PAPER

Reliability and Failure Impact Analysis of Distributed Storage Systems with Dynamic Refuging

Hiroaki AKUTSU[†a)], **Kazunori UEDA**[††], **Takeru CHIBA**[†], **Tomohiro KAWAGUCHI**[†], ***Nonmembers***, ***and*** **Norio SHIMOZONO**[†], ***Member***

SUMMARY In recent data centers, large-scale storage systems storing big data comprise thousands of large-capacity drives. Our goal is to establish a method for building highly reliable storage systems using more than a thousand low-cost large-capacity drives. Some large-scale storage systems protect data by erasure coding to prevent data loss. As the redundancy level of erasure coding is increased, the probability of data loss will decrease, but the increase in normal data write operation and additional storage for coding will be incurred. We therefore need to achieve high reliability at the lowest possible redundancy level. There are two concerns regarding reliability in large-scale storage systems: (i) as the number of drives increases, systems are more subject to multiple drive failures and (ii) distributing stripes among many drives can speed up the rebuild time but increase the risk of data loss due to multiple drive failures. If data loss occurs by multiple drive failure, it affects many users using a storage system. These concerns were not addressed in prior quantitative reliability studies based on realistic settings. In this work, we analyze the reliability of large-scale storage systems with distributed stripes, focusing on an effective rebuild method which we call Dynamic Refuging. Dynamic Refuging rebuilds failed blocks from those with the lowest redundancy and strategically selects blocks to read for repairing lost data. We modeled the dynamic change of amount of storage at each redundancy level caused by multiple drive failures, and performed reliability analysis with Monte Carlo simulation using realistic drive failure characteristics. We showed a failure impact model and a method for localizing the failure. When stripes with redundancy level 3 were sufficiently distributed and rebuilt by Dynamic Refuging, the proposed technique turned out to scale well, and the probability of data loss decreased by two orders of magnitude for systems with a thousand drives compared to normal RAID. The appropriate setting of a stripe distribution level could localize the failure.

key words: *erasure coding, highly redundant storage systems, reliability, rebuild, Monte Carlo simulation*

1. Introduction

1.1 Background

The amount of digital data worldwide is exponentially increasing; it is expected to reach 40 zeta (10^{21}) bytes in 2020 [1]. In recent data centers, it is not uncommon to have large-scale storage systems storing petabytes of data. In addition, high-density platter technology will increase the capacity of hard disk drives at the rate of 1.4 times per year [2]. Hard disk drives will continue to be used as low-cost media to store a large amount of data.

Manuscript received March 31, 2016.
Manuscript publicized June 17, 2016.
[†]The authors are with Research & Development Group, Hitachi, Ltd., Yokohama-shi, 244–0817 Japan.
[††]The author is with Waseda University, Tokyo, 169–8555 Japan.
a) E-mail: hiroaki.akutsu.cs@hitachi.com
DOI: 10.1587/transinf.2016EDP7139

On the other hand, as the capacity of drives increases, two problems arise regarding data reliability: (i) increase in rebuild time and (ii) increase in the unrecoverable read error rate. For (i), an increase in rebuild time contributes to an increase in the probability of data loss due to multiple drive failures. For (ii), if the amount of data that are read in the rebuild process increases, so does the probability of unrecoverable read errors.

Some recent storage systems use highly redundant techniques, such as erasure coding [3], to prevent data loss. Erasure coding can decrease the probability of data loss by increasing the redundancy level, but normal data write operation will incur a performance penalty known as write penalty. It also requires additional storage capacity when the redundancy level is increased, albeit to a lesser degree than mirroring techniques.

There is a technique to distribute the set of redundant code and data (stripes) to a large number of drives [4]–[6] for load balancing. This makes it possible to exploit parallelism, i.e., to read data simultaneously from many drives and reduce rebuild time compared to other techniques such as traditional RAID. However, there are concerns regarding reliability. Although the rebuild time is reduced, when the number of drives in storage systems is large, say more than a thousand, multiple drive failures may hit a single stripe distributed to a large number of drives.

This paper focuses on highly redundant storage systems that use erasure coding. To achieve high reliability at the lowest possible redundancy level, we introduce an effective rebuild method called Dynamic Refuging. This method (i) rebuilds failed blocks from those with the lowest redundancy level for reliability and (ii) strategically selects blocks to read for repairing lost data. In the analysis of its reliability, we take into account that the redundancy of each block dynamically changes due to multiple drive failure. Quantitative evaluation of the characteristics and techniques described above could not be found in previous reliability studies. We carried out extensive quantitative evaluation through Monte Carlo simulation, modeling the realistic characteristics of drives by using Weibull distribution, and evaluated the effect of distributed stripes and Dynamic Refuging on reliability.

We have already published the initial version of reliability analysis of Dynamic Refuging [19]. The present paper expands these results by showing a failure impact model and the method for localizing the failure.

Distributing stripes to a large number of drives makes it possible to balance I/O not only for rebuilding but also for normal processing. However, if data loss is caused by multiple drive failure, it affects many users who use a storage system. For example, in cloud data centers, stripes with data loss area are distributed to many logical volumes and it will affect many users who use virtual machines. We carried out extensive quantitative evaluation through simulation and modeling of these characteristics and studied how to achieve both I/O load balancing and the localization of the failure.

1.2 Related Work

Mean time to data loss (*MTTDL*) is a reliability index of storage systems that can easily be calculated and has been widely used in the field [7]. However, some authors report discrepancies between the actual rate of data loss and MTTDL [7], [8].

Recently, there have been studies on reliability analysis of storage systems with high redundancy [9], [10] and on the reliability of RAID6 (dual parity) storage systems on the basis of actual failure patterns of field storage systems [11]. Compared with earlier work, MTTDL provides an approximate expression with higher accuracy. However, it was assumed that RAID6 stripes were not distributed among many drives, and higher redundancy beyond level 2 was not considered.

The reliability achieved by distributing stripes to many drives was also studied [12], [13]. The effect of the dynamically changing redundancy of each block due to multiple drive failure was not considered, as well as an appropriate rebuild method for that situation. In addition, the mapping of stripes was calculated using fully static expressions, and it was difficult to add drives dynamically. In our study, we take a hybrid approach in which data stripes are randomly and evenly distributed but their mapping can be calculated using a unique small static mapping table that is shared among nodes.

There are storage systems in which stripes are randomly distributed, called "distributed file systems". For such systems, e.g., HDFS and GFS, data protection methods for distributing stripes among a server cluster have been developed. These systems employ erasure coding to improve storage capacity efficiency compared to replication [3], [14]. However, methods such as Dynamic Refuging have not been implemented, and there had been no quantitative evaluation of reliability.

