March - April 2020
ISSN: 0193-4120 Page No. 4651 - 4657

Analysis of Recovery Characteristics of Erasure Coding and Replication

Jeong-Joon Kim[1], Dong-Oh Kim[2], Heong-Yeon Kim[2], Seung-Hwa Lee[3], Jeong-Min Park[*4]
[1]Professor, Department of Software, Anyang University, 22 Samdeok-ro 37 beon-gil, Manan-gu, Anyang, Gyeonggi-do, 14028, South Korea
[2]Researcher, Electronics and Telecommunications Research Institute, 218 Gajeong-ro Yuseong-gu Daejeon, 34129, South Korea
[3]Professor, Division of Information and Communication, Baekseok University, 76, Munam-ro, Dongnam-gu, Cheonan-si, Chungcheongnam-do, South Korea
[*4]Professor, Department of Computer Engineering, Korea Polytechnic University, 237 Sangidaehak-ro Gyeonggi-Do Siheung, 15073, South Korea
jjkim@anyang.ac.kr[1], dokim@etri.re.kr[2], kimhy@etri.re.kr[2], sh.lee@bu.ac.kr[3], jmpark@kpu.ac.kr[*4]

Article Info
Volume 83
Page Number: 4651 - 4657
Publication Issue:
March - April 2020

Article History
Article Received: *24 July 2019*
Revised: *12 September 2019*
Accepted: *15 February 2020*
Publication: *26 March 2020*

Abstract
The replication has been widely used to ensure data availability in a distributed file system. In addition, the erasure coding has been adopted to overcome a problem of space efficiency in the replication. However, the erasure coding has a number of performance degradation factors. In particular, the replication requires only replacement of replica and original data, whereas the erasure coding requires reading distributed data in many nodes and restoration of original data through decoding. In this paper, a number of characteristics related to fault recovery are studied prior to identifying an efficient restoration in the erasure coding. Through this, a number of characteristics that are considered when introducing the erasure coding in the distributed file system are derived. In particular, problems and considerations of the erasure coding in relation to fault recovery are verified.

Keywords: *Replication, Erasure Coding, Distributed File System, Fault Recovery*

1. Introduction

In recent years, the data replication that stores replica data over physically distributed devices has been widely used to ensure data availability in the distributed file system. The replication is a technique that stores a number of replications of original data in distributed locations and replaces the original data with replicated data during fault occurrence [1].

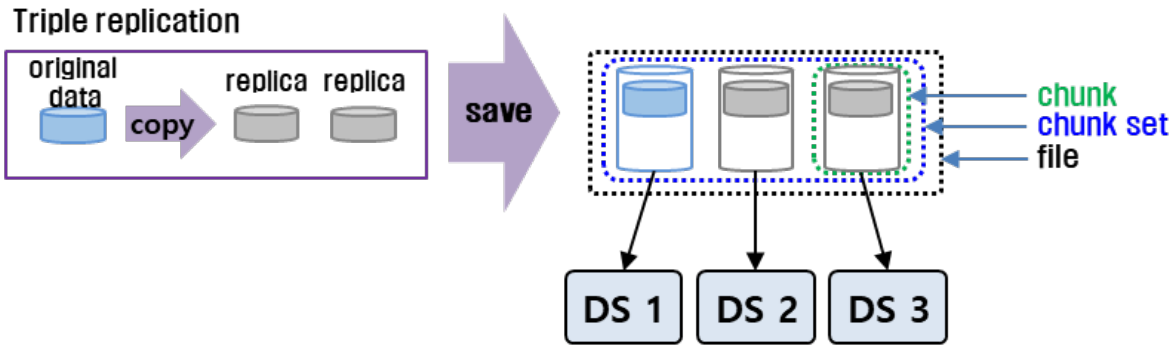

Figure 1. Example of the triple replication

March - April 2020
ISSN: 0193-4120 Page No. 4651 - 4657

Figure 1 shows an example of the triple replication, which creates two replicas of the original data. Here, a file stored in each data server (DS) is called a chunk, and a set of chunks storing original and replicated data is called a chunk set. In Figure 1, a file consisting of a single chunk set in triple replication is shown; a single chunk set consists of three chunks in the figure. Since the same data are stored in a number of servers in the replication, failures in DSs or disks can be recovered simply using replicated data [2].

