Comparison of RAID-6 Erasure Codes

Dimitri Pertin, Alexandre van Kempen, Benoît Parrein, Nicolas Normand

▶ To cite this version:

Dimitri Pertin, Alexandre van Kempen, Benoît Parrein, Nicolas Normand. Comparison of RAID-6 Erasure Codes. The third Sino-French Workshop on Information and Communication Technologies, SIFWICT 2015, Jun 2015, Nantes, France. hal-01162047

HAL Id: hal-01162047

https://hal.science/hal-01162047

Submitted on 13 Mar 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Comparison of RAID-6 Erasure Codes

Dimitri Pertin
Université de Nantes
IRCCyN UMR 6597
Rozo Systems

Alexandre Van Kempen
Université de Nantes
IRCCyN UMR 6597

Benoît Parrein
Université de Nantes
IRCCyN UMR 6597

Nicolas Normand
Université de Nantes
IRCCyN UMR 6597

Abstract

Coding techniques for RAID-6 storage systems, providing a double fault-tolerance, are varied. They all come with their respective benefits and limitations. In this paper, we compare the characteristics of five prominent erasure codes. We show that the general-purpose Reed-Solomon codes, either based on Vandermonde or Cauchy matrices, are outperformed by Array codes (i.e. EVENODD, RDP) which are specifically designed for RAID-6 storage systems. However codes based on a geometrical approach, such as the Mojette erasure code, show even better performances at the cost of a slight extra storage overhead. We outline the differences between these codes in terms of encoding, updating and decoding complexity. We believe that such an analysis can be valuable to system designers to figure out which code would best suit their requirements.

1 Introduction

In distributed storage systems, erasure codes are known to offer an efficient alternative to replication since they provide the same fault-tolerance while reducing significantly the storage overhead. Redundant Array of Independent Disks (RAID) is a technology that distributes data among a disk array to improve performances and/or reliability [7]. Performances are improved by striping across multiple disks while redundancy brings reliability. The well-know RAID-4 and RAID-5 schemes have been long deployed as a way to tolerate a single disk failure, without the cost of the plain replication used in the mirroring RAID-1 scheme. RAID-6 schemes have been proposed to tolerate a second failed disk. Methods for implementing RAID-6 are varied, including classic erasure coding techniques like Reed-Solomon coding and more specialized codes such as EVENODD [1] and RDP [4]. However, there is no "one-size-fits-all" approach, and all the implementations come with their benefits and limitations. As such it is important for RAID-6 storage designers to have at their disposal various coding techniques from which to choose.

In this paper, we review the traditional RAID-6 codes and provide a thorough comparison of their coding performance, namely the *encoding*, *decoding* and *updating* costs. Moreover, we add to this comparison a new code called the *Mojette* erasure code, which is based on a geometrical approach, contrary to the classic algebraic code [5]. We show that this new code leads to significant performance improvements at a price of a slight storage overhead. We believe that this new alternative can be of particular interest to storage systems designers seeking for high-performance erasure codes.

The rest of the paper is organized as follows. We first remind the basics of the RAID-6 erasure coding and introduce common notations we use throughout the paper. We then briefly describe each code, and provide their *encoding*, *decoding* and *updating* costs. Finally, we sum up and compare all these codes before concluding.

2 RAID-6 Terminology and Erasure Coding

Storage systems implementing a RAID-6 erasure code aggregate the storage space of multiple disks while protecting users' data against any double disk failures. More precisely, they are usually represented as an **array** of n disks of the same storage capacity which is composed of two parts: the first k disks contain the **data**, while the remaining $(n-k)$ disks store encoded information, also called **parity**. Each disk is divided into w **strips** of β sequential bits. Figure 1 depicts a simplified disk array where $w=2$ strips. In practice, w is usually larger than that. An encoded element has the size of a strip, and its computation is based on operations over a strip from each disk. The strips involved in this computation form a **stripe**. An (n,k) erasure code aims at providing $(n-k)$

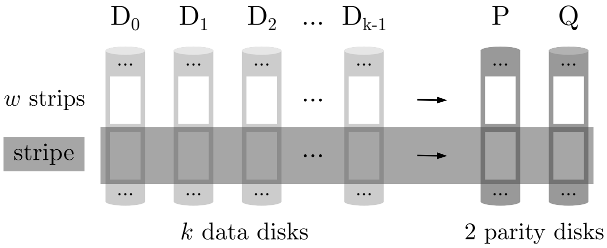

Figure 1: Representation of a storage array using RAID-6 erasure coding. An array of k data disks is used to encode 2 parity disks: P and Q. Disks are fragmented into w strips. Any set of n strips involved in the encoding process forms a stripe. Inspired by [10].

encoded strips from k data strips (where $n > k$). Particularly, **Maximal Distance Separable** (MDS) codes provide optimal reliability for a given storage overhead. They can retrieve the data strips from any subset of k strips, thus it is possible to handle any $(n-k)$ failures.

