TELKOMNIKA, Vol. 11, No. 12, December 2013, pp. 7689~7698
e-ISSN: 2087-278X

NCStorage: A Prototype of Network Coding-based Distributed Storage System

Wang Lei[1], Yang Yuwang*[2], Zhao Wei[3], Lu Wei[4]
Computer Science and Engineer School, Nanjing University of Science and Technology,
200 Xiaolingwei street, Nanjing, Jiangsu, China. telp/fax: +86-025-84315660/84315660
Corresponding author, e-mail: kingstones@163.com[1], yuwangyang608@yahoo.com.cn[2],
zhaoweinjust@163.com[3], 308060963@njust.edu.cn[4]

Abstract

Recent studies have shown that network coding can improve the performance of the distributed storage systems. However, most of these studies are theoretical which mainly focus on the bandwidth efficiency. This paper aims to provide a practical system, so NCStorage, a network-coding-based distributed storage system, is implemented. NCStorage implements network coding on the Internet, so users from all over the world can access it. Unlike traditional technologies such as erasure coding and fountain coding, re-encoding operation at storage servers is required by NCStorage. We observe that, benefiting from the re-encoding at the storage servers, the required repair bandwidth employed to repair a failed storage server is reduced, the computation overhead is balanced, and the security is enhanced. Both the encoding at the clients and the re-encoding at the storage servers are based on a deterministic algorithm. Finally, we deploy 8 storage servers in different places to evaluate the performance of the NCStorage, and the experimental results validate the analysis results.

Keywords: *network coding, distributed storage system, data repair, regenerating code*

1. Introduction

Currently, more and more data (such as large scale files, video data and sensory data) need to be stored. Therefore, how to effectively manage these data becomes a challenge. In order to protect the data from permanent failures of storage servers, the distributed storage systems [1] (DSS) are generally employed in practical applications. In DSS, the files are stored in different servers, so that when a small part of servers fails, the original files can be recovered from the surviving servers. The most common method for DSS is replication. In a replication-based DSS, there may be multiple copies of an original file, so this method could enhance the reliability. However replication will also introduce more data redundancy.

Erasure coding [2] [3] is another useful technology for DSS. It divides the original file into N encoded sub-files, in which any *K* out of these *N* sub-files can recover the original file, and the codes can tolerate N-K failures of the storage nodes. Therefore, erasure coding can significantly reduce the redundancy without sacrificing the reliability. For this reason, erasure coding is widely employed in practical applications. However, erasure coding has its disadvantage. When a storage node fails, the failed sub-file needs to be repaired at a new storage server so that the level of reliability can be maintained. To this end, the client needs to download at least *K* different sub-files, recover the original file, calculate a regenerating code, and then regenerate a new sub-file with the original file and the regenerating code.

Therefore, how to reduce the required repair bandwidth becomes an interesting topic [4] **Error! Reference source not found.**. Recently, network coding, proposed by Ahlswede et al. [6] , is considered as a promising technology for distributed storage systems. A significant feature of network coding is that re-encoding operations at intermediate nodes are introduced. Benefiting from the re-encoding operations, the performance (such as throughput and energy efficiency) of multicast networks is improved. Then, Li et al. **Error! Reference source not found.** proved that linear network coding is sufficient to achieve the multicast capacity equal to the minimal max-flow from the source node to different sink nodes. Ho. T et al. [8] proposed the random linear network coding (RLNC) in which the encoding vectors of the generator matrix are randomly obtained from a specified finite field. Based on RLNC, Chou et al. [9] proposed a

Received June 30, 2013; Revised August 22, 2013; Accepted September 5, 2013

practical network coding scheme which has been widely employed in subsequent studies [10-12]. Moreover, there are also some coding schemes based on deterministic algorithms in previous studies [13-15]. On the aspect of data coding and storage, the network coding-based DSS is similar to that based on traditional erasure coding, and the improvement of the network performance mainly benefits from the re-encoding operation of the intermediate nodes. In recent years, the re-encoding at intermediate nodes has been regarded as a powerful way to improve the performance of data repair, which can not be achieved by traditional erasure coding. Some theoretical studies [16-18] show that, benefiting from the re-encoding operation at intermediate nodes, network coding could reduce the required repair bandwidth. In the field of practical application, Y Hu et al proposed two prototypes in their studies [18] [20] , and they addressed some practical problem of network coding-based DSS such as the architecture and the code construction.