2. Distributed Storage Systems

2.1 System Overview

We assume distributed storage systems in which data stripes are distributed across many drives of a system. Figure 1 shows an example of a storage system's configuration.

A storage system consists of (i) several processor nodes to handle I/O from host computers, (ii) drives for storing data, and (iii) expander switches to connect processor nodes with multiple drives. The network between the processor nodes is assumed to be a high-speed network, e.g., 10-Gbit Ethernet commonly used in data centers. We also assume that the bandwidth of an expander switch is sufficiently high compared with that of connected drives. Each node provides storage volume to the host computers, encodes data by erasure coding, and distributes stripes among all drives of the system. If drive failure occurs, each processor node of the entire system reads blocks that are necessary to repair the data of the failed drive. We do not consider node failure because it will not immediately cause data loss as long as the drives are normal. However, since node failure affects system availability, we discuss a data placement method to minimize data unavailability.

2.2 Data Striping

Figure 2 shows the stripe data structure in the storage system we assume for this study. Host computers access user data that are divided into *chunks*. A chunk is further divided into a group of *data blocks*. A small box, denoted as *x_yz*, represents an individual block that is either a data block or a *redundancy block*. The letter *x* represents the identifier of the stripe, *y* represents the type of the block (*D:* data block, *C:* redundancy block), and *z* is the order index of the data block or the redundancy block. A chunk is divided into *d* pieces, and the storage system generates *p* redundancy blocks from the data blocks and writes them to the drives. *Erasure coding* is a method for generating redundancy blocks. By using an erasure coding method with *p* redundant blocks and the

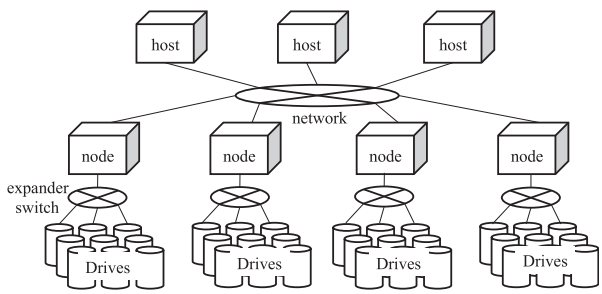

Fig. 1 Storage system overview.

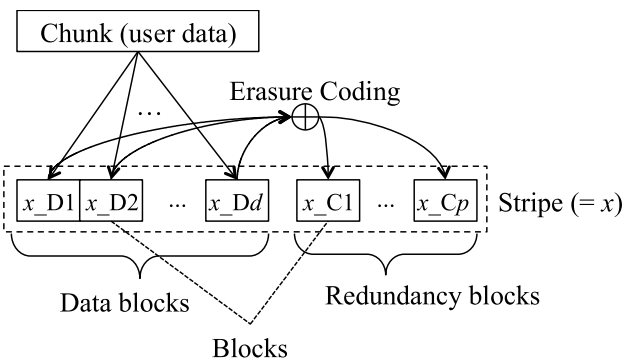

Fig. 2 Stripe data structure in storage system.

maximum distance separable (*MDS*) property, it is possible to recover up to p of any data blocks that were lost.

When p is increased, the performance penalty of storage systems increases. We show two characteristics of performance penalty: (i) cost of calculating redundancy blocks, and (ii) drive access cost.

For (i), erasure coding called Cauchy Reed Solomon (CRS) which has the MDS property is used commonly in many implementations. Its encoding cost (number of XOR operations) is $O(d \times p \times L^2)$ and the decoding cost is $O(d \times k \times L^2)$ [18], where L represents the Galois field parameter of $GF[2^L]$, and k represents the number of redundancy blocks available for repairing data blocks. The condition of L is $(d + p) < 2^L$, so if $d/(d + p)$ is constant and p is increased, L increases and the encoding and decoding cost increases. Even if L is assumed to be constant, the cost of encoding and decoding increases in proportion to p.

For (ii), in a general storage system, the performance is degraded by read-modify-write in case of small-size random write access. In the case of simple replication, p additional write accesses are necessary. In the case of parity-based methods, $2p + 1$ additional accesses ($p + 1$ read accesses and p write accesses) are necessary to modify redundancy blocks by XOR difference.

From (i) and (ii), the increase of the redundancy level leads to the increase of performance penalty. We need to achieve high reliability at the lowest possible redundancy level.

We now explain the stripe distribution method. We distribute blocks in an essentially random fashion because it is generally necessary to dynamically increase or decrease the storage capacity of the set of drives in storage systems. In our study, we took the hybrid approach in which blocks are distributed randomly under the following rules: (i) Data blocks and redundancy blocks of the same stripe are distributed to all different drives. (ii) Each row of a mapping arrangement pattern is cyclically repeated with a fixed period (= c). (iii) The blocks in the same stripe must be placed evenly between the nodes.

Figure 3 shows data mapping with the above rules using a specific example.

A large cylinder in Fig. 3 represents a *logical partition*. A logical partition is a set of stripes (red rectangle in Fig. 3). Storage systems provide data blocks to hosts in the form of *logical volumes* that are sets of data blocks from the same logical partition. A small cylinder in Fig. 3 represents a physical drive. The stripes consisting of data blocks and redundancy blocks are mapped to one or more physical drives.

In Fig. 3, it is assumed that a node uses only one logical partition for simplicity, but a node can use multiple logical partitions.

The number of logical partitions in the storage system is equal to g. In cloud data centers, each node provides logical volumes that belong to logical partitions; and hosts use the logical volumes for running virtual machines.