In this formalism, *RAID-6* codes are specific-purpose erasure codes that compute two parity disks (usually named P and Q) from k data disks. In coding theory words, they are defined as a $(n = k+2, k)$ erasure code, which can prevent against any two disk failures. They can be adapted from general-purpose codes such as *Reed Solomon* codes, or they can be designed for this specific aim, like *EVENODD* codes.

In the RAID-6 schemes presented in this paper, the contents of the parity drive P are calculated as the parity (XORing) of data, just as in RAID-5. The way the other parity disk Q is computed differs according to each specific code, as explained below. We compare the different codes according to the following metrics:

- *Encoding cost* is the number of operations to compute encoded data stored in P and Q, from the data contained in the k data disks.

- *Update cost* is the number of operations required to update parity disks when a single data strip is modified. We consider *diff*-based updates, where instead of re-encoding the data, we only compute the difference with the original data and apply this *diff* to the parity drives, as done in [14]. We will see that for some codes, the strip location has a significant impact on performances.

- *Decoding cost* is the number of operations to retrieve lost information either due to transient or permanent failures. In this paper, we consider that a failure leads to the whole data disk unavailability. We will see that some codes perform better when decoding using Q rather than P for a single failure.

3 RAID-6 Codes Comparison

While the most popular erasure codes are the general-purpose Reed-Solomon (RS) codes, Array codes provide better performances, but are limited to few parity disks. We describe the two main Reed-Solomon implementations (Vandermonde-RS and Cauchy-RS), and two different Array codes (EVENODD, RDP). Moreover, we show that the Mojette erasure code can offer an appealing alternative in terms of performance. Note that due to space reason, we only sketch the description of each code and refer the interested reader to their respective papers.

3.1 Standard MDS Codes

In storage systems, Reed-Solomon codes can usually be implemented in two ways. While they have formerly been relying on Vandermonde matrices, Cauchy matrices have been proposed as an efficient alternative to increase their performances.

Vandermonde Reed Solomon Reed-Solomon codes are based on linear algebra and the encoding process is determined by a **generator** matrix. Vandermonde $(n \times k)$ matrices are convenient since any sub-matrix obtained by the removal of $(n-k)$ rows is invertible. When data erasures occur, impacted lines are removed and the inversed matrix leads to decoding. It operates on binary words of β bits, where $k+m \leq 2^{\beta}$. In this code, the parameter β should fit a computer word (e.g. $\beta = 8$ or 16 or 32 bits). Let consider two sets i and j of k and w integers, thus $d_{i,j}$ corresponds to the data strip located at the column i and row j. Then, the parity strips P_j and Q_j are computed as follows:

$$P_j = \bigoplus_{i=0}^{k-1} d_{i,j}, \tag{1}$$

$$Q_j = \bigoplus_{i=0}^{k-1} d_{i,j}\alpha^i. \tag{2}$$

Standard implementations suffer from the **Galois field** arithmetic multiplications, used in Equation (2) to compute Q. To face such slow computations, alternatives have been proposed to replace those multiplications by cyclic shifts [2] or by avoiding them using look-up tables [12].

Encoding the P drive boils down to XOR k values as depicted by Equation (1) thus incurring $(k-1)$ XORs. Equation (2) shows that $(k-1)$ XORs and k multiplications are required to encode the Q drive. To encode both parity drives, one has thus iterate this process on the w strips performing a total of $2 \times (k-1)w$ XORs and $k \times w$ multiplications. The update of a single data strip impacts

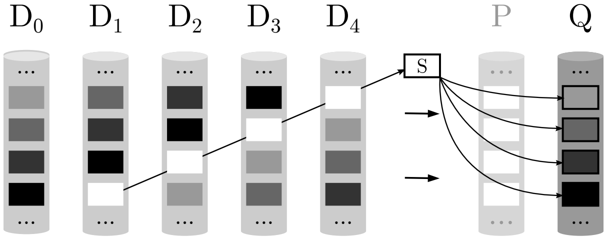

Figure 2: EVENODD codes for a $(k=5, w=4)$ array. The figure focuses on the computation of the parity disk Q using the information contained in the data disks D_i. The *adjustor* S is computed from the white diagonal. Its value is added to each strip value of Q, which are computed from diagonal parities. Inspired by [10].

