International Journal of Internet and Distributed Systems, 2018, 3, 1-8
http://www.scirp.org/journal/ijids
ISSN Online: 2327-7165
ISSN Print: 2327-7157

A Network Coding Based Cloud Storage Scheme

Yantao Liu[1], Yasser Morgan[2]

[1]College of Engineering, Bohai University, Jinzhou, China
[2]Faculty of Engineering and Applied Science, University of Regina, Regina, Canada
Email: liuyantaocn@163.com

How to cite this paper: Liu, Y.T. and Morgan, Y. (2018) A Network Coding Based Cloud Storage Scheme. *International Journal of Internet and Distributed Systems*, **3**, 1-8.
https://doi.org/10.4236/ijids.2018.31001

Received: December 25, 2017
Accepted: January 30, 2018
Published: February 2, 2018

Abstract

With this paper, we propose a network coding based cloud storage scheme. The storage system is in the form of an $m * n$ data array. The n columns stand for n storage nodes, which are comprised of a part of systematic nodes storing source symbols and a part of nonsystematic nodes storing parity symbols. Every row of the data array is a (n, k) systematic Maximum Distance Separable (MDS) code. A source symbol is only involved in the encoding with the unique row; it locates at and is not used by other rows. Such a design significantly decreases the complexity of encoding and decoding. Moreover, in case of single node failures, we use interference alignment to further reduce repair bandwidth. Compared to some existing cloud storage schemes, our scheme significantly reduces resource consumption on storage, update bandwidth and repair bandwidth.

Keywords

Network Coding, Cloud Storage, MDS Code, Interference Alignment

1. Introduction

Cloud storage services, such as Dropbox, Microsoft One Drive, and Amazon S3, etc., greatly assist users to manage data at any time and from anywhere. The basic requirement of a cloud storage system is reliability, which can be generally achieved by adding data redundancy. Although the simplest way for redundancy is to store the replica of data in multiple storage nodes, coding has been proved to be more storage efficient and has been playing prominent roles in distributed storage for long time. The techniques of storage coding include redundancy array of independent disks (RAID), erasure coding, and network coding, etc.

A cloud storage system inevitably consumes a variety of network resources. In between, *storage*, *update bandwidth* and *repair bandwidth* are three important performance metrics for evaluating a cloud. Storage measures the memory space

on drives occupied by a file. Additionally, in a coding storage system, even the change of a single block of original data will outdate all coding blocks, so the system needs to initiate an update procedure to recalculate, retransmit, and replace outdated data on drives. The number of symbols transported during this procedure is defined as update bandwidth. Similar to update problem, when one or more storage nodes or drives malfunction, down or leave a cloud, the system needs to initiate a repair procedure by requesting data from survival nodes to restore the lost data within newly built nodes. The number of symbols shipped for the repair procedure is defined as repair bandwidth. So, update bandwidth and repair bandwidth measure the computation and communication cost of a cloud system for file update and data restore.

Building an efficient cloud storage system with low consumptions of storage, update bandwidth and/or repair bandwidth is an everlasting goal for a cloud storage system designer. Dimakis *et al.* [1] [2] addressed the repair problem of distributed storage. They proposed a network coding based storage scheme, named by regenerating code. With this type of code, they showed how to decrease repair bandwidth by using network coding. Dimakis *et al.* viewed the repair problem as a kind of multicast transmission. By utilizing the method of information flow graph, they found quantitative relations between storage and repair bandwidth. The seminal work of [1] [2] shed light on the potential of network coding for distributed storage. Acedański *et al.* [3] thoroughly compared the efficiency between three storage schemes, including uncoded random storage, traditional erasure coding, and random linear network coding. They made detailed calculations to the decoding probability as a function of storage space and bandwidth, and demonstrated that random linear network coding performs as well as traditional erasure coding but greatly reduces storage consumption.

Besides the theoretical studies of [1] [2] [3], many researchers put great efforts into finding or designing practical efficient cloud storage schemes based on network coding. Zakerinasab *et al.* [4] [5] proposed a method to reduce update bandwidth for network coding based cloud. The key point is to keep the coding coefficients unchanged so that a new version code word turns out to be the sum of an old version word and a difference Δ between two versions. Hence, the calculation of recoding for update could be simplified significantly. Wu and Dimakis [6] proposed a smart method based on interference alignment to reduce repair bandwidth. Interference alignment [7] is a communication technique for signal design which produces useful fading at ignorant receivers while keeps its sensitivity to intended receivers. Taking the similar idea, [6] deal with a group of equations with multiple variables so as to decrease the amount of variables. The method of [6] is usable for our storage scheme too.