To the best of our knowledge, the NCFS (network coding-based file system) proposed by Hu Y et al. [18] was the first practical application of network-coding-based DSS in which a practical and extensible platform was provided for realizing the theory of regenerating codes. Nevertheless, the studies on practical network coding-based DSS are still very few. Some our work is inspired by the work of Hu Y et al.. For example, NCStorage also employs a layered design which enables extensibility, and the function for each layer can be extended without affecting the logic of other layers, so NCStorage can be rebuilt into some other systems such as cloud storage and big data storage as long as the interfaces are provided. NCFS mainly focuses on the bandwidth efficiency. In addition to the bandwidth efficiency, the advantages brought by NCStorage such as load balancing are addressed in this paper. Moreover, there is a difference between their work and ours. Like some other studies [16] **Error! Reference source not found.**, we also assume that the storage nodes are programmable, which is not required in NCFS. Because the performance gain of NCStorage is mainly generated by the re-encoding at storage servers, we have to ensure that the storage servers are programmable. In NCStorage, the storage servers are regarded as intermediate nodes, and the re-encoding operations are performed on these servers.

Our contributions can be summarized as follows: Firstly, we propose NCStorage, which is a prototype of network-coding-based DSS. Secondly, re-encoding operations at intermediate nodes are employed and implemented in NCStorage, which can significantly improve the performance of data repair. Thirdly, some issues load balancing and security of files are elaborated on from the perspective of practical application.

2. Preliminaries

Network coding consists of three steps: encoding at the source node, re-encoding at the intermediate nodes and decoding at the sink nodes. In order to increase readability of this paper, we will also start with traditional linear network coding **Error! Reference source not found.**. The principle of linear network coding is that the packets at the source are linearly mixed with some original packets.

In a multicast network, when the source node starts to transmit some original packets to a group of sink nodes, in order to ensure all sink nodes receive the packets at the multicast capacity (h), the source node needs to encode the original packets with an algebraic approach. For example, if the multicast capacity is h, in each transmission period, the source node needs to select h original packets ($p_1, p_2, ..., p_h$) to encode. According the number n of outgoing links of the source node, the source node needs to obtain an $n \times h$ generator matrix G. After using a random or deterministic algorithm, a generator matrix G can be obtained as follows:

$$G = \begin{pmatrix} a_{11} & \cdots & a_{1h} \\ \vdots & \ddots & \vdots \\ a_{h1} & \cdots & a_{hh} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nh} \end{pmatrix} \quad (1)$$

Then the n new packets $Y_k, (k \in (1,...n))$ can be obtained by the following equation:

$$Y_k = \sum_{i=1}^{h} (a_{ki} \times p_i), k \in [1, n] \quad (2)$$

After obtaining the new packets, the source node sends them out along with corresponding coefficient vectors from its outgoing links, respectively. When a sink node receives the new packets or newer packets re-encoded by the intermediate nodes, the received coefficient vectors in different packets form a receiving matrix R. According to the matrix R, the sink node can recover the original packets ($p_1, p_2, ..., p_h$) by the Gauss elimination method. Notice that the re-encoding at the source nodes is similar to that in traditional erasure coding.

In classical network coding schemes, the nodes in the network are always divided into three categories: namely, the source nodes, the intermediate nodes and the sink nodes. Accordingly, network coding consists of encoding at the source nodes, re-encoding at the intermediate nodes and decoding at the sink nodes. In NCStorage, these concepts are somewhat different. We will firstly introduce how these roles correspond to the entities in NCStorage. In NCStorage, the clients in which the applications are installed are considered as the source nodes, and the storage servers on the Internet are considered as the intermediate nodes. After the encoding operation, the source nodes need to upload the encoded sub-files to different servers. On the other hand, during the downloading process, the encoded sub-files are downloaded to the clients from different storage servers, and the clients can be considered as sink nodes at this stage. Therefore, NCStorage regards the clients as both the source nodes and the sink nodes, but in different stages. We have mentioned that the advantages of network coding always come from the re-encoding operations at intermediate nodes. In a network coding-based DSS, the storage servers will overtake the work of re-encoding, and then several advantages are achieved. We will conduct detailed discussion in Section 3.