two parity strips. One XOR is done to compute the data update difference, two more are required to update the related P and Q strips, and an extra multiplication is necessary for the Q strip. Thus, a data update costs 3 XORs and one multiplication. To decode data in case of a single failure, the decoding process should use P, thus, incurring $(k-1)\times w$ XORs since the multiplications required when Q is involved make the computation more expensive. The decoding process in case of a double failure is equivalent to encode the two parity drives hence requiring $2(k-1)w$ XORs and $k\times w$ multiplications.

Cauchy Reed Solomon This version of Reed-Solomon codes are based on Cauchy matrices rather than Vandermonde that eases the matrix inversion [11]. Furthermore, it expands the generator matrix by a factor of β in each direction to compute the product by fast XOR operations. Its performance is then related to the number of ones in the matrix, and no closed form of this number is known to date. Efforts were made to decrease this number by choosing sparser matrices [11].

3.2 Array Codes

Array codes were formerly designed as an alternative to Reed-Solomon codes to avoid operations over Galois fields. While Array codes are limited to few parity disks, they perform only XOR operations. P and Q are respectively generated by *horizontal* and *diagonal* stripes. For array codes, the computation of P is the same as Equation (1), what differs is how to compute Q for optimal performances.

EVENODD EVENODD codes were designed in 1995 by Blaum et. al [1]. They restrict w such that $(w+1)$ must be prime and $k\leq w+1$. When Q is involved, either for encoding or decoding, an intermediate value S (called the *adjustor*) is necessary to guarantee the MDS property. For encoding, it requires an extra amount of $(k-2)$ operations to be computed. Figure 2 depicts the process to compute Q. The strip values of Q are precomputed by XORing the elements from the different diagonals represented by different colors. Then, the value of S is computed as the XOR of the elements on the white diagonal before being added to each strip of Q.

The encoding process requires $(k-1)w$ operations to generate P and $(k-1)w+k-2$ for Q. Thus, Q is generated using more operations than P. Strip update performances depend on the location of the concerned strip. Most of the time, data strip updates affect the optimal amount of 2 parity elements. It is necessary to compute the difference between the previous and the new data value, before updating a single strip respectively in P and Q. Thus, the update cost is 3. However, when the updated data strip affects the value of S, every strip values of Q are modified. Thus w XORs are done, one XOR is required to compute the difference, and one more is necessary to update the related strip in P. Therefore, this worst case costs $(w+2)$ XORs. During decoding, several scenarios might occur. When a failure impacts a data disk, it is better to decode from P than from Q since the last one required the computation of S which brings computation overhead. A single disk failure is thus counterbalanced using $(k-1)w$ operations. When two failures affect data disks, it is first required to retrieve S. Since the computation of S depends on the erasure scheme, we call $c(S)$ the number of operations required to compute S. Let consider two integers i and j, respectively the first and second failed disk index, then:

- if $i=0$ and $j\geq k$, then we use the encoding way to compute S in $c(S)=k-2$ operations;
- if $i<k$ and $j=k$, then we compute S from any other diagonal parity stripe in $c(S)=k-1$ operations;
- if $i<k$ and $j<k$, S is computed from parity disks in $c(S)=2(k-2)$ operations.

Once S is calculated, it is possible to process a diagonal parity decoding iteration from Q. From this reconstructed strip, we can rebuild the data by an iteration from its horizontal parity strip in P. Then we alternate these iterations until the array is rebuilt. The total decoding cost is $2(k-1)w+c(S)$. Then, two data disk failures correspond to the worst case and would cost $2(k-1)w+2(k-2)$ XORs.