However, the replication incurs a large amount of space waste due to the large volume of replicated data. As a result, the cost of system construction and management is increased rapidly, in proportion to an increase in system size. In particular, the space efficiency problem becomes more important when higher availability is required. To overcome the space efficiency problem, the Erasure Coding (EC) has been adopted [3,4]. The EC has been used in many file systems, including HDFS [5], GlusterFS [6], and Ceph [7]. The EC is to encode data using erasure code and recover original data through decoding upon fault occurrence [8].

The EC generates M parities through encoding after dividing the original data into K datasets. This is called K+M EC. The encoding is performed at a unit of encoding called a stripe, which refers to a set of original data and parity data blocks related to a single encoding operation. Figure 2 shows the distributed file system that supports the EC.

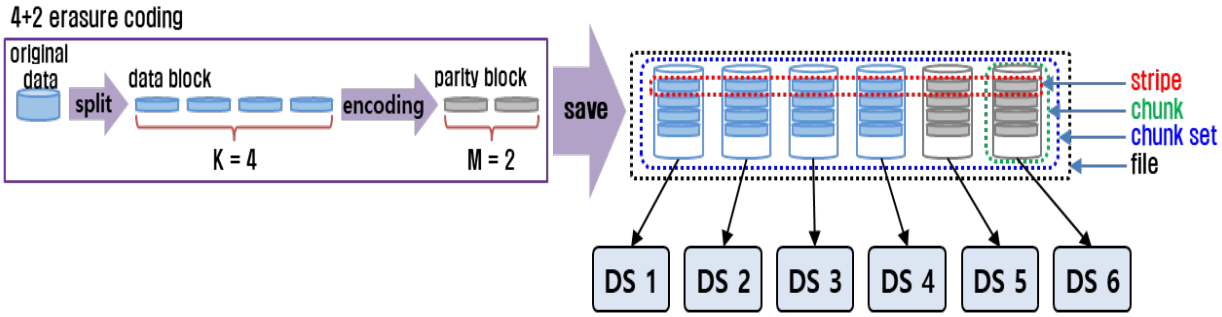

Figure 2. Example of the 4+2 EC

Figure 2 shows an example of 4+2 EC, which generates two parity blocks through encoding after dividing the original data into four datasets. The generated 4+2 blocks are divided and then stored in different DSs, and each file that is stored in each DS is called a chunk, and a set of chunks (which stores a parity block and original data block) is called a chunk set. In the EC, when the original data block is lost due to the failure of DS or disks, another original data block and parity block located in another DS is read and decoded to recover the original data.

The EC has high space efficiency. However, since it requires data division and encoding, a number of performance degradation factors are present, such as parity calculation, data distribution cost, small I/O problem, read-modify-write, and degraded I/O (repair I/O) [9-11]. This study analyzes various characteristics of the EC in comparison with the replication to derive the problems prior to studying the recovery method using EC. In particular, this study discusses the characteristics, problems, and consideration in the EC in relation to fault recovery.

2. Fault tolerance of EC and replication

To ensure high availability, multi-fault tolerance should be provided in a distributed file system. The number of faults that are processed is dependent on the number of replicated datasets in the replication and the number of parities in the

March - April 2020
ISSN: 0193-4120 Page No. 4651 - 4657

EC. The higher the number of replicated datasets or parities is, the stronger the fault tolerance is. That is, the higher the fault tolerance is, the higher the data availability is in both the replication and EC.

