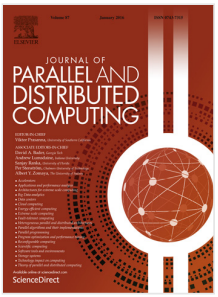
Accepted Manuscript



A new reliability model in replication-based big data storage systems

Jun Wang, Huafeng Wu, Ruijun Wang

PII: S0743-7315(17)30049-7
DOI: http://dx.doi.org/10.1016/j.jpdc.2017.02.001
Reference: YJPDC 3631

To appear in: *J. Parallel Distrib. Comput.*

Received date: 15 June 2015
Revised date: 30 January 2017
Accepted date: 1 February 2017

Please cite this article as: J. Wang, H. Wu, R. Wang, A new reliability model in replication-based big data storage systems, *J. Parallel Distrib. Comput.* (2017), http://dx.doi.org/10.1016/j.jpdc.2017.02.001

A new reliability model in replication-based Big Data storage systems

Jun Wang[1 2], Huafeng Wu[2 *], Ruijun Wang[1]

Abstract

Reliability is a critical metric in the design and development of replication-based big data storage systems such as Hadoop File System (HDFS). In the system with thousands of machines and storage devices, even in-frequent failures become likely. In Google File System, the annual disk failure rate is 2.88%,which means you were expected to see 8,760 disk failures in a year. Unfortunately, given an increasing number of node failures, how often a cluster starts losing data when being scaled out is not well investigated. Moreover, there is no systemic method that can be used to quantify the reliability for multi-way replication based data placement methods, which has been widely used in enterprise large-scale storage systems to improve the I/O parallelism.

In this