RDP Row Diagonal Parity (RDP) codes are the result of Corbett et. al's work in 2004 [4]. It requires $k\leq w$. They are similar to EVENODD but do not require the adjustor S. However, P is necessary computed before Q since its values are used in the computation of Q. RDP

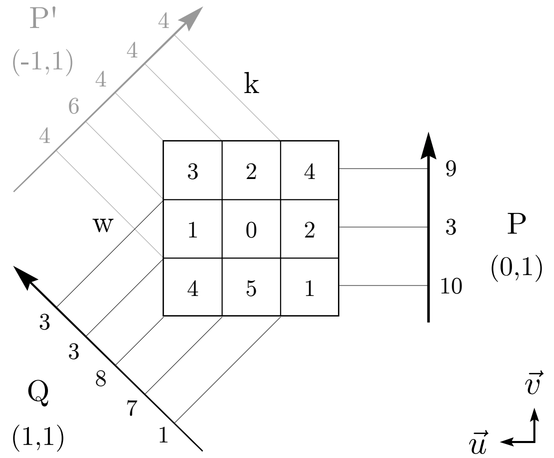

Figure 3: Mojette transform of a $k \times w = 3 \times 3$ image for directions (p,q) in the set $\{(-1,1),(0,1),(1,1)\}$. The basis is represented by $\vec{u}$ and $\vec{v}$. P, Q and P' represent a set of parity disks.

provides better encoding, updating, and decoding performance than Reed-Solomon and EVENODD codes.

For both P and Q, encoding is done in $(k-1)w$ operations. Similarly to EVENODD, updates depend on the target data strip. When it is located on the first row, or on the special diagonal, only 2 encoded strips are impacted respectively on P and Q, thus requiring 3 operations. For all the other cases however, 3 strips are modified due to the inter-dependence between both parity disks: one in P and two in Q. This situation, that corresponds to the worst case, involves 4 XORs. Decoding is done in $(k-1)w$ operations whatever parity disk used for a single erasure since the adjustor diagonal is not stored in Q strips. Thus, $2(k-1)w$ operations are required to decode from two data failures.

3.3 The Mojette erasure code

Geometric codes are an appealing alternatives to arithmetic-defined erasure code since their approach is not based on matrices or linear algebra. They are discrete versions of the *Radon transform*, a mathematic tool formerly used for tomography. Projections are computed from an array following different discrete directions. When enough projections are fetched, it is possible to uniquely inverse the operation, thus retrieving the original data [5].

Mojette A projection direction is defined by a couple of co-prime integers (p,q). The number B of elements in projections depends on the array size and the projection

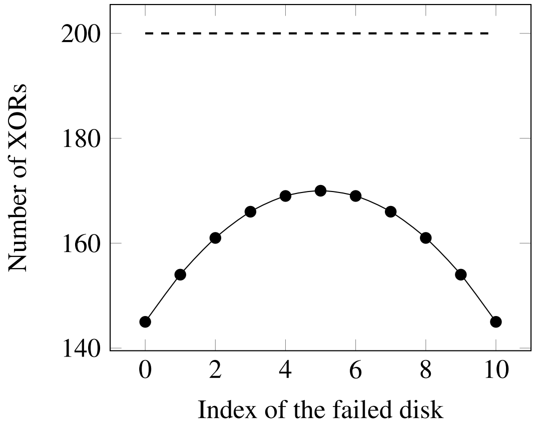

Figure 4: Mojette decoding cost, depending on the position of the failed disk in the array, for $k = 11$ and $w = 20$. The dashed line stands for the number of XORs reached by RDP codes (i.e. $(k-1)w$).

direction (p,q), given by:

$$B(k,w,p,q) = |p|\,(k-1) + |q|\,(w-1) + 1. \qquad (3)$$

Using a simplifying design that constraints $q_i = 1$, the Mojette transform behaves as an erasure code where k corresponds to the numbers of columns in the grid, and n equals the number of computed projections. Since projection sizes slightly vary depending on the related direction given, the Mojette erasure code is considered as a $(1+\varepsilon)$-MDS erasure code due to this small projection overhead [6]. Figure 3 depicts the Mojette transform of a $(k=3) \times (w=3)$ array along directions in $\{(0,1),(1,1),(-1,1)\}$. In this paper, we use a systematic version of the Mojette transform. Drive P is represented by the projection $(0,1)$ while the drive Q is represented as projection $(1,1)$. It is noticeable that the projection $(-1,1)$, depicted in gray, could be use in the same way as the projection $(1,1)$ for Q.