However, as fault tolerance increases, space efficiency is degraded due to the increasing replicated data or parities. The space efficiency in a storage is represented by a proportion of original data out of the entire storage capacity in general. Figure 3 shows the space efficiency according to fault tolerance of replication and EC.

As shown in Figure 3, space efficiency in the EC is basically better than that in the replication. Based on the generally used dual fault tolerance, the space efficiency of the EC (K = 4) is twice that of the replication, and that of the EC (K = 16) is approximately 2.7 times efficient than that of the replication. Based on the quintuple fault tolerance, the space efficiency of the EC (K = 16) is approximately 4.6 times better than that of the replication.

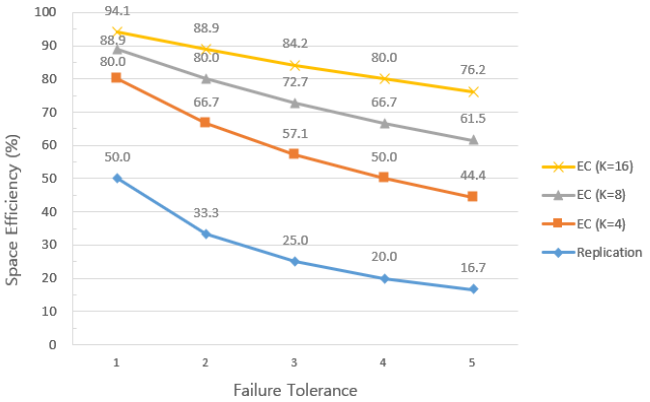

Figure 3. Space efficiency according to fault tolerance

Space efficiency is rapidly degraded as fault tolerance increases in the replication. In the case of the EC, space efficiency depends on the change in K and M values, and the maximum space efficiency is determined by the K value (i.e., the number of original datasets). Thus, it is necessary for users to setup K and M values according to the required space efficiency and reliability.

The EC is absolutely superior in terms of space efficiency compared to the replication. However, the EC has a number of performance degradation factors. This is because the EC is basically based on original data split. In particular, the EC has a large number of chunks that have to be managed in exchange for higher space efficiency by increasing the K value.

3. Fault tolerance and the number of chunks

In the replication, original and replicated chunks exist. However, original data is divided into K datasets and M parities are generated in the EC. Thus, the number of distributed chunks is different between replication and EC. Figure 4 shows the number of chunks required to store files when dual fault tolerance is enforced (M = 2).

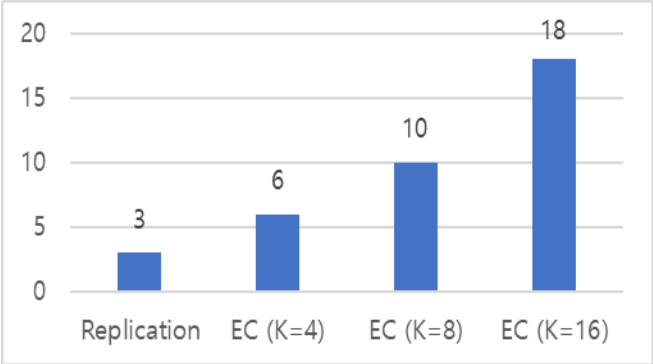

Figure 4. Number of chunks assigned during dual fault tolerance (single chunk set)

As shown in Figure 4, EC (K = 16) has six times more chunks that are managed compared to the replication. That is, when a file is generated, the replication requires three chunks, whereas EC (K = 4) and (K = 16) require six and 18 chunks, respectively. Generally, the larger the number of chunks that are managed in a distributed file system is, the larger the management cost is.