The upper half of Fig. 3 corresponds to normal RAID

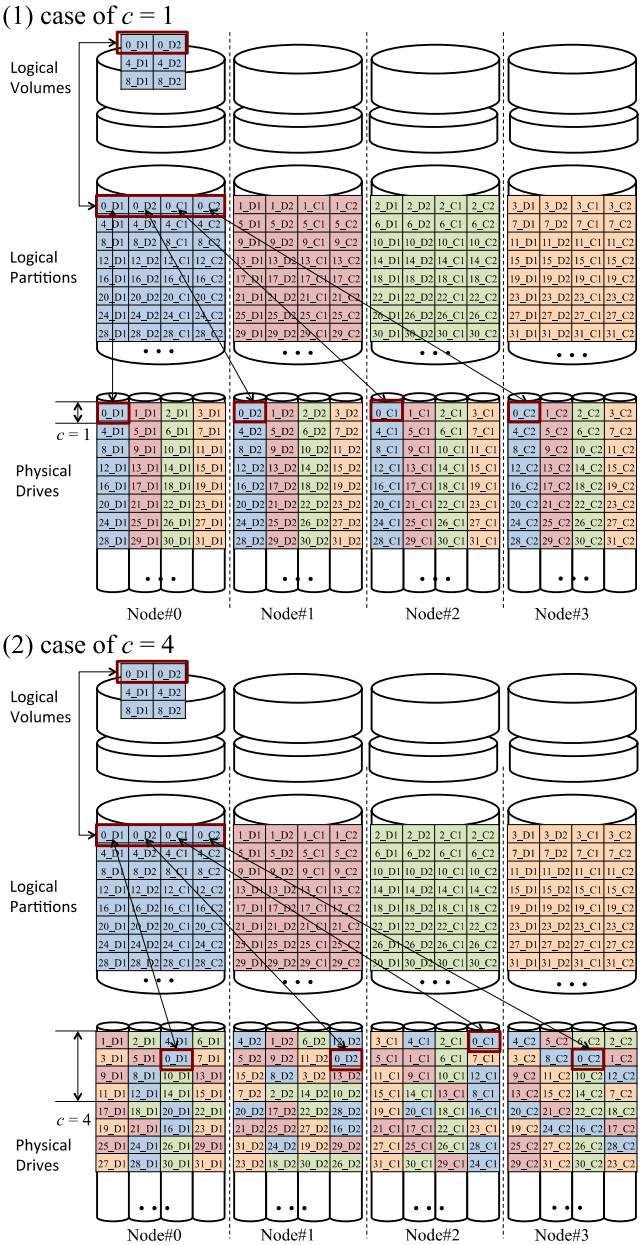

Fig. 3 Data striping with different stripe distribution levels (= c).

with $c = 1$, where the set of stripes whose blocks belong to a certain logical volume are mapped to a small, fixed number of physical drives. In the example shown in the lower half of Fig. 3, with $c = 4$, the same pattern is repeated every four rows. When c is increased, reads for rebuilding are distributed to a larger number of drives, improving the degree of parallelism. Furthermore, by placing the blocks in the same stripe evenly between the nodes, it is possible to perform data access continually during node failure.

In Sect. 4, we evaluate data loss probability at varying stripe distribution levels.

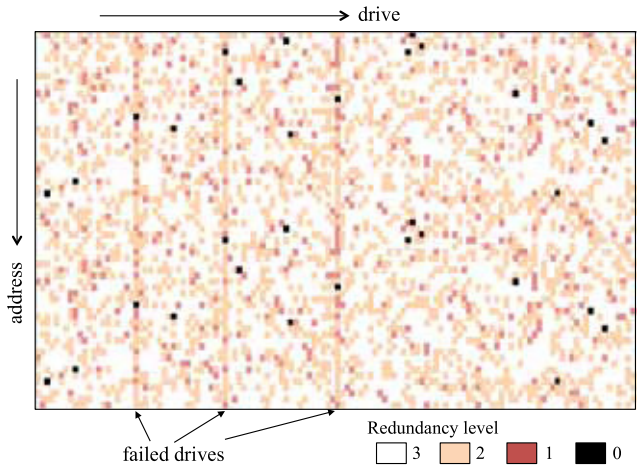

Fig. 4 Area map of each redundancy level (k = 3, p = 3, d = 13, g = 8, c = 32).

Table 1 Parameter definitions.

Name	Definition
p	Number of redundancy blocks in stripe, which represents redundancy level of the entire storage system
d	Number of data blocks in stripe
D	Number of drives in entire storage system
s	Number of blocks in stripe (= $p + d$)
g	Number of parity groups in entire storage system (= D / s)
c	Number of distribution levels, which represents number of rows of random distribution pattern of stripes
r	Current redundancy of entire storage system (minimum redundancy level of all stripes in entire storage system)
k	Number of failed drives
$MTTR_{base}$	Basic rebuild time (Rebuild time of the c = 1 case)
$O_{fail}(t)$	Drive operational failure probability distribution at time t
$O_{failc}(t)$	Cumulative drive operational failure probability at time t
S	Mean time to scrubbing of drive
E	Mean time to unrecoverable error of drive
$F(k, r)$	Number of drives containing stripes of redundancy r when k drives fail
L	Ideal data transfer throughput of rebuilding
C	Capacity of drive
N	Average number of logical partitions which have data loss blocks

2.3 Dynamic Refuging

Dynamic Refuging is an efficient method of rebuilding for highly redundant storage systems in which stripes are distributed among the drives. It combines the following two ideas: (i) rebuild failed blocks from those with the lowest redundancy level and (ii) strategically select blocks to read for repairing lost data.

(i) If stripes are distributed, the redundancy of each area changes dynamically when multiple drive failures occur. Figure 4 shows an example redundancy map after three drive failures, where the configuration is set to p = 3, d = 13, and g = 8. Each of the 128 columns represents a single drive, and each row represents the addressing of data placement of each drive (i.e., Logical Block Address). The color of each point represents the redundancy level (0-3) of each block. Since the number of black areas is small, rebuilding lost data with low redundancy first makes it possible to efficiently maintain the redundancy of the entire storage system.

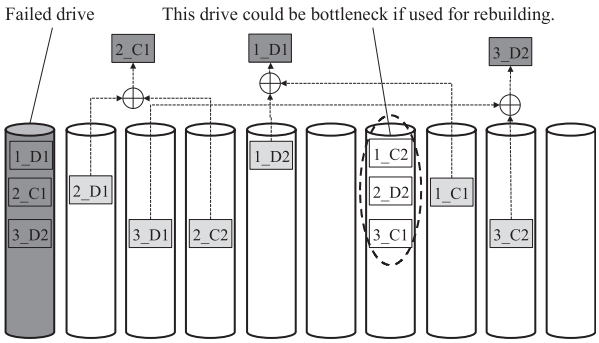

Fig. 5 Strategically selecting stripe data in rebuild process.

Algorithm 1: Stripe data set selection for rebuilding

```
1  for stripeid = 0 to g·c −1 do
2      level = get_stripe_redundancy_level (stripeid)
3      if (level = r) do
4          clear (IgnoreList)
5          for j = 0 to s −1 do
6              drive = get_drive (stripeid, j)
7              for w = 0 to level do
8                  if (size(IgnoreList) < level) then
9                      insert (IgnoreList, drive)
10                     break
11                 else if (PrefetchCount[IgnoreList[w]] <
                                PrefetchCount[drive]) then
12                     IgnoreList[w] = drive
13                     break
14                 end if
15             end for
16         end for
17         for j = 0 to s −1 do
18             drive = get_drive (stripeid, j)
19             if drive ∉ IgnoreList then
20                 insert (RebuildDriveList[stripeid], drive)
21                 PrefetchCount[drive] = PrefetchCount[drive] + 1
22             end if
23         end for
24     end if
25 end for
```

(i) During the rebuild process, the drive that is most heavily accessed will become a bottleneck. Therefore, we select blocks to read for repairing lost data in such a way that the bottleneck (dashed circle in Fig. 5) is avoided and the amount of access will be balanced.