3. Design and Implementation of NCStorage

3.1. Architecture of the NCStorage

NCStorage is a prototype system which is available to the users from all over the world as long as they download the client program. The architecture of NCStorage is shown in Figure 1.

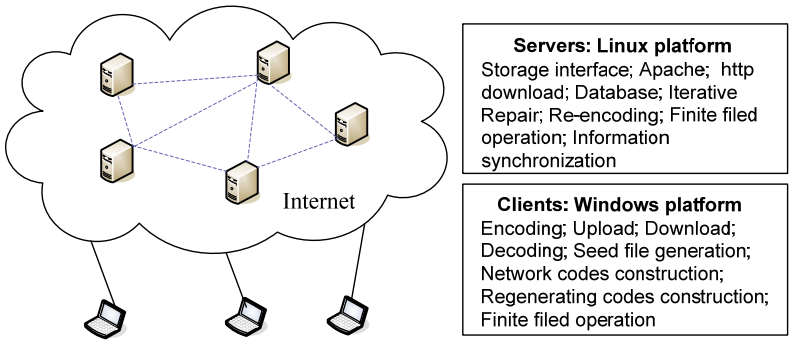

Figure 1. Architecture of NCStorage

For the servers, the storage interfaces should be provided to respond the sub-files uploading request of the clients. Through these interfaces, the servers can receive the files and store them. The uploaded files should be downloadable once they are completely received by the storage servers. In NCStorage, the repair operations are performed at the servers, and they must join into the repair work in consequence since the servers are deployed in different places. For the clients, they are able to encode the original files, and then upload them to the servers. After encoding each original file, a seed file will be generated for the original file. In the seed file, the parameters such as N, K and m (the order of the employed Galois field) as well

as the locations where the sub-files are stored are recorded. When the user needs to retrieve an original file, it is required to open the corresponding seed file, and at least K sub-files should be selected and downloaded so that the original file can be recovered. The seed file plays an important role in NCStorage. In addition to the downloading operation, it is also required when the user needs to repair a sub-file or add a new sub-file. After analyzing the seed file, the client will obtain the regenerating code and generate a repair scheme in which a logical repair path is included. The repair process will be initiated after the client sends the scheme to the first storage server in the repair path.

3.2. Network Coding-based Distributed Storage Technology

In a network coding-based DSS, the clients are regarded as the source nodes. Firstly, the source nodes need to split the original file of length L into K segments, each with a length of L/K. Secondly, according to the requirement of the user, the source nodes need to obtain an $N \times K$ generator matrix G to establish the network codes. In NCStorage, the Vandermonde matrix is employed to obtain the generator matrix G because this matrix has the MDS (Maximum Distance Separable) property. For the $N \times K$ generator matrix G, N different elements $a_1, a_2, ..., a_N$ must be selected from a finite field, and the i^{th} row vector in G is $a_i^1, a_i^2, ..., a_i^K (i \in [1, N])$. NCStorage must ensure that the field size is larger than N, which guarantees that some additional elements can be selected to construct the regenerating codes when the user intends to repair some sub-files. We consider that using Vandermonde matrix to be the generator matrix is a deterministic scheme since the elements in the matrix are organized in a manual manner. In fact, the widely used RLNC scheme can also be supported by NCStorage. However, the deterministic scheme is preferred in NCStorage. Firstly, although it is widely proved that the generator matrix obtained in a randomized algorithm has the MDS property with a high probability when the finite field is large enough. However, the probability can not be theoretically 1, so the client may occasionally fail to recover the original file even if it downloads K sub-files. Secondly, a large finite field requires much more memory space, which is not reasonable in a practical network coding-based DSS. Thirdly, if a randomized algorithm is employed, the encoding vectors in the generator matrix must be recorded in the sub-files so that the storage servers will be able to re-encode and the clients will be able to decode. When K is a big number, the length of the encoding vectors is also long, which may result in additional overhead of transmission and storage. Fortunately, the deterministic algorithm does not require additional transmission overhead or storage overhead. NCStorage only needs to store the first elements of the row vectors, and the other $K-1$ elements for each row vector can be easily deduced. In addition, the main elements are used as a part of the file names of the encoded sub-files. Therefore, in the deterministic algorithm, the transmission and storage overhead of encoding vectors is saved.