However, since the EC has a large number of chunks, a large volume of data can be stored in a single chunk set. For example, when size of a chunk is 64 MB, the size that can be stored in a single chunk set is 64 MB in the replication, whereas 4 + 2 EC is 256 MB and 16 + 2 EC is

March - April 2020
ISSN: 0193-4120 Page No. 4651 - 4657

1,024 MB. Next, an equation that calculates the number of chunk sets according to a file size is presented.

$$\text{No. of chunk sets} = \left\lceil \frac{file\ size}{chunk\ size * K} \right\rceil, \quad (1)$$

where K refers to the split number of original data. For example, K is one in the case of triple replication, four in the case of 4 + 2 EC, and 16 in the case of 16 + 2 EC. The reason for rounding numbers up is because at least one chunk set has to be assigned, even if a size of data is small.

Figure 5 shows an example of the number of chunk sets required according to a file size when dual fault tolerance is enforced with a chunk size of 100 MB

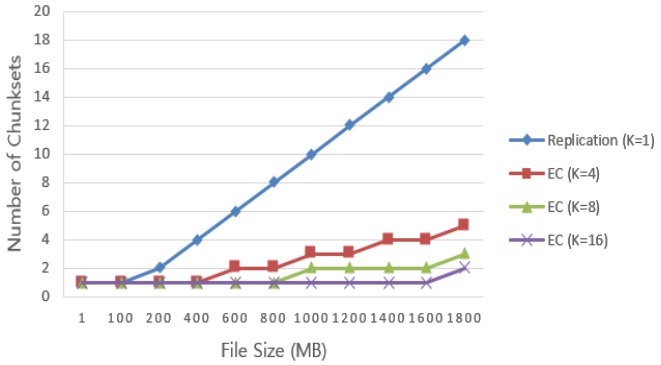

Figure 5. No. of chunk sets according to a file size (M=2)

In Figure 5, the number of chunk sets becomes smaller as the K value increases. In particular, the difference in the number of chunk sets between EC and replication increases as file size increases when size of chunk set is 100 MB. This is because the number of chunks included in a single chunk set increases as the K value increases, so that the data that can be stored in a single chunk set increases. Thus, the actual number of chunks distributed over the DSs is different from the number of chunk sets. Next, an equation that calculates the number of chunks according to a file size is presented.

$$\text{No. of chunks} = \text{No. of chunk sets} * (K + M) = \left\lceil \frac{file\ size}{chunk\ size * K} \right\rceil * (K + M), \quad (2)$$

where K refers to the split number of original data, and M refers to the number of replicated data for fault tolerance. Thus, in the case of triple replication, K is one and M is two. In the case of 4 + 2 EC, K is four and M is two. Figure 6 shows an example of the number of chunks required according to file size when dual fault tolerance is enforced with a chunk size of 100 MB.

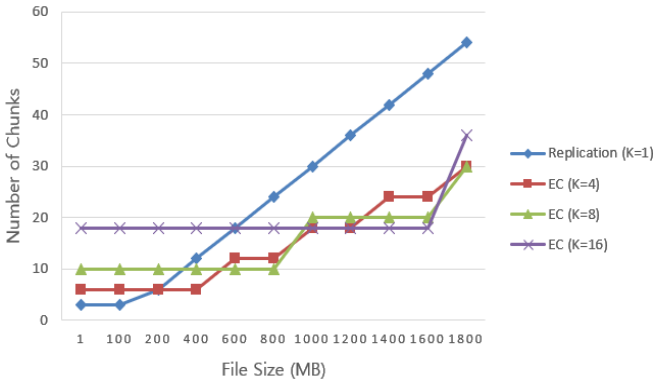

Figure 6. No. of chunks according to file size when a chunk size is 100 MB (M = 2)

As shown in Figure 6, the number of chunks is determined by the K value, M value, chunk size, and file size. If a file size is a single chunk set size, the number of chunks is the least in the replication. However, the required number of chunks increases as file size increases and K becomes smaller; thus, the number of chunks managed in the case of the replication is the largest if a file size is larger than 600 MB. In the case of the EC, the number of chunks that is managed varies depending on the EC setup and file size.