Algorithm 1 describes how to select stripe blocks for rebuilding. For each stripe being restored, *IgnoreList* represents the set of drives not read for recovering the lost data, *PrefetchCount*[i] represents the number of blocks read from drive i, and *RebuildDriveList*[j] is the final output of the algorithm, which represents an array of drives to be read for repairing stripe j.

This algorithm runs when new drive failure occurs and

the redundancy of the entire system (= r) is changed. The algorithm checks each stripe, and if the redundancy level of the stripe is equal to that of the entire storage system, lost blocks are restored. When $level > 0$, the rebuild process can omit reading $level$ blocks, so the drives whose *Prefetch-Count* are among the top r are selected in a greedy approach and are placed in *IgnoreList*. Finally, the set of drives to be read for repairing is given in *RebuildDriveList*.

Next, we analyze the cost of identifying stripes that need to be recovered. In general methods like Consistent hashing, the cost depends on drive capacity because it is necessary to search the whole drives to identify stripes that have the maximum number of lost blocks as in the Dynamic Refuging approach. Recently, the drive capacity is on the order of terabytes, and the number of stripes in a storage system is on the order of megabytes. Thus, reducing the cost of stripe identification becomes a challenge.

Since our algorithm reuses the same pattern cyclically for mapping stripes to physical blocks as described in Fig. 3, so the calculation cost of stripe identification does not depend on the drive capacity. The calculation cost of stripe identification is $O(g \times c \times p \times (p+d))$ that is proportional to the number of drives ($D = g \times (p+d)$) and the distribution level (= c). Actually, it will take only seconds to calculate for each drive failure even assuming thousands of drives and a high level of stripe distribution (e.g. $c = 1024$).

3. Reliability Model

3.1 Drive Failure Model

There are two major types of drive failure:

1. operational failure of the entire drive due to mechanical failure of components such as the seek head, and
2. unavailability of only a part of the drive data.

For the type-1 failure, failure probability is time-dependent (e.g., due to initial failure) [11], [15]. We assume the Weibull distribution and use the parameters given by Elerath and Schindler [11], [16]. For the type-2 failure, we can reduce the chance of data loss by *disk scrubbing* of the storage system. We use the parameters provided by the same authors for the mean time to the occurrence of the unrecoverable read error area and the mean time to detect and recover the error area. Table 2 lists these parameters obtained from [16].

Table 2 Drive parameters.

Operational failure rate	*Rebuild Time*	*Read error rate with Disk Scrubbing*
Weibull ($\eta = 12584\ days$, $\beta = 1.13$)	Constant (20.3 hours)	Uniform (Mean time of disk scrubbing = 186.0 hours, Mean time of read error occurrence = 514.0 days)

3.2 System State Transition Model

Figure 6 shows the state transition diagram of storage systems with Dynamic Refuging. Because drive failure probability related to the individual transition probabilities changes over time, it is not strictly a Markov model. The state of the entire storage system can be defined by two parameters, one is the number of failed drives (= k) and the other is the minimum redundancy level of all stripes in the entire storage system (= r). Each state is denoted as "k, r" in Fig. 6. The state O_{lost} represents the case in which drive failure occurs when the redundancy level of the system is 0, resulting in data loss. The state U_{lost} represents the case in which the redundancy level of the system is 0 and unrecoverable read error occurs in the read data in rebuilding. There is a difference between the damage of data loss of the state O_{lost} and of the state U_{lost}. The state O_{lost} represents the case in which multiple data blocks are lost, while U_{lost} represents the case in which a single data block is lost. The probability of U_{lost} is much higher than that of O_{lost} in realistic conditions as described later.

Each state transition is labeled as follows.

FF: System redundancy level decreases due to operational drive failure
BF: The system redundancy level does not decrease due to operational drive failure
UF: The stripe of redundancy level 0 cannot be restored because of unrecoverable read error in rebuilding
R: The system redundancy level increases by rebuilding

We now define the transition probabilities of *FF*, *BF*, *UF*, and *R*.

The transitions *FF* and *BF* shown in Fig. 6 occur when operational drive failure occurs. If the failed drive contains the stripe that is at the minimal redundancy level (= r), it corresponds to *FF*, otherwise it corresponds to *BF*. Each transition probability is expressed by the following equations:

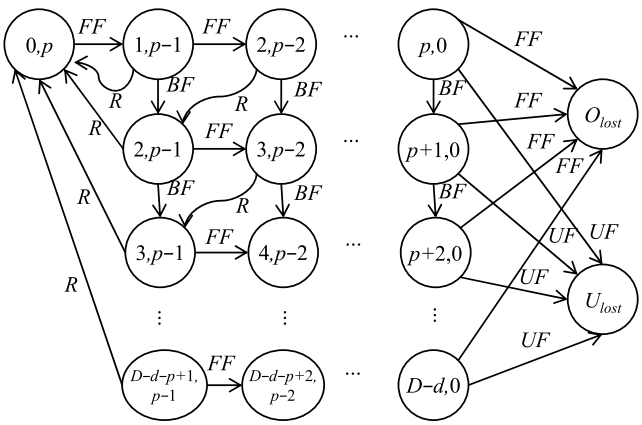

Fig. 6 State transition model of system redundancy with Dynamic Refuging.

$$FF(t,k,r) = O_{fail}(t) \cdot F(k,r) \quad (1)$$

$$BF(t,k,r) = O_{fail}(t) \cdot (D - k - F(k,r)) \quad (2)$$

where $O_{fail}(t)$ is the probability density of operational failure when time t, and $F(k,r)$ represents the number of drives that contain stripes with redundancy level r or less when the number of failed drives is k. These transition probabilities change over time (t) and depend on the degree of distribution (c) and the system scale (g). Although this time dependency is implicit in Fig. 6, our simulation takes it into account, as we discuss in Sect. 4.