3.3. Network Coding-based Decentralized Iterative Repair Technology

Data storage is the main function of DSS, with data repair as another important function. NCStorage implements three kinds of data repair: exact repair, functional repair **Error! Reference source not found.** and new sub-file regeneration. For the exact repair, when a storage server suffers from a performance failure, NCStorage needs to repair the sub-file in the failed server, and the sub-file stored in a new storage server must be the same as that in the failed server. For the functional repair, the requirement is released. As long as the MDS property is maintained, the repaired sub-file is considered valid. For the third kind, if the user needs to enhance the level of reliability of their file, a new sub-file needs to be added to a new storage server. And then the (n,k) codes become $(n+1,k)$ codes. If the operation of adding a new sub-file is performed for j times, the (n,k) codes will become $(n+j,k)$ codes.

As mentioned above, for each original file, NCStorage will create a seed file in which the values such as N, K, m and the server address for each sub-file are also stored. The seed file will be accessed under two situations: Firstly, when the user needs to retrieve an original file, it is required to open the seed file. Secondly, if the user needs to repair the sub-file in a failed server, the seed file is also required. According to the seed file, NCStorage can obtain the regenerating code, and the algorithm employed by NCStorage is as follows:

Step 1: The user needs to determine the repair kind, and then the first element $b,(b \in GF(2^m))$ of corresponding vector for the target sub-file will be determined as well.

Step 2: NCStorage obtains other K main elements of K surviving sub-files from the seed file to re-construct the K encoding vectors $\alpha_i = a_i^1, a_i^2, ..., a_i^K (i \in [1, K])$, and meanwhile the addresses of the K corresponding sub-files can be obtained.

Step 3: The target sub-file has the main element b, and therefore the corresponding encoding vector is $\beta = (b^1, b^2, ..., b^K)$. Then, NCStorage will calculate K values $(x_1, x_2, ..., x_K)$ such that the equation $\beta = x_1\alpha_1 + x_2\alpha_2 + ... + x_K\alpha_K$ holds. These K values can be easily obtained by solving a system of linear equations. Obviously, the group of data $(x_1, x_2, ..., x_K)$ must exist since the matrix $[\alpha_1, \alpha_2, ..., \alpha_K]^T$ is a Vandermonde matrix which must be a full rank matrix.

By the above algorithm, the regenerating code will be obtained by the client. Note that NCStorage does not require any transmission before obtaining the regenerating code. After the regenerating code is obtained, the next step is to generate the target sub-file. In traditional erasure codes, it is required to download the K encoded sub-files. In NCStorage, we implement a network coding-based scheme which requires less bandwidth to generate the target sub-file.

It should be noted that the regenerating code is obtained at the client, but the repair operations are performed at the intermediate nodes. NCStorage employs an iteration method to repair the failed storage nodes, which is explained by the diagram below.

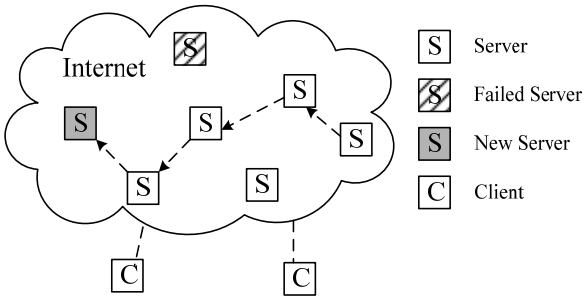

Figure 2. Repair Process of NCStorage

In the example shown in Figure 2, the sub-files are stored in six storage servers, and any four out of the six sub-files suffice to recover the original file. The seed file for each original file is kept in the corresponding client. If a storage server fails, the client needs to select a new storage server, and a repair scheme should be generated in which a logical repair path is included. Then, the client sends the scheme to the first storage server in the repair path. Each storage node i in the repair path must finish three functions. Firstly, multiply the sub-file $\alpha_i (i \in [1, K])$ it stores with corresponding coefficient $x_i (i \in [1, K])$ obtained at Step 3. Secondly, add the result to the file received from its upstream node. Thirdly, forward the new file to its downstream node until the file is received by the new storage server. After the new sub-file is stored in the new storage server, the client will update the seed file. For the exact repair, the client will update the address of the sub-file with the address of the new storage server; for functional repair, the client will remove the address of the failed file, and add the new file address into the seed file; for adding a new sub-file, the client will add a new file address into the seed file.