Papailiopoulos *et al.* [8] proposed a simple regenerating coding scheme. In their scheme, a number of $2k$ source symbols $\beta_{11},\cdots,\beta_{1k}$ and $\beta_{21},\cdots,\beta_{2k}$ are encoded with two (n, k)-MDS encoder (See **Figure 1**). Then, the encoded words $c_1,\cdots,c_n$, and $d_1,\cdots,d_n$, as well as the sum $s_i = c_i + d_i \left(i = 1,\cdots,n\right)$ are stored in

n storage nodes with an elaborately arranged sequence as **Figure 2**. In this scheme, if one node fails, lost symbols on this node can be restored by extracting symbols from other nodes. For instance, assuming node 1 fails, the lost symbols c_1, d_2, and s_3 could be restored by accessing d_1, s_1, c_2, s_2, c_3, and d_3 from other nodes. An evident advantage of this scheme is the encoding and decoding of a source symbol is confined within one single row. In comparison, recall the scheme of [1] [2] uses inter-row references for encoding, *i.e.*, symbols at multiple rows are entangled together for encoding. From this sense, [8] is more lightweight than [1] [2]. However, there are several disadvantages with the scheme of [8] showed in **Figure 1** and **Figure 2**. First, the MDS encoder in **Figure 1** is not in a systematic form so that decoding is complex. Second, it consumes an extra of n symbols of storage than normal MDS codes. In **Figure 2**, the storage for $2k$ original symbols are $3n$ code symbols, *i.e.*, the code rate equals $(2k/3n)$, which is lower than the standard MDS code rate (k/n). Third, to rebuild a failed node in **Figure 2**, the repair bandwidth will be 6 symbols, which is more than the lower bound of minimum bandwidth regenerating code [1] [2]. Fourth, file updating is very heavy in that the update of a single source symbol β will incur with a concomitant change of $2n$ code symbols in total. Last but not the least, if there are more than one failed nodes, the system will collapse and the lost symbols could never be recovered.

Mostly motivated by [1] [8], this paper aims to decrease the resource consumption and design complexity for a cloud storage system. We contribute a network coding based cloud storage scheme. Compared to [1], the scheme is

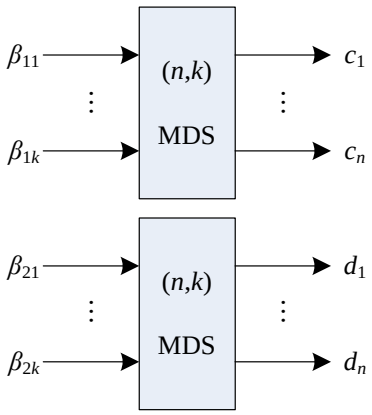

Figure 1. Two MDS encoders are used in the scheme of [8].

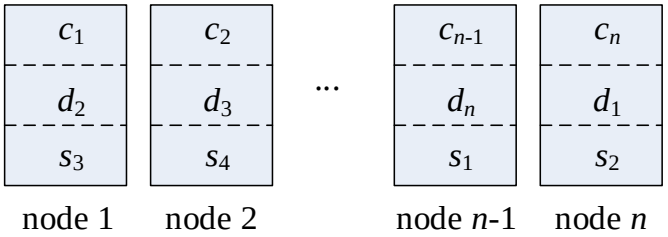

Figure 2. The arrangement of symbols in the scheme of [8].

lightweight because only intra-row encoding is permitted. Moreover, it fixes the above problems of [8]. Analysis shows our scheme significantly exceeds [8] on all performance metrics, including storage, update bandwidth, and repair bandwidth.

2. A Network Coding Based Cloud Scheme

Our scheme extends Figure 1 to the number of m (n, k)-MDS encoders. Specifically, assume the source file is composed of mk symbols over the finite field GF(q). Divide the source file into m disjoint groups, each of which consists of k symbols, say $\beta_{i1},\cdots,\beta_{ik}$, ($i=1,\cdots,m$). Then, every group is encoded with a (n, k)-systematic MDS encoder. See Figure 3. The output of the ith encoder is composed of two parts: The one is the systematic part composed of k original symbols $\beta_{i1},\cdots,\beta_{ik}$; The other is the nonsystematic part composed of ($n-k$) redundant parity symbols $c_{i1},\cdots,c_{i(n-k)}$. Within one row, parity symbols are generated by the linear combination of original symbols, *i.e.*,

$$c_{ij}=\sum_{h=1}^{k} l_{ijh}\beta_{ih} \tag{1}$$

The generator matrix of the ith MDS encoder, *i.e.*, the ith row in Figure 3, is as below.