In *UF*, strictly speaking, it is possible that multiple read errors overlap in the same stripe. However, since the affected area of error is very small relative to the capacity of the entire drive, the possibility of multiple occurrences of the affected area in a stripe is negligibly small compared with the assumed failure rate of the other failure patterns, and is ignored for this study. Thus, *UF* is assumed to occur only when the storage system has no redundancy ($r = 0$). We define $w(k,r)_i$ to be the number of blocks to read for rebuild in each cycle (of size c) for each drive i when the current redundancy level is r and the number of currently failed drives is k. Then the transition probability of *UF* is represented by the following equation.

$$UF(k,0) = \frac{S}{E} \cdot \sum_{i \in Drives} \frac{w(k,0)_i}{c} \quad (3)$$

Finally, R is determined by the following two factors: (i) the drive with the largest amount of blocks to read becomes a bottleneck and (ii) data transfer throughput in the rebuild process will not exceed the upper limit ($= L$). Then R is expressed by the following equation.

$$R(k,r) = min\left(\begin{array}{c} \dfrac{c}{MTTR_{base} \cdot \max\limits_{i \in Drivers} w(k,r)_i}, \\ \dfrac{L}{C \cdot \sum_{i \in Drives} \dfrac{w(k,r)_i}{c}} \end{array} \right) \quad (4)$$

We evaluate this transition probability in Sect. 4. Strictly speaking, depending on the state transition path, the number of blocks that have been rebuilt may change. It may happen that the redundancy of a stripe once repaired is reduced by another drive failure. Our model and the simulation took this possibility into account and let the parameters $F(k,r)$ *and* $w(k,r)_i$ count stripes at redundancy levels lower than r (as well as those at level r) to simplify the state transition model. Thus our reliability analysis is slightly conservative but this will not have a significant effect on our simulation results.

Refuged data in a rebuild process can be saved in the spare area of a drive which did not fail and in which another block of the same stripe is not already stored. Alternatively, they can be saved in other storage systems. After refuging, data are restored to spare drives (or replaced drives) asynchronously by the hot spare mechanism. The size of the refuge area is considered to be negligible compared with the entire storage, and it does not greatly affect the probability of data loss. Therefore, for simplicity, the refuge area and the sparing process are not reflected in the state transition model of this paper.

3.3 Failure Impact Model

Now we model the number of logical partitions which have data loss blocks ($= N$) when O_{lost} occurs. First, we formulate the number of logical partitions affected by failed drives.

Since $w(k,r)_i$ stripes are randomly mapped to logical partitions, N can be formulated by the same approach as the general stochastic problem like the dice problem [20].

Firstly, let $E(n)$ be the probability that, when we randomly choose logical partitions n times from g logical partitions, the n-th partition is a partition chosen for the first time.

$$E(n) = \left(\frac{g-1}{g}\right)^{n-1} \quad (5)$$

Then, let $X(n)$ be the average number of selected logical partitions when we randomly select logical partitions n times.

$$X(n) = \sum_{i=1}^{n} E(i) = \left(1 - \left(\frac{g-1}{g}\right)^{n}\right) g \quad (6)$$

$N(k)$ represents the average number of logical partitions which have data loss blocks when k drives fail. From $X(n)$, we can calculate $N(k)$ as

$$N(k) = \frac{\sum_{i \in Drives} \left(1 - \left(\frac{g-1}{g}\right)^{w(k,0)_i}\right) g}{D} \quad (7)$$

And we can calculate N as a weighted average of $O_{lost}(k)$ which represents the probability of O_{lost} occurring when $k$drives fail as

$$N = \frac{\sum_{u=p}^{D-d} (N(u) \cdot O_{lost}(u))}{\sum_{u=p}^{D-d} (O_{lost}(u))} \quad (8)$$

3.4 Approximation Closed Formula of Reliability

We derive a simplified, approximated formula of data loss probability and analyze the characteristics of data loss probability with Dynamic Refuging. First, assuming the degree of distribution is sufficiently large, the probability of *BF* transitions can be regarded as nearly equal to zero. As can be seen from the parameters in Table 2, the probability of the state transition is considered to enjoy $R \gg FF$. Thus data loss occurs mainly along the topmost state transition sequence of the diagram shown in Fig. 6. Furthermore, from the assumption that the degree of stripe distribution is sufficiently large, rebuild throughput is approximately g times of the case without stripe distribution, and the rebuild target area is approximately reduced to $1/g^{k-1}$ due to Dynamic Refuging. Therefore, we can approximate the number of

blocks to read as $w(k, p-k)_i \approx c \cdot 1/g \cdot 1/g^{k-1} = c/g^k$. We also approximate the FF transition probability using an approximate hazard function $O_{failc}(t)/t$ (approximated by a straight line for lifetime period and find the slope of the line), which is the approach mentioned by Elerath and Schindler [11]. Figure 7 shows two main types of failure state transitions, in which R_L can be calculated by Eq. (4) as

$$R_L = min\left(\frac{g}{MTTR_{base}}, \frac{L}{C \cdot d}\right) \tag{9}$$

From this model, we can calculate the approximate probability of data loss in time t with the following equations.

$$\begin{aligned} &O_{Lost}(t) \\ &\approx t \cdot \left(\frac{O_{failc}(t) \cdot D}{t}\right) \\ &\quad \cdot \prod_{i=1}^{p} \left(\frac{\frac{O_{failc}(t) \cdot (D-i)}{t}}{R_L \cdot g^{i-1} + \frac{O_{failc}(t) \cdot (D-i)}{t}}\right) \\ &\approx t \cdot \frac{\left(\frac{O_{failc}(t)}{t}\right)^{p+1} \cdot \prod_{i=0}^{p}(g \cdot s - i)}{g^{\frac{p^2-p}{2}} \cdot {R_L}^p} \end{aligned} \tag{10}$$

$$\begin{aligned} &U_{Lost}(t) \\ &\approx t \cdot \left(\frac{O_{failc}(t) \cdot D}{t}\right) \\ &\quad \cdot \prod_{i=1}^{p-1} \left(\frac{\frac{O_{failc}(t) \cdot (D-i)}{t}}{R_L \cdot g^{i-1} + \frac{O_{failc}(t) \cdot (D-i)}{t}}\right) \\ &\quad \cdot d \cdot \frac{S}{E} \cdot \frac{1}{g^{p-1}} \\ &\approx t \cdot \frac{\left(\frac{O_{failc}(t)}{t}\right)^{p} \cdot d \cdot \frac{S}{E} \cdot \prod_{i=0}^{p-1}(g \cdot s - i)}{g^{\frac{p^2-p}{2}} \cdot {R_L}^{p-1}} \end{aligned} \tag{11}$$

$$Lost(t) = O_{Lost}(t) + U_{Lost}(t) \tag{12}$$

Equation (10) represents the probability of data loss due to multiple drive operational failure, resulting in data loss. Equation (11) represents the probability of data loss due to unrecoverable read error occurring in rebuilding. We approximate $D - i$ as D for simplicity in Eqs. (10) and (11). The probability of data loss of the storage systems is given by (12), which is the sum of (10) and (11).

