-1})(\lambda_j - \lambda_{j+1}) \cdots (\lambda_j - \lambda_n)} =$$

$$= w_{j,0} + w_{j,1}x + \cdots + w_{j,n-1}x^{n-1}$$

From Lagrange Interpolation
SDCs and Failures

\[ l = \text{number of failures} \]
\[ p = \text{number of SDCs} \]
\[ p + 2l \leq m \]

1. Calculate syndromes \( S = VY \)
2. Construct polynomial

\[
z(x) = \prod_{i=1}^{l} (x + a^{N-k_i-1}) = z_0 + z_1x + \cdots + z_lx^l
\]
SDCs and Failures

3. Calculate

\[ T_i = \sum_{j=0}^{l} z_j S_{i+j}, \quad i = 0, m - l - 1 \]

4. \( T \rightarrow \text{BMA} \)

5. Chien’s Search (to find SDC positions)

6. Recovery