Comparison of RAID-6 Erasure Codes

Dimitri Pertin1,2, Alexandre Van Kempen1, Benoît Parrein1 and Nicolas Normand1
1LUNAM Université, Université de Nantes, IRCCyN UMR 6597, Nantes, France
2Rozo Systems, Nantes, France

1 Reliability in distributed storage systems

Redundant Array of Independant Disks (RAID) distributes data over an array of disks to benefit from:
1. performance: striping accross multiple disks
2. reliability: compute redundant data

Two means to compute redundancy to provide reliability:
1. Replication (RAID-1)
   - n copies (e.g. 3)
   - n – 1 storage overhead
2. Erasure Coding (RAID-5,6)
   - same protection
   - only \( \frac{n}{k} – 1 \) overhead

Problem: While saving a significant amount of storage capacity, erasure coding brings complexity for encoding (writing) and decoding (reading) - a critical problem for real-time applications.

Our contribution: we propose the Mojette erasure code as a trade-off between storage consumption and performance.

2 Comparison metrics of RAID-6 codes

RAID-6 erasure code: two parity disks P and Q
- for each code, P corresponds to horizontal parity (RAID-5)
- the way Q is computed varies

Figure 1: Representation of a storage array using RAID-6 erasure coding. An array of k data disks is used to encode 2 parity disks: P and Q. Disks are fragmented into w strips. Any set of n strips involved in the encoding process forms a stripe.

Metrics: The number of operations required for:
1. Encoding P and Q
2. Updating a single data strip (diff-based)
3. Decoding when a disk set is unavailable

<table>
<thead>
<tr>
<th>Code</th>
<th>Encode P</th>
<th>Encode Q</th>
<th>Update</th>
<th>Decode from P</th>
<th>Decode from Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS</td>
<td>((k-1)w)</td>
<td>((k-1)w + (kw))</td>
<td>3 + 1</td>
<td>((k-1)w)</td>
<td>((k-1)w + (kw))</td>
</tr>
<tr>
<td>EVENODD</td>
<td>((k-1)w)</td>
<td>((k-1)w + k - 2)</td>
<td>(w + 2)</td>
<td>((k-1)w)</td>
<td>((k-1)w + 2(k-2))</td>
</tr>
<tr>
<td>RDP</td>
<td>((k-1)w)</td>
<td>((k-1)w)</td>
<td>4</td>
<td>((k-1)w)</td>
<td>((k-1)w)</td>
</tr>
<tr>
<td>Mojette</td>
<td>((k-1)w)</td>
<td>((k-1)w + k + 1)</td>
<td>3</td>
<td>((k-1)w)</td>
<td>((k-1)w)</td>
</tr>
</tbody>
</table>

Table 1: Comparison table of the XOR number required for different erasure codes for each metric. For Red-Solomon codes, extra multiplications in Galois fields are required and are symbolized by \( \Phi \). When different results are possible, the worst case is displayed.

Figure 2: RDP codes for a \((k = 5, w = 4)\) array. The figure focuses on the computation of Q. It requires \((k-1)w\) additions for both encoding and decoding.

3 Cost comparison of RAID-6 erasure codes

1 Reed-Solomon codes (algebraic representation)

\[ P_j = \sum_{i=0}^{k-1} d_{i,j}, \quad Q_j = \sum_{i=0}^{k-1} d_{i,j} \alpha^i. \]

require \(kw\) multiplications in Galois fields

2 Array codes: EVENODD and RDP

<table>
<thead>
<tr>
<th>Disk</th>
<th>Data Disks</th>
<th>Parity Disks</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3: Mojette transform of a \(3 \times 3\) image for directions \((p, q)\) in the projection set \((0, 1), (1, 1)\). Projection \((1, 1)\) could be use in the same way as \((1, 1)\) for Q.

3 Mojette erasure code

![Mojette erasure code diagram](image)

![Figure 3: Mojette transform of a 3 \times 3 image for directions (p, q) in the projection set \((0, 1), (1, 1)\). Projection \((1, 1)\) could be used in the same way as \((1, 1)\) for Q.)

Figure 4: Mojette decoding cost, depending on the position of the failed disk in the array, for \(k = 11\) and \(w = 20\). The dashed line stands for the number of XORs reached by RDP codes (i.e. \((k-1)w\)).

Conclusion:
1. Mojette erasure code requires less operations
2. But it costs a few more data in Q
3. Need to extend to further codes and parameters
4. Memory management significantly impacts perf