However, more than 99% of files in a storage are less than 1 MB in size [12], and most of files are still less than 1 GB even if average file size has been increased due to the development of the high-performance computing field [13]. In addition, generally, a chunk size is set to a large size to solve the chunk problem in the replication over a distributed file system when there are many large files. That is, most files can be stored in a single chunk set, and the number of chunks that have to be managed increases when the space efficiency is set to high.

March - April 2020
ISSN: 0193-4120 Page No. 4651 - 4657

4. Characteristics of fault recovery of erasure coding and replication

Generally, disk faults occur more frequently as the numbers of servers and disks per server increase in a large-capacity storage system. Furthermore, the number of files requiring recovery will increase upon disk failure as the EC is applied and disk capacity increases [14]. This section discusses the characteristics of fault recovery in detail in consideration of the above circumstances.

When 100 files are stored, both triple replication and 16 + 2 EC methods require 100 chunk sets. However, the number of chunks stored in DSs is 300 in the case of triple replication but 1,800 in the case of 16 + 2 EC. Thus, the number of chunks stored per disk is different, which means that the probability that a corresponding file becomes a recovery target during fault occurrence would be different in the same system. The next equation represents probability that a specific file becomes a recovery target out of the entire files during a fault in a specific disk of the storage system.

$$\text{Probability to of becoming a recovery target} = \left(\frac{K+M}{\text{No.of DSs} * \text{No.of DISKs per DS}}\right) \quad (3)$$

Figure 7 shows the probability of becoming a recovery target during disk faults according to the number of 24 disk-mounted DSs.

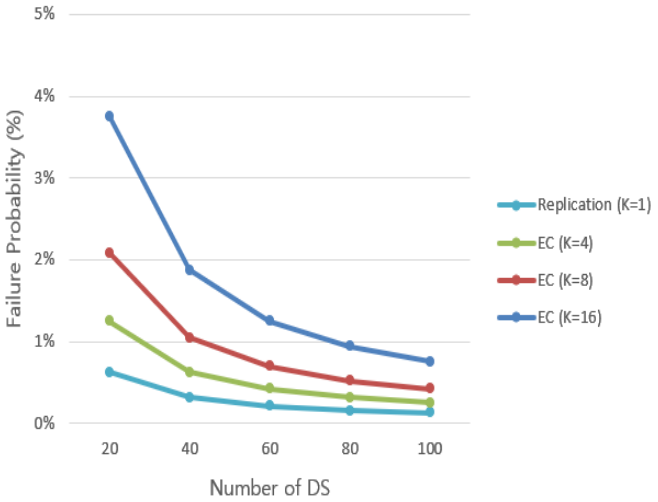

Figure 7. Probability of becoming a recovery target

As shown in Figure 7, probability of a file requiring fault recovery during a single disk failure is up to six times larger in the EC than in the replication. In particular, as the number of DSs becomes smaller, a fault probability increases. When the number of DSs is 20, only 0.6% of the entire files are faults, whereas 1.3% and 3.8% are faults in the case of EC (K = 4) and EC (K = 16) upon a single disk failure.

Figure 8 shows the number of damaged files upon a single disk failure assuming that 100,000 files of a single chunk set size are distributed evenly in 20 DSs, each mounted with 24 disks as in Figure 7.