In our opinion, the repair operation of erasure coding-based DSS is a centralized method since at least K sub-files must be downloaded to the client, and the computation overhead of decoding and regenerating is overtook by the client. On the contrary, the repair method in NCStorage can be considered a decentralized one since it does not require gathering the sub-files to a client, and the computation overhead is shared by multiple storage servers.

In classical studies on network coding, network coding is always employed to improve the throughput, reliability and energy efficiency, and these advantages always benefit from the re-encoding operations at the intermediate nodes. In our system, the storage servers are regarded as the intermediate nodes, and they are required to implement re-encoding for iterative repair. Then, we need to discuss that in NCStorage, what advantages can be provided by using the storage servers as the intermediate nodes for re-encoding.

NCStorage provides an interface to show the regenerating process, which is shown as follows.

Figure 3. Practical Interface of the Client

On the repair path, there are six nodes, and the red node is the new storage serve. In Figure 3, the third node has finished the re-encoding operation, and it is sending the encoded file to next hop.

3.4. Bandwidth Advantage of NCStorage

In the example shown in Figure 2, N is 6, and K is 4. Therefore, there are 6 storage servers storing the 6 sub-files, and the original file can be recovered with any 4 sub-files. If the intermediate nodes are not allowed to re-encode, at least 4 sub-files must be sent to the client. Then, the client must recover the original file, and regenerate a new sub-file. After using network coding, the re-encoding operations at the intermediate nodes are allowed. Only one sub-file is required to be transmitted, and the only requirement is that the sub-file must go through the other $K-1$ intermediate nodes, and the sub-file should be updated when it goes through an intermediate node. For the storage nodes, K sub-files are transmitted, which equals to that in traditional repair method. But for the network, the required bandwidth will be reduced. Take Figure 3 as example, we assume that each storage server connects to the Internet via a router, and then each router in the repair path only needs to transmit one sub-file. Therefore, the bandwidth overhead of network is reduced.

A G Dimakis et al. [18] proposed an interesting example (see Figure 1 in [18]) to show the benefit of network coding-based DSS in their paper. Inspired by the example, we are able to achieve more bandwidth advantage in NCStorage. In their example, there were two sub-files in each storage server which enables the storage servers to re-encode the two sub-files into a new file without transmission. Therefore, instead of sending the two sub-files, sending the new file may be sufficient to regenerate. In this way, the required repair bandwidth is further improved. This idea is employed by NCStorage to achieve more bandwidth advantage. NCStorage also allows the multiple sub-files of an original file to be stored in the same server, and the following diagram demonstrates the bandwidth advantage.

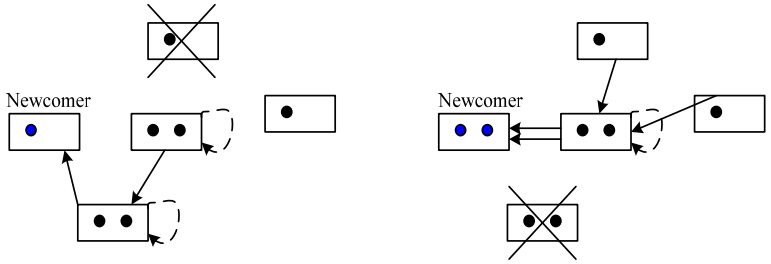

Figure 4. Storage Servers with Multiple Sub-files

In Figure 4, the original file is encoded into six sub-files, and any four out of the six sub-files suffice to recover. However, there are only four storage servers in the initial state. Therefore, two servers are allowed to store two sub-files. When a storage server fails, it is possible to regenerate a new sub-file with the bandwidth of two sub-files. Therefore, under this circumstance, the performance of network coding is better reflected since the storage node can re-encode the multiple sub-files without transmission, so the bandwidth efficiency is further improved. Even if the storage node with two sub-files fails, only 4 sub-files need to be transmitted. If traditional erasure coding is employed, 6 sub-file need to be transmitted. NCStorage adopts this strategy, there can be more than one sub-files in a storage server, but the number can not exceed $(N-K)$. After using this strategy in NCStorage, we observe that the required repair bandwidth is significantly reduced.