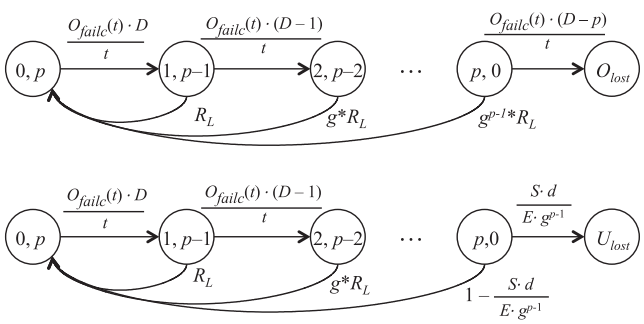

Fig. 7 Main failure state transition of two failure types.

In these equations, parameters that relate to scalability (i.e., the number of drives) are g and R_L. For R_L, it is assumed that the read throughput of rebuild data increases by at most g times but does not exceed L. In Eq. (11), $\prod_{i=0}^{p-1}(g \cdot s - i) \approx g^p$, so Eq. (11) is $O(g^{3p-p^2})$ with respect to g. We can see that when $p \geq 3$, U_{lost} is constant or improves even if the number of drives is increased and network bandwidth is limited. This is the Dynamic Refuging effect. When $p \geq 4$, O_{lost} is constant or improves even if the number of drives increases. Since U_{lost} is two orders of magnitude higher than O_{lost} assuming the parameters in Table 2, the probability of data loss in the case of $p = 3$ can be maintained at a constant level under realistic conditions. We discuss this property in further detail based on realistic simulation in Sect. 4.

Next, we derive an approximation formula of the number of nodes affected by data loss. From the assumption of approximation described in Fig. 7 (topmost state transition sequence of diagram), we consider only the case of $k = p$. The number of blocks to read in a cycle, $w(p, 0)_i$, is larger than 1 because data loss occurs. We can approximate the number of blocks prioritized for rebuild by dynamic refuging by c/g^p, so we can approximate $w(p, 0)_i$ by $\max(1, \frac{c}{g^p})$. From the above, N is represented by the following formula,

$$N \approx \left(1 - \left(1 - \frac{1}{g}\right)^{\left(1, \frac{c}{g^p}\right)}\right) \cdot g. \tag{13}$$

4. Simulation Results

Monte Carlo simulation that simulates the reliability of highly redundant storage systems is very time consuming because many runs are required to obtain adequate accuracy. We implemented a simulator and calculated the reliability under multiple redundancy levels and multiple distribution levels using GNU Scientific Library [17]. For redundancy level 3, we needed to run a 48-core x86-64 cluster server for several weeks to calculate the probability of data loss for five years of operation. We assume eight nodes in the storage system. We also assume that rebuilding does not use I/O bandwidth so much as to affect normal I/O processing. Specifically, we assume $L = 1$ GB/s, which is about 10% (125 MB/s in each node) of the 10-Gbit Ethernet maximum bandwidth commonly used in many data centers.

The simulator runs in two phases. In the first phase, the simulator generates a mapping table according to the rules described in Sect. 2, selects k drives randomly, and calculates $w(k, r)_i$ and $F(k, r)$ by counting blocks whose redundancy levels do not exceed r. The generation of the mapping table and the calculation are repeated multiple times, and we take their average value. In the second phase, the simulator determines the lifetime of the drives based on Weibull distribution and computes state transitions over the 5-year life-

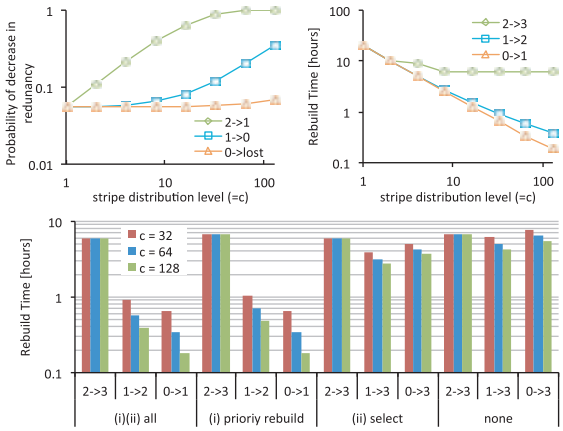

Fig. 8 Simulation results of probability of redundancy decrease and rebuild time ($p = 3$, $d = 21$, $D = 384$).

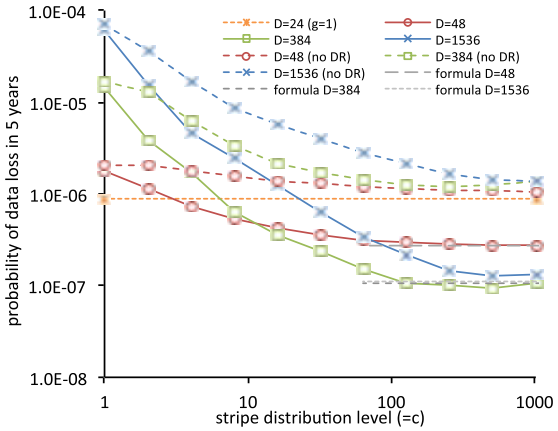

Fig. 9 Simulation results of data loss rate of redundancy level 3 ($p = 3$, $d = 21$).

time period of systems. The two phases are repeated until the probability of data loss lies within a certain confidence interval.

Figure 8 shows the results of simulating state transitions assuming $p = 3$, $d = 21$, $D = 384$. The upper-left graph shows the average reduction of redundancy during drive failure ($= FF/(BF + FF)$). In the graph, $x \rightarrow y$ represents the state transition from redundancy level x to level y when the number of failed drives is $p - x$. Increasing the degree of stripe distribution tended to decrease system redundancy. This is a penalty of stripe distribution. When the current system's redundancy level ($= r$) is low, the probability of decrease in redundancy is low because the number of drives storing the stripes of that redundancy level is small. The upper right graph shows the average redundancy recovery time with Dynamic Refuging. We can see that increasing the degree of distribution of the stripes reduces the rebuild time. The rebuild time under a lower system redundancy level is shorter. The reason for this is that a small amount of stripes with the minimum redundancy level are rebuilt with higher priority. The bottom graph shows the rebuild-time simulation results that compare application or non-application of the two key techniques of Dynamic Refuging: (i) rebuild from the stripes with the lowest redundancy level and (ii) strategically select blocks to read. Without (i), the redundancy level goes directly back to normal, e.g., $0 \rightarrow 3$. With (i), the redundancy level is recovered incrementally, e.g., $0 \rightarrow 1$, $1 \rightarrow 2$, and then $2 \rightarrow 3$. This dramatically reduces time for recovering redundancy from level 0 to nonzero, as shown in the graph. When applying both (i) and (ii), the rebuild time can be reduced further. For example, rebuild time for the $1 \rightarrow 2$ transition is reduced by 28% in case of $c = 128$.