$$G_i=\begin{bmatrix} 1 & \cdots & 0 & l_{i11} & \cdots & l_{i(n-k)1} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & l_{i1k} & \cdots & l_{i(n-k)k} \end{bmatrix} \tag{2}$$

The property of MDS requires arbitrary k columns of G_i should be independent. The arrangement of code words is illustrated in Figure 3. To highlight three parameters in Figure 3, we call it (n, k, m) code in the following. It should be stressed that there is none of cross reference between rows in Figure 3, *i.e.*, the encoding related to a source symbol β_{ij} is confined within the ith row. Such a structure is beneficial to users on two facets: On the one hand, it decreases the complexity of encoding and decoding significantly. On the other hand, the file updating of Figure 3 becomes very lightweight. To follow the change of any single original symbol, say β_{ij}, only β_{ij} and ($n-k$) nonsystematic symbols

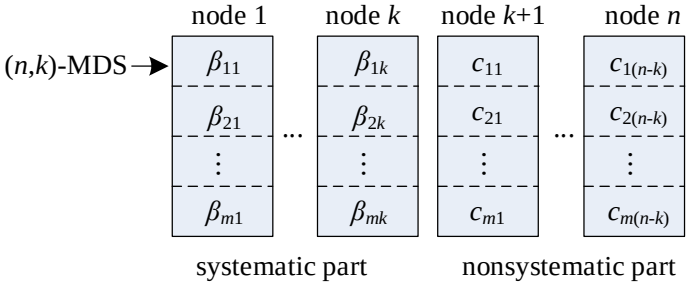

Figure 3. The array structure of (n, k, m) code. Every row is a systematic (n, k)-MDS code; Every column is a storage node. Node 1 to node k store source symbols; Node $k+1$ to node n store parity symbols.

$\{c_{i1},\cdots,c_{in}\}$ on the ith row are to be updated. All other symbols are kept intact. Thus, the update bandwidth for a single source symbol is $(n-k+1)$ symbols. Moreover, since the coding coefficients are invariant in our scheme, the method of [4] [5] can be utilized to decrease update bandwidth further.

Next, consider the repair procedure of **Figure 3**. Based on the property of MDS, any failure of less than $(n-k)$ nodes can be recovered by accessing k survival nodes. In the following, denote such a repairing way by "*regular way*". In the regular way, the repair bandwidth for any failure of less than $(n-k)$ nodes equals k nodes (or equivalently, km symbols). In comparison, we mention that the scheme of [8] is unrecoverable in front of the failures of more than one nodes.

Furthermore, the most frequently happening failures in practice are related to one node, so it is more meaningful to discuss the failures of a single node. Within this category, ones can implement linear operations on the systematic MDS codes in **Figure 3** so that further reduce repair bandwidth than *regular way* by using the method of interference alignment. Consider failures related to a systematic node first, *i.e.*, node 1 to node k in **Figure 3**. Without loss of generality, assume node 1 collapses, *i.e.*, $\{\beta_{11},\cdots,\beta_{m1}\}$ are lost. From node 2 to node $k-1$, we can take a set of symbols

$$\Lambda=\left\{\left(\beta_{12},\cdots,\beta_{m2}\right),\cdots,\left(\beta_{1(k-1)},\cdots,\beta_{m(k-1)}\right)\right\} \tag{3}$$

From node k, a sum symbol θ could be got as below

$$\theta=\sum_{i=1}^{m}\beta_{ik} \tag{4}$$

Moreover, symbols in a nonsystematic node can be combined and transformed into a linear function of Λ, θ and $\{\beta_{11},\cdots,\beta_{m1}\}$. Take the jth nonsystematic node as an example, we can build an equation

$$\sum_{i=1}^{m}\alpha_{ij}c_{ij}=f_j\left(\beta_{11},\cdots,\beta_{m1},\Lambda,\theta\right) \tag{5}$$

where α_{ij} is a coefficient assigned for c_{ij}, such that $\{\beta_{1k},\cdots,\beta_{mk}\}$ could be combined into θ. In total, we get a set of $(n-k)$ linear equations for all nonsystematic nodes as below

$$\Xi=\left\{f_1\left(\beta_{11},\cdots,\beta_{m1},\Lambda,\theta\right),\cdots,f_{n-k}\left(\beta_{11},\cdots,\beta_{m1},\Lambda,\theta\right)\right\} \tag{6}$$

By collecting the symbol of Λ, θ and Ξ, the lost symbols of $\{\beta_{11},\cdots,\beta_{m1}\}$ can be resolved and restored by solving equations in (6) given that $m\le(n-k)$. Hence, the repair bandwidth equals $1+k(m-1)-2m+n$ symbols. It should be noted that in Ξ, there may exist some linearly dependent equations which do not contribute to the solution, so the coding coefficients $\{l_{ijh}\}$ should be assigned elaborately to guarantee there are at least m independent equations in Ξ.