3.5. Load Balance Advantage of NCStorage

In traditional erasure coding-based DSS, the client should recover the original file, and then regenerate a new sub-file. In NCStorage, the decoding overhead is saved since NCStorage does not require the knowledge of the original file. Because NCStorage has avoided the overhead required by decoding while only requires the overhead of regenerating, so from the perspective of total amount of the computation overhead, the computation overhead required by NCStorage is less. Since the repair operations are performed by the intermediate nodes rather than the source node, the computation overhead of regenerating is shared by the storage servers. From the perspective of load balancing, our strategy is superior.

3.6. Security of the Repair Strategy

In traditional erasure coding, the storage nodes do not perform any coding operation. In order to generate a repaired sub-file, the source node needs to download K different sub-files. Unfortunately, the downloading operation will degenerate the security of DSS to the minimum since the original file will appear in the repair process. As long as the source node is occupied by the eavesdropper at that time, it can easily obtain some meaningful information.

In our application, NCStorage can overcome this disadvantage because our repair method is a decentralized one. It makes the repair operation to be performed on multiple intermediate nodes (namely the storage servers), which guarantees that no original data appears during the repair operations. From this perspective, the security of the original file is enhanced.

4. Experiments and Results

At current stage, there are 10 storage servers in NCStorage, and these servers are deployed in different places. After the deployment, we evaluate the performance of NCStorage. Moreover, the client we employed to conduct some local experiments is an Intel Core 2 Duo 1.8GHz computer. Figure 5 shows the main interface of the client in NCStorage.

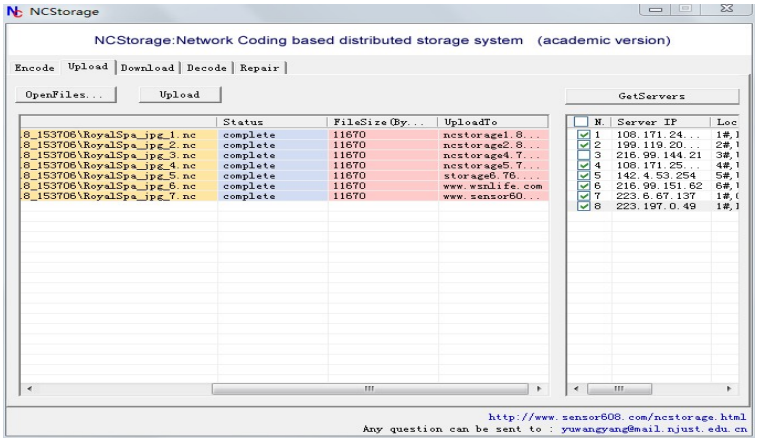

Figure 5. Main Interface of Client

4.1. Bandwidth Efficiency of Regenerating New Sub-files

The re-encoding operations at intermediate nodes can not only balance the load of network, but also reduce the required bandwidth for regenerating new sub-files. In the experiment, the original file is 10MBbytes, and it is encoded into 10 sub-files. Any 6 out of the 10 sub-files suffices to recover the original file. The experimental results are shown as follows:

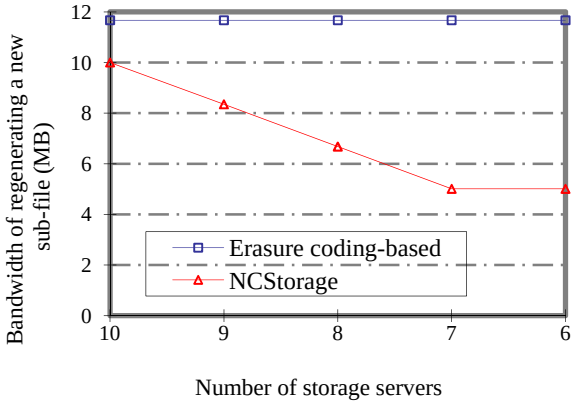

Figure 6. Bandwidth Efficiency

When the number of available storage servers is 10, i.e., each with one sub-file, the bandwidth required by NCStorage is less than that required by the erasure coding-based DSS. In this case, the bandwidth efficiency can be increased by 14.3%. When the number of available servers is less than 10, some storage servers have to store more than 1 sub-file. In this case, the bandwidth is significantly improved since each storage server on the repair path may have 2 sub-files, which enables the storage server to re-encode without additional transmission. Therefore, in a distributed storage system, when the number of storage servers is less than K, the advantage of NCStorage can be given full play to.

In accordance with all the above experiments, we know that the performance of NCStorage is much better than previous erasure coding-based DSS. Although NCStorage is implemented to be performed on the Internet, it does not require any change of hardware. Therefore, the advantages of network coding can be implemented in DSS without any restriction.

4.2. Load Balancing of NCStorage

In traditional erasure coding-based DSS, the repair work is performed at the clients. In NCStorage, the repair work is performed at K storage servers, so the load of nodes (clients and servers) in the network is balanced. Firstly, in the erasure coding-based DSS, the client needs to download K sub-files before regenerating a new sub-file, and then it will decode the original file, in NCStorage, the client does not need to download the K sub-files, which means that the decoding work is not required in the process of repair, and therefore the computation overhead of decoding is saved. Secondly, in the erasure coding-based DSS, the client needs to regenerate a new sub-file, so the client overtakes all the computation overhead of regenerating, while in NCStorage, the computation overhead of regenerating is equally distributed to K storage servers.

In the experiment, we recovered a 10Mbytes file from 6 sub-files, and there were 8 sub-files locating in different storage servers. We observe that decoding the original file costs more computation overhead than regenerating a new sub-file, as shown in Figure 7. We have mentioned that the computation overhead of decoding the original file is saved in NCStorage. Then, we conclude that when traditional erasure coding is employed, the total and average amount of computation overhead must be bigger than in NCStorage.

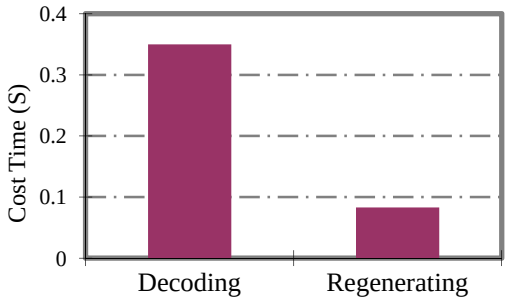

Figure 7. Overhead of Decoding and Regenerating

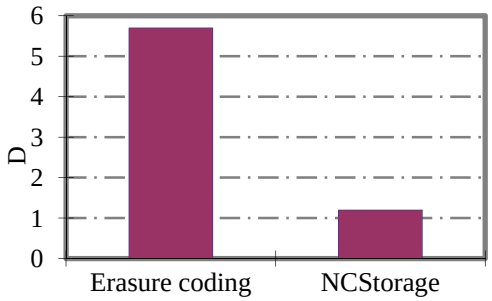

Figure 8. Load Balance of Different Method

Then, according the experimental result, we made statistic calculation of the standard deviations of these two schemes in Figure 8. We observe that the standard deviations of erasure coding-based DSS is much higher than that of NCStorage, suggesting that the computation overhead is balanced in NCStorage.

5. Conclusion

In this paper, we present NCStorage, which is a prototype of network coding-based distributed storage system. Based on the prototype, we have conducted extensive experiments. In accordance with the experimental results, we observe that NCStorage can reduce the required bandwidth of repair, increase the security of original files and balance the computation overhead. Moreover, the experimental results invalidate the anaysis.

NCStorage is implemented on the Internet, and it is designed as a public experimental platform of network coding-based DSS, all the interfaces such as encoding, decoding, file writing and file reading are provided, so it is available to other researchers and users. The researchers can easily use the resource of NCStorage to evaluate their network coding-based schemes. The software and the source codes of NCStorage are available at http://www.sensor608.com/ncstorage.html.