To see the overall properties of the probability of data loss under the two trade-off factors (higher redundancy reduction rate due to drive failure and shorter rebuild time at higher stripe distribution levels), we simulated the probability of data loss. Figure 9 shows the probability of data loss of redundancy level-3 storage systems. The 95% confidence

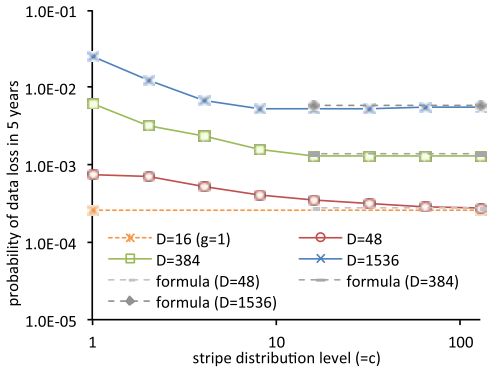

Fig. 10 Simulation results on probability of data loss of multiple drive types of redundancy level 2 ($d = 14$, $p = 2$).

interval of the simulation results was ±10%. We found that by increasing the degree c of stripe distribution, reliability greatly improved. When stripes are distributed (without Dynamic Refuging, dashed lines in Fig. 9), the probability of data loss was about 40 times less than the non-distributed case when $D = 1536$. With Dynamic Refuging (solid lines), the probability of data loss was about 10 times less than without Dynamic Refuging when $D = 1536$. Interestingly, Dynamic Refuging was significantly more effective in larger systems with higher distribution levels. A somewhat surprising result is that, under the identical policy, a system with more drives can attain a lower data loss probability than a smaller system, as shown by three solid-line plots (for $D = 48, 384, 1536$) in Fig. 9.

Next, Fig. 10 shows the probability of data loss with initial redundancy level 2. The 95% confidence interval of the simulation results was ±5%. We compared the simulation results of $c = 1$ (no distribution) with the RAID6 reliability calculator [16] results and those were approximately the same. We used the same capacity ratio ($= 1{:}7$) of redundant blocks and data blocks as the simulation with redundancy level 3. The results show that by increasing the degree of stripe distribution, the probability of data loss decreased by about 5 times when $D = 384$. The "reliability

inversion" phenomenon observed in Fig. 9 was not observed in this redundancy level.

To justify our simulation results, we compared them with the values of Eq. (12) (short horizontal dotted lines in Fig. 9 ($c = 1024$) and Fig. 10 ($c = 128$)). The error was 8% on average and at most 22%. This seems to support the simulation results.

Figure 11 shows the probability of data loss for various numbers of drives. We compared Dynamic Refuging (with stripe distribution) with triple-parity RAID (no stripe distribution). When D is increased, the probability of data loss increases with triple-parity RAID but decreases with Dynamic Refuging because of the reduction of block to be repaired.

Figure 12 shows the number of logical partitions that contain data loss blocks when data loss occurs by multiple drive failures, computed using the approximating formula (Eq. (13)) and by simulation. We can see that the values of the approximation formula and simulation results are almost the same. This also supports the simulation results.

The approximate formula (13) shows that the average number of logical partitions that contain data loss blocks ($= N$) is only 1 when $c \leq g^p$, which means we can localize the failure impact. From Figs. 9 and 10, if the number of drives is larger than hundreds and c is larger than g^{p-1}, data loss probability is close to the theoretical limit value. So, it is recommended that c is set to g^{p-1} to minimize the data loss probability and the effect of data loss if network bandwidth is unlimited. Since our simulation considers the limit of network bandwidth, stripe distribution effect of reducing data loss probability is limited. So it is recommended that c is set to a value smaller than g^{p-1}.

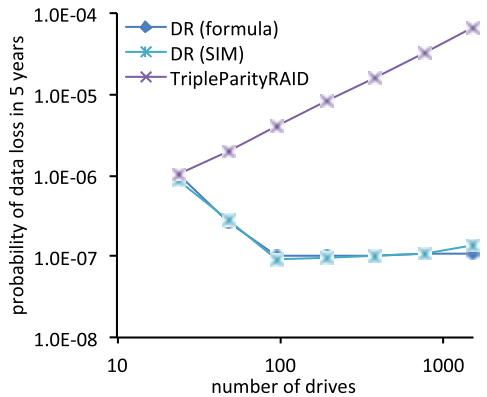

Fig. 11 Comparison of data loss rate with various numbers of drives ($p = 3$, $d = 21$, $c = 1024$).

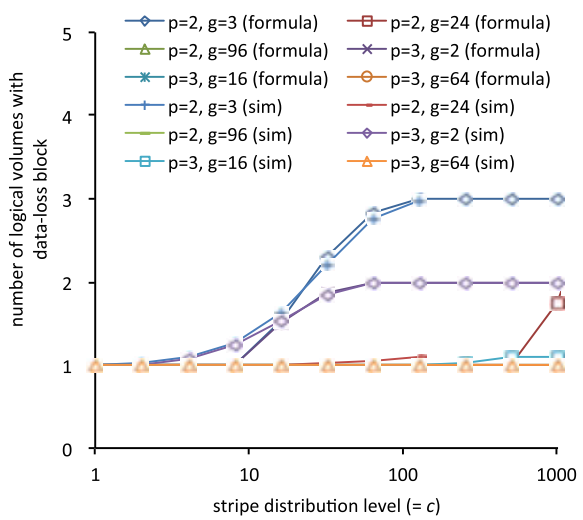

Fig. 12 Number of logical volumes with data loss blocks.