Next, consider the failure of a nonsystematic node. For the output of an (n, k)-MDS encoder, *i.e.*, a row in **Figure 3**, the property of MDS guarantees that every symbol can be uniquely determined by a group of k distinct symbols, no

matter these symbols are in the systematic part or in the nonsystematic part. Accordingly, any group of k symbols can be viewed as a systematic part and the other $(n-k)$ symbols as a nonsystematic part. It is to say the roles of systematic part and nonsystematic part in Figure 3 are interchangeable, so the interference alignment method is still effective to decrease repair bandwidth when a nonsystematic node fails. Moreover, the method of interference alignment is effective to repair the failures of multiple nodes if $m \le (n-k)$ satisfies. Take (7, 3, 2) code as an example. When two nodes fail, five sum symbols and four equations as (5) could be constructed from the residue nodes so that the two lost symbols could be resolved, so the repair bandwidth equals 5 symbols. However, we note that the repair bandwidth benefit in this case is got with the cost of rate loss.

Finally, an example based on (5, 3) MDS code is given to illustrate the scheme.

Example: Figure 4 is a (5, 3, 2) coding scheme over GF(7). Two systematic generator matrices are

$$G_1 = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 & 3 \end{pmatrix}, \quad G_2 = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 3 \\ 0 & 0 & 1 & 3 & 4 \end{pmatrix}$$

First, apply the method of interference alignment to repair the failure of a systematic node. Take node 1 as an example. In this case, lost symbols are β_{11} and β_{21}. So, $\Lambda = \{\beta_{12}, \beta_{22}\}$, $\theta = \beta_{13} + \beta_{23}$, and

$$\begin{cases} f_1 = 3c_{11} + c_{21} = 3\beta_{11} + \beta_{21} + 3\beta_{12} + \beta_{22} + 3\theta \\ f_2 = 4c_{12} + 3c_{22} = 4\beta_{11} + 3\beta_{21} + \beta_{12} + 2\beta_{22} + 5\theta \end{cases} \tag{7}$$

With the values of β_{12}, β_{22}, θ, f_1 and f_2, one can restore β_{11} and β_{21} by solving (7).

Next, take node 4 as an example to repair the failure of a nonsystematic node. When node 4 fails, the lost symbols are c_{11} and c_{12}. Exchange the roles of node 4 and node 3 in Figure 4 with linear operations over GF(7). As Figure 5 shows, c_{11} and c_{12} move to the systematic part. Thus, using interference alignment, we have $\Lambda = \{\beta_{11}, \beta_{21}\}$, $\theta = \beta_{12} + \beta_{22}$ and

$$\begin{cases} f_1 = \beta_{13} + 3\beta_{23} = c_{11} + c_{21} + 6\beta_{11} + 6\beta_{21} + 6\theta \\ f_2 = 2c_{12} + 3c_{22} = 6c_{11} + c_{21} + 3\beta_{11} + 6\beta_{21} + 5\theta \end{cases} \tag{8}$$

node 1	node 2	node 3	node 4	node 5
β_{11}	β_{12}	β_{13}	$c_{11}=\beta_{11}+\beta_{12}+\beta_{13}$	$c_{12}=\beta_{11}+2\beta_{12}+3\beta_{13}$
β_{21}	β_{22}	β_{23}	$c_{21}=\beta_{21}+\beta_{22}+3\beta_{23}$	$c_{22}=\beta_{21}+3\beta_{22}+4\beta_{23}$

Figure 4. An example of (5, 3, 2) code.

node 1	node 2	node 4	node 3	node 5
β_{11}	β_{12}	c_{11}	$\beta_{13}=c_{11}+6\beta_{11}+6\beta_{12}$	$c_{12}=3c_{11}+5\beta_{11}+6\beta_{12}$
β_{21}	β_{22}	c_{21}	$\beta_{23}=5c_{21}+2\beta_{21}+2\beta_{22}$	$c_{22}=5c_{21}+2\beta_{21}+4\beta_{22}$

Figure 5. Exchange the roles of node 4 and node 3 in Figure 4.

Table 1. Comparison Between Secure LNC Codes.