Acknowledgement

The work in this paper is supported by Key Technology R&D Program of Jiangsu, China (SBE201230225, SBE201000478, BE2011342). High School Research Industrialization Project of Jiangsu, China (JHZD2012-2) “333” Project of Jiangsu, China (AB41080)

References

[1] Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems*. 2008; 26(2): 4.
[2] L Rizzo, Effective erasure codes for reliable computer communication protocols. *ACM Computer Communication Review*. 1997; 27(2): 24–36.
[3] L Xu, J Bruck. X-code: MDS array codes with optimal encoding. *IEEE Trans. on Information Theory*. 1999; 45(1): 272–276.
[4] KV Rashmi, NB Shah, PV Kumar, K Ramchandran. *Explicit construction of optimal exact regenerating codes for distributed storage*. Communication, Control, and Computing, 2009. 47th Annual Allerton Conference on. IEEE. 2009: 1243-1249.
[5] Shum KW. *Cooperative regenerating codes for distributed storage systems*. Communications (ICC). IEEE International Conference on. IEEE. 2011: 1-5.
[6] Ahlswede R, Cai N. Network Information Flow. *IEEE Trans. on Information Theory*. 2000; 46(4): 1204-1216.
[7] Li SYR, Yeung RW, Cai N. Linear network coding. *IEEE Trans. on Information Theory.* 2003; 49(2): 371-381.

[8] Ho T, Médard M, Koetter R, et al. A random linear network coding approach to multicast. *IEEE Trans. on Information Theory*. 2006; 52(10): 4413-4430.
[9] Chou PA, Wu Y, Jain K. *Practical network coding*. Proceedings of the Annual Allerton Conference on Communication Control and Computing. 2003; 41(1): 40-49.
[10] H Yomo, P Popovski. Opportunistic scheduling for wireless network coding. *IEEE Trans. Wireless Commun.*, 2009; 8(6): 2766–2770.
[11] C Fragouli, J Widmer, JYL Boudec. Efficient broadcasting using network coding. *IEEE/ACM Trans. Networking*. 2008; 16(2): 450–463.
[12] Y Yang, C Zhong, Y Sun, J Yang, Network coding based reliable disjoint and braided multipath routing for sensor networks. *Journal of Network and Computer Applications*. 2010; 33(4): 422-432.
[13] NJA Harvey, DR Karger, K Murota. *Deterministic network coding by matrix completion*. Proc. 16th Annual ACM-SIAM Symposium on Discrete Algorithms. 2005; 489-498.
[14] P Sanders, S Egner, L Tolhuizen. *Polynomial time algorithms for network information flow*, in Proc. 15th ACM Symp. Parallel Algorithms and Architectures. 2003; 286–294.
[15] S Jaggi, P Sanders, PA Chou, M Effros, S Egner, K Jain, LMGM Tolhuizen. Polynomial time algorithms for multicast network code construction. *IEEE Trans. Inf. Theory*. 2005; 51(6):1973-1982.
[16] AG Dimakis, PB Godfrey, Y Wu, M Wainwright, K Ramchandran. Network coding for distributed storage systems. *IEEE Transactions on Information Theory*. 2010; 56(9): 4539-4551.
[17] Y Hu, Y Xu, X Wang, C Zhan, P Li. Cooperative recovery of distributed storage systems from multiple losses with network coding. *IEEE JSAC*. 2010; 28(2): 268–276.
[18] Dimakis AG, Ramchandran K, Wu Y, et al. *A survey on network codes for distributed storage*. Proceedings of the IEEE. 2011; 99(3): 476-489.
[19] Y Hu, HCH Chen, PPC Lee. *NCCloud: Applying Network Coding for the storage repair in a Cloud-of-Clouds*. Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST '12), CA. 2012.
[20] Y Hu, CM Yu, YK Li, PPC Lee, JCS Lui. *NCFS: On the Practicality and Extensibility of a Network-Coding-Based Distributed File System*. In Proc. Of NetCod. 2011: 1-6.