5. Conclusion

For highly redundant storage systems using erasure coding and distributed stripes, we have built a reliability model that reflects the behavior of dynamically changing redundancy of the blocks by multiple drive failure, and introduced an efficient rebuild method called Dynamic Refuging. We evaluated data loss probability and the impact of failure by Monte Carlo simulation with a realistic drive failure characteristics. We also derived an approximate closed formula of data loss possibility and failure impact. The findings we obtained are summarized as follows:

i. For redundancy level 2, we have quantitatively shown that distributing stripes over many drives lowers data loss probability of the whole system compared to the non-distributed case (as in normal RAID ($c = 1$)). The result was obtained by simulation under realistic settings which assume a modest use of network bandwidth for rebuilding and take into account the dynamic change of drive failure rates and of the number of blocks at individual redundancy levels.
ii. For redundancy level 3, we have quantitatively shown that aggressive stripe distribution combined with Dynamic Refuging allows us to scale the system up to more than a thousand drives in the sense that it maintains or even lowers data loss probability in this scale range. This was confirmed both by simulation under realistic settings and approximate closed formulas. Similar phenomena were not observed with redundancy level 2, suggesting the importance of careful analysis of different individual cases.
iii. For systems with a thousand drives, it is recommended that c is set to g^{p-1} to minimize the data loss probability and the effect of data loss, or to a value smaller than g^{p-1} when we consider network bandwidth.

In summary, even with more than a thousand drives and assuming limited network bandwidth in data centers, a high level of stripe distribution and Dynamic Refuging will jointly make highly redundant storage systems more reliable and viable.

References

[1] J. Gantz and D. Reinsel, "The Digital Universe in 2020: Big

Data, Bigger Digital Shadows, and Biggest Growth in the Far East," http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf, 2012.
[2] R. Freitas, J. Slember, W. Swadon, and L. Chiu, "GPFS Scans 10 Billion Files in 43 Minutes," http://www.almaden.ibm.com/storagesystems/resources/GPFS-Violin-white-paper.pdf, 2011.
[3] K. Peter, "Reliability study of coding scheme for wide-area distributed storage systems," 19th Euromicro International Conference on Parallel, Distributed and network-based Processing, pp.19–23, 2011.
[4] K. Hwand, H. Jin, and R. Ho, "RAID-x: A New Distributed Disk Array for I/O-Centric Cluster Computing," 9th High-Performance Distributed Computing, pp.279–286, 2000.
[5] Q. Xin, E.L. Miller, T.J.E. Schwarz, S.J., "Evaluation of Distributed Recovery in Large-Scale Storage Systems," 13th IEEE International Symposium on High Performance Distributed Computing, pp.172–181, 2004.
[6] K. Hwang, H. Jin, R. Ho, and W. Ro, "Reliable Cluster Computing with a New Checkpointing RAID-x Architecture," 9th Heterogeneous Computing Workshop, pp.171–184, 2000.
[7] J.G. Elerath and M. Pecht, "A Highly Accurate Method for Assessing Reliability of Redundant Arrays of Inexpensive Disks (RAID)," IEEE Trans. Comput., vol.58, no.3, pp.289–299, 2009.
[8] K.M. Greenan, J.S. Plank, and J.J. Wylie, "Mean time to meaningless: MTTDL, Markov models, and storage system reliability," 2nd USENIX Workshop on Hot Topics in Storage and File Systems, pp.1–5, 2010.
[9] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures," IEEE Trans. Comput., vol.44, no.2, pp.192–202, 1995.
[10] A. Leventhal, "Triple-Parity RAID and Beyond," ACM Queue, vol.7, no.11, 2009.
[11] J.G. Elerath and J. Schindler, "Beyond MTTDL: A Closed-Form RAID 6 Reliability Equation," ACM Trans. Storage, vol.10, no.2, p.7, 2014.
[12] Y. Gao, D. Meister, and A. Brinkmann, "Reliability Analysis of Declustered-Parity RAID6 with Disk Scrubbing and Considering Irrecoverable Read Errors," Fifth IEEE International Conference on Networking, Architecture, and Storage, pp.126–134, 2010.
[13] X. Wu, J. Li, and H. Kameda, "Reliability Analysis of Disk Array Organizations by Considering Uncorrectable Bit Errors," IEICE Trans. Inf. & Syst., vol.E81-D, no.1, pp.73–80, 1998.
[14] G. Gibson, B. Fan, W. Tantisiriroj, and L. Xiao, "DiskReduce v2.0 for HDFS," https://opencirrus.org/system/files/4%20Gibson-OpenCirrus-Jan28-10.pdf, 2010.
[15] B. Schroeder and G.A. Gibson, "Disk Failures in the real world: What does an MTTF of 1,000,000 hours mean to you?," 5th USENIX Conference on File and Storage Technologies, no.1, 2007.
[16] RAID 6 Equation Calculator, http://raideqn.netapp.com
[17] GSL—Gnu Scientific Library, http://www.gnu.org/software/gsl/
[18] J. Blomer, "XOR-Based Erasure-Resilient Coding Scheme," Technical Report TR-95-048, International Computer Science Institute, Berkeley, 1995.
[19] H. Akutsu, K. Ueda, T. Chiba, T. Kawaguchi, and N. Shimozono, "Reliability Analysis of Highly Redundant Distributed Storage Systems with Dynamic Refuging," 23rd Euromicro International Conference on Parallel, Distributed and Network-based Processing, pp.261–268, 2015.
[20] M. Conroy, "A Collection of Dice Problems," http://www.madandmoonly.com/doctormatt/mathematics/dice1.pdf

Hiroaki Akutsu received the B.S. and M.S. degrees in Information Sciences from Waseda University in 2003 and 2005, respectively. He now with Hitachi, Ltd. Research & Development Group, Storage Research Dept. working as a Researcher.

Kazunori Ueda received his M. Eng. and Dr. Eng. degrees from the University of Tokyo in 1980 and 1986, respectively. He joined NEC in 1983, and from 1985 to 1992, he was with the Institute for New Generation Computer Technology (ICOT) on loan. He joined Waseda University in 1993 and has been Professor since 1997. He is also Visiting Professor of Egypt-Japan University of Science and Technology sinde 2010. His research interests include design and implementation of programming languages, concurrency and parallelism, high-performance verification, and hybrid systems.

Takeru Chiba received the B.S. and M.S. degrees in Information Sciences from School of Engineering, TOHOKU University in 2007 and 2009, respectively. He now with Hitachi, Ltd. Research & Development Group, Storage Research Dept. working as a Researcher.

Tomohiro Kawaguchi received the B.E. defress in Physical Science and Engineering in 2001 and M.E. degrees in Computational Science and Engineering in 2003 from Nagoya University. In 2003, he joined System Develeopmet Laboratory, Hitachi, Ltd., Kanagawa, Japan. Since 2003, he has been engaged in the reseach of storage functionalities and solutions.

Norio Shimozono received the Ph.D. degrees in information science and technology from Osaka University in 2015, respectively. He now with Hitachi, Ltd. Research & Development Group, Storage Research Dept. working as a Senior Researcher.