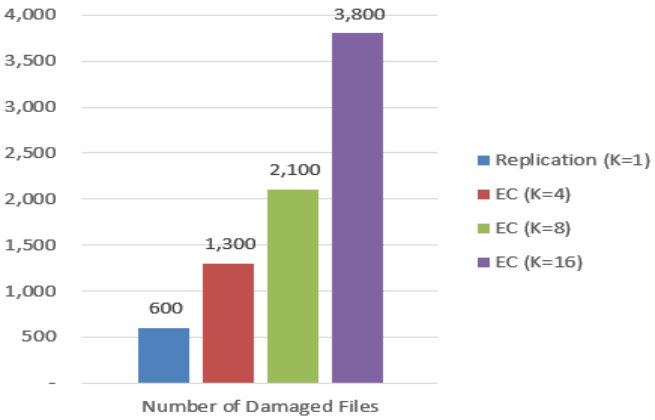

Figure 8. No. of files that require recovery upon single disk failure

As shown in Figure 8, the replication has the least damaged files, while the EC (K = 16) has the largest. In addition, the replication and EC have approximately six times difference in the number of damaged files, but the number of damaged files is not so large compared to the numbers of DSs and entire files. Figure 9 shows the number of chunks that require access for recovery-needed files as shown in Figure 8.

As shown in Figure 9, the replication and EC have significant difference in the number of chunks requiring read access to recover damaged files. In particular, read access is rapidly increased in EC as K value increases. That is, when comparing replication (K = 1) and EC (K = 16), the number of damaged files is six times greater but the number of total chunks that needed to be accessed

March - April 2020
ISSN: 0193-4120 Page No. 4651 - 4657

for recovery is approximately 100 times greater. Thus, the number of input/output (I/O) requests for fault recovery in the EC increases rapidly, which causes various bottlenecks, increasing a performance delay significantly.

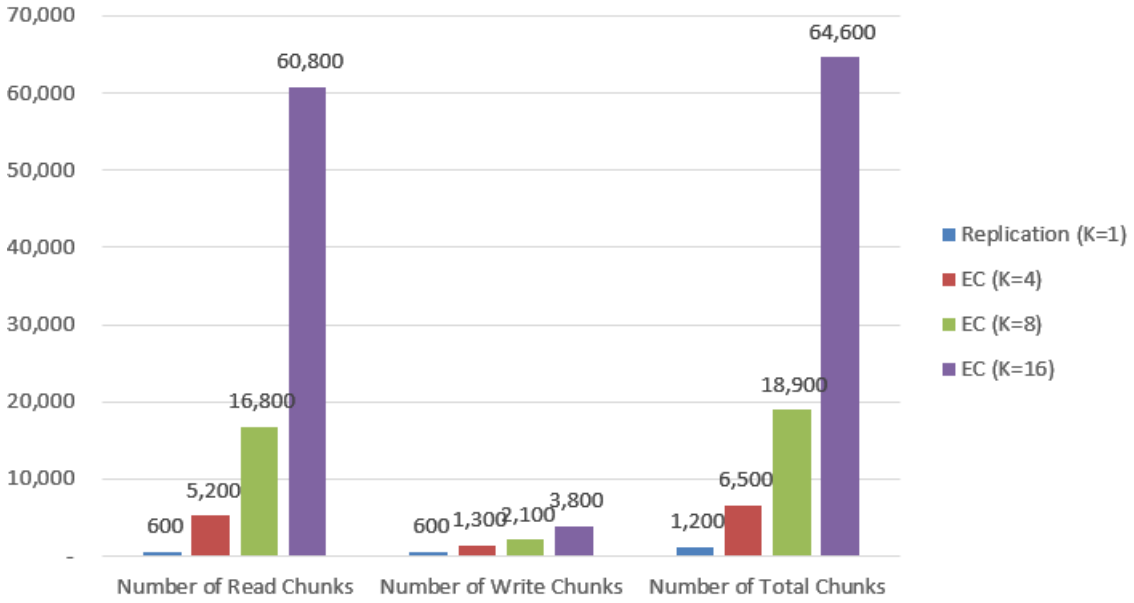

Figure 9. No. of chunks that require access for fault recovery

Table 1: No. of chunks that require access for fault recovery (M = 2)