The number of operations per projection depends on the array size and the projection directions. It is given by:

$$c(k,w,p,q) = k \times w - B(k,w,p,q), \qquad (4)$$

where $B(k,w,p,q)$ comes from Equation (3). Since the disk P corresponds to the projection along $(0,1)$, $B(k,w,0,1) = w$. Thus P requires $(k-1)w$ operations. As we consider Q as the projection $(1,1)$, $B(k,w,1,1) = k+w-1$, and the number of XORs for its computation is $(k-1)w-k+1$. It shows that the more strips, the less operations. Since we desire to approach the MDS property, it is relevant to choose a second projection that minimizes the number of strips. Thus we choose projections $(1,1)$ (or equivalently $(-1,1)$) as Q [13]. It is then noticeable that it is faster to compute

Code	*Encode P*	*Encode Q*	*Update*	*Decode from P*	*Decode from Q*
RS	$(k-1)w$	$(k-1)w+(kw)_{\otimes}$	$3+1_{\otimes}$	$(k-1)w$	$(k-1)w+(kw)_{\otimes}$
EVENODD	$(k-1)w$	$(k-1)w+k-2$	$w+2$	$(k-1)w$	$(k-1)w+2(k-2)$
RDP	$(k-1)w$	$(k-1)w$	4	$(k-1)w$	$(k-1)w$
Mojette	$(k-1)w$	$(k-1)w-k+1$	3	$(k-1)w$	$c_{decode}(l,k,w)$ (i.e. Eq. (5))

Table 1: Comparison table of the XOR number required for different erasure codes for each metric described in Section 2. For Reed-Solomon codes, extra multiplications in Galois fields are required and are symbolized by $\otimes$. When different results are possible, the worst case is displayed (e.g. EVENODD decoding from Q depends on the computation of S).

Q than P. Additionally, note that it would be possible to choose $(-1,1)$ as P (such as represented in Figure 3) in order to reduce the number of operations, but it would cost extra storage consumption. Next, we consider P as $(0,1)$. Updates are optimal as the modification of a strip impacts only the related strips in each projections. Since we consider only RAID-6 codes, it impacts only two parity strips. In case of a single failure, it is more advantageous to decode using the Q drive instead of the P drive. Indeed, while decoding with the P drive would require $(k-1) \times w$ XORs, with the Q drive it incurs:

$$c_{decode}(l,k,w) = (k-1)w - \frac{l(l+1)}{2} - \frac{(k-l-1)(k-l)}{2}, \quad (5)$$

where $l \in [0, k-1]$ represents the missing column (*i.e.* disk). Interestingly, the required number of XORs to decode a failed disk depends on which disk has failed *i.e.* its index l, in the array. For example, we plot in Figure 4 the decoding cost for $k = 11$ and $w = 20$. This cost is maximal when the failed disk is in the middle of the array ($l = 5$ here), and then decreases (quadratically) as its position approaches the borders of the array. The minimal values are reached when the failure occurs either on the first ($l = k-1$), or the last disk ($l = 0$).

4 Discussion

Table 1 gives an overview of the theoretic performance of the codes analysed in the previous study. We express the number of XOR operations required given the metrics established in Section 2. We distinguish these metrics depending on the related parity drive since performances can differ for P and Q during encoding and decoding. It is relevant to note that P is generated in $(k-1)w$ XOR operations for each code. Similarly, decoding a failed disk from P is done in $(k-1)w$. Clearly, the cost for such operations using Q is a determining factor in this study since it varies for each code.

Reed Solomon specificities Theoretic performances of Reed-Solomon codes are hard to define since the multiplications in Galois fields can be implemented in different ways. Table look-ups do not require coding-related computations but tables have to be generated and grow significantly as the field size increases, thus impacting memory. While XOR-based Reed-Solomon implementations avoid this issue, to the best of our knowledge, there is no way to express the number of operations for a given set of code parameters. In Table 1, we distinguish multiplications from XOR operations using the $\otimes$ symbols.

Performance analysis To analyse performances related to the Q drive encoding, let consider $(k-1)w$ as a reference, reached by RDP codes. Both Reed-Solomon and EVENODD codes add extra computations: while RS code require extra multiplications, EVENODD codes require $(k-2)$ additional operations because of S. However, the Mojette code requires less operations to compute Q. Note that it is the only code that computes Q faster than P.

For decoding a failure using Q, the same statement can be done. RDP codes require $(k-1)w$ XORs to retrieve the failed disk. Decoding costs for RS and EVENODD codes are still higher, respectively due to the multiplication overhead, and the computation of S. For the Mojette erasure code, the number of required operations depends on the lost disk (see Equation (5)). Figure 4 depicts that the Mojette decoding is always lower than the $(k-1)w$ XORs reached by RDP codes.

For updating, while every code is able to reach the optimal value of 3 XORs, most of them have situations that involve more computations. Table 1 displays these worst case computations. The Mojette erasure code is the only one that achieves this optimal.

Thus, the Mojette code outperforms the other erasure codes for RAID-6 storage systems. However, it requires a slight extra storage consumption for drive Q (as depicted in Equation (3)).