Performance metrics	[8]	(5, 3, 2) code
Storage	15 symbols	10 symbols
Code rate	2/5	3/5
Repair bandwidth for one node failure	6 symbols	5 symbols
Repair bandwidth for two or more node failures	Nonrepairable	6 symbols
Update bandwidth for one symbol updating	10 symbols	3 symbols

With the values of β_{11}, β_{21}, θ, f_1 and f_2, one can restore c_{11} and c_{12} by solving (8). In this example, the repair bandwidth for a single node failure is 5 symbols. Recall that 6 symbols are needed if we use the *regular way* or the scheme of [8].

Consider the update procedure of **Figure 4**. Take the update of β_{11} as an example, only β_{11}, c_{11} and c_{12} need to be updated, so the update bandwidth related to one symbol in this example equals 3 symbols. While, the update bandwidth of [8] equals 10 symbols.

Last, with this example, a comparison is made between [8] and our scheme within **Table 1**. Both take (5, 3)-MDS code. In [8] or **Figure 2**, the needed space for storing 6 source symbols are 15 symbols, so the code rate equals 2/5; In our scheme, it needs 10 symbols, so the rate equals 3/5. To repair the failure of one node, [8] needs to ask 6 symbols from survival nodes; our scheme only needs 5 symbols by using interference alignment. [8] cannot recover from the failures of more than two nodes; On the contrary, our scheme can survive from the failures of two nodes with the regular way of repair. Due to the usage of nonsystematic MDS code in **Figure 1**, any change of a source symbol cause concomitant changes of 5 code symbols and 5 sum symbols in **Figure 2**, so the update bandwidth equals 10 symbols; While, as mentioned, only 3 symbols in our scheme change to follow the change of a source symbol. As a result, our scheme outperforms [8] in all performance metrics.

3. Conclusion

With this paper, we propose a network coding based cloud storage scheme. The key points of our scheme include systematic MDS code and none of inter-row reference for encoding. Moreover, the method of interference alignment is utilized to reduce repair bandwidth in the case of single node failures. These techniques bring significant advantages to a cloud storage network with respect to system simplicity, resource consumption, and communication loads, etc. Detailed analysis and an example show that our scheme keeps the simplicity of [8] but is superior to that scheme from all performance metrics of storage, update bandwidth, and repair bandwidth, especially for the case of single node failure. Moreover, because there is no inter-row reference, our scheme is more lightweight and flexible than [1].

Acknowledgements

This work is supported in part by NSFC with No. 61471045 and Natural Science

Foundation of Liaoning Province with No. 20170540008.

References

[1] Dimakis, A.G., Godfrey, P.B., Wu, Y., Wainwright, M.J. and Ramchandran, K. (2010) Network Coding for Distributed Storage Systems. *IEEE Transactions on Information Theory*, **56**, 4539-4551. https://doi.org/10.1109/TIT.2010.2054295

[2] Dimakis, A.G., Ramchandran, K., Wu, Y. and Suh, C. (2011) A Survey on Network Codes for Distributed Storage. *Proceeding of the IEEE*, **99**, 476-489. https://doi.org/10.1109/JPROC.2010.2096170

[3] Acedański, S., Deb, S., Médard, M. and Kötter, R. (2005) How Good Is Random Linear Coding Based Distributed Networked Storage. *Proceeding of the First Workshop on Network Coding* (*NetCod*'05), Riva del Garda, April 2005.

[4] Zakerinasab, M.R. and Wang, M. (2012) An Update Model for Network Coding in Cloud Storage Systems. The 50*th Annual Allerton Conference on Communication, Control, and Computing*, Allerton, 1-5 October 2012, 1158-1165. https://doi.org/10.1109/Allerton.2012.6483349

[5] Zakerinasab, M.R. and Wang, M. (2013) DeltaNC: Efficient File Updates for Network-Coding-Based Cloud Storage Systems. *IEEE* 21*st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*. San Francisco, 14-16 August 2013, 360-364.

[6] Wu, Y. and Dimakis, A.G. (2009) Reducing Repair Traffic for Erasure Coding-Based Storage via Interference Alignment. 2009 *IEEE International Symposium on Information Theory (ISIT*'09*)*, Seoul, 28 June-3 July 2009, 2276-2280. https://doi.org/10.1109/ISIT.2009.5205898

[7] Cadambe, V.R. and Jafar, S.A. (2008) Interference Alignment and Degrees of Freedom of the K-User Interference Channel. *IEEE Transactions on Information Theory*, **54**, 3425-3441. https://doi.org/10.1109/TIT.2008.926344

[8] Papailiopoulos, D.S., Luo, J., Dimakis, A.G., Huang, C. and Li, J. (2012) Simple Regenerating Codes: Network Coding for Cloud Storage. *INFOCOM,* 2012 *Proceeding IEEE,* Orlando, 25-30 March 2012, 2801-2805. https://doi.org/10.1109/INFCOM.2012.6195703