	1Kbyte	4Kbyte	16Kbyte	64Kbyte	256Kbyte	1024Kbyte
Replication (K=1)	1,024Byte	4,096Byte	16,384Byte	65,536Byte	262,144Byte	1,048,576Byte
EC (K=4)	256Byte	1,024Byte	4,096Byte	16,384Byte	65,536Byte	262,144Byte
EC (K=8)	128Byte	512Byte	2,048Byte	8,192Byte	32,768Byte	131,072Byte
EC (K=16)	64Byte	256Byte	1,024Byte	4,096Byte	16,384Byte	65,536Byte

As presented in Table 1, 4 KB I/O can be possible in the replication when a file size is 4 KB, whereas only 256 byte I/O can be possible in the case of EC (K = 16). Due to this difference, the replication does not require special processing for fault recovery. However, the EC experiences small I/O processing for large-size files during fault recovery so that storage performance can be degraded rapidly if efficient processing is not supported.

5. Conclusion

The replication has low space efficiency because it stores a large volume of replicated data. Thus, it can increase system size as well as construction and management cost exponentially. On the other hand, EC has high space efficiency; thus, system size is not significantly increased accordingly. However, EC requires a process of original data recovery through decoding after reading a large number of original data and parity blocks during fault recovery. In EC, probability of files becoming a fault recovery targets during fault recovery is also increased significantly, although its space efficiency is high. In particular, the number of chunks required during recovery increases rapidly in EC.

6. Acknowledgment

This work was partly supported by Institute for Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.2019-0-00118, Research and Development on Memory-Centric

March - April 2020
ISSN: 0193-4120 Page No. 4651 - 4657

OS Technologies of Unified Data Model for Next-Generation Shared/Hybrid Memory) and IITP grant funded by the Korea government(MSIP) (No.2018-0-00201, Development of High Confidence Information Trading Platform Based on blockchain (PON algorithm))

References

[1] Mirzoev T. Synchronous replication of remote storage. Journal of Communication and Computer 2009;6(3), 34-9.
[2] Abhijith S. The Pros and Cons of Erasure Coding & Replication vs. RAID in Next-Gen Storage Platforms.
[3] Jun L, Baochun L. Erasure coding for cloud storage systems: A survey. Tsinghua Science and Technology 2013;18(3):259-72
[4] Erasure Coding Support inside HDFS. https://issues.apache.org/jira/browse/HDFS-7285/.
[5] Apache Hadoop 3.0.0. http://hadoop.apache.org/docs/r3.0.0-alpha4.
[6] Glusterfs. https://access.redhat.com/products/red-hat-storage/.
[7] Ceph. http://docs.ceph.com/docs/master/rados/.
[8] James SP. Erasure Codes for Storage Systems. The Usenix Magazine 2013;38(6):44-50.
[9] Dongdong S, Yinlong X, Yongkun L, Si W, Chengjin T. Efficient Parity Update for Scaling RAID-like Storage Systems. Proceedings of the 2016 IEEE International Conference on Networking 2016;1-10.
[10] Haddock W, Curry ML, Bangalore PV, Skjellum A. GPU Erasure Coding for Campaign Storage. Proceedings of the Int. Conf. High Performance Computing 2017;145-159.
[11] Mohan LJ, Harold RL, Caneleo PIS, Parampalli U, Harwood A. Benchmarking the Performance of Hadoop Triple Replication and Erasure Coding on A Nation-Wide Distributed Cloud. Proceedings of the 2015 International Symposium on Network Coding (NetCod). 2015;61-5.
[12] Tanenbaum AS, Herder JN, Bos H. File Size Distribution on UNIX Systems-Then and Now. ACM SIGOPS Operating Systems Review. 2006;40(1):100-4.
[13] Welch B, Noer G. Optimizing a hybrid SSD/HDD HPC storage system based on file size distributions. Proceedings of the 2013 IEEE 29th Symposium on Mass Storage Systems and Technologies (MSST). 2013;1-12.
[14] Anton S. Market Views: HDD Shipments Down 20% in Q1 2016, Hit Multi-Year Low. https://www.anandtech.com/show/10315/market-views-hdd-shipments-down-q1-2016/3.