Further work While this paper focused on the number of operations required for RAID-6 codes to operate encoding, updating and decoding, it has been shown that memory management significantly impacts performances [10]. Particularly, memory temporal and spatial locality are major considerations. Furthermore, the set of codes selected in this paper is limited to the most famous ones. However, some alternatives give promising results. For instance, minimal density codes such as the ones designed by Blaum and Roth [3], as well as Liberation [8] and Liber8tion [9] codes are proved to hold the minimum number of ones per row in their generating bit-matrices. Finally, while array codes are in general specific to RAID-6 systems, Reed-Solomon and Mojette codes are flexible in coding parameters and could be compared for higher parity disks.

5 Conclusion

In this study, we proposed a comparison of the number of operations required by different erasure codes, for multiple metrics. In the context of RAID-6, it shows that Array codes and the Mojette code are interesting alternatives to the classic Reed-Solomon. However, the former one lack of flexibility in the general case. Moreover, we claim that erasure codes based on a geometrical approach can outperforms classic arithmetic codes at a cost of a slight extra storage consumption. Thus, we offer to system designers a particularly suitable tool to implement high-performance erasure-coded storage systems.

6 Acknowledgements

We are grateful to Professor Jérôme Lacan for pointing out the reference of EVENODD codes [1]. This material is based upon work supported by the Agence Nationale de la Recherche (ANR) through the project FEC4Cloud (ANR-12-EMMA-0031-01).

References

[1] Blaum, M., Brady, J., Bruck, J., and Menon, J. EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures. *IEEE Transactions on Computers 44*, 2 (Feb. 1995), 192–202.

[2] Blaum, M., and Roth, R. New array codes for multiple phased burst correction. *IEEE Transactions on Information Theory 39*, 1 (Jan. 1993), 66–77.

[3] Blaum, M., and Roth, R. On lowest density MDS codes. *IEEE Transactions on Information Theory 45*, 1 (Jan 1999), 46–59.

[4] Corbett, P., English, B., Goel, A., Grcanac, T., Kleiman, S., Leong, J., and Sankar, S. Row-diagonal parity for double disk failure correction. In *Proceedings of the 3rd USENIX Conference on File and Storage Technologies* (Berkeley, CA, USA, 2004), FAST '04, USENIX Association, pp. 1–14.

[5] Guédon, J. P., and Normand, N. The Mojette transform: The first ten years. In *Discrete Geometry for Computer Imagery*, E. Andres, G. Damiand, and P. Lienhardt, Eds., vol. 3429 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 79–91.

[6] Parrein, B., Normand, N., and Guédon, J. P. Multiple description coding using exact discrete Radon transform. In *Proceedings of the Data Compression Conference* (Washington, DC, USA, 2001), DCC '01, IEEE Computer Society, p. 508.

[7] Patterson, D. A., Gibson, G., and Katz, R. H. A case for redundant arrays of inexpensive disks (RAID). In *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 1988), SIGMOD '88, ACM, pp. 109–116.

[8] Plank, J. S. The RAID-6 Liberation codes. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies* (Berkeley, CA, USA, 2008), FAST'08, USENIX Association, pp. 7:1–7:14.

[9] Plank, J. S. The RAID-6 Liber8tion code. *International Journal of High Performance Computing Applications* (2009).

[10] Plank, J. S., Luo, J., Schuman, C. D., Xu, L., and Wilcox-O'Hearn, Z. A performance evaluation and examination of open-source erasure coding libraries for storage. In *Procedings of the 7th Conference on File and Storage Technologies* (Berkeley, CA, USA, 2009), FAST '09, USENIX Association, pp. 253–265.

[11] Plank, J. S., and Xu, L. Optimizing Cauchy Reed-solomon codes for fault-tolerant network storage applications. In *NCA-06: 5th IEEE International Symposium on Network Computing Applications* (Cambridge, MA, July 2006).

[12] Rizzo, L. Effective erasure codes for reliable computer communication protocols. *SIGCOMM Comput. Commun. Rev. 27*, 2 (Apr. 1997), 24–36.

[13] Verbert, P., Ricordel, V., and Guédon, J. P. Analysis of Mojette transform projections for an efficient coding. In *Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)* (Lisboa, Portugal, Apr 2004).

[14] Zhang, F., Huang, J., and Xie, C. Two efficient partial-updating schemes for erasure-coded storage clusters. *2014 9th IEEE International Conference on Networking, Architecture, and Storage* (2012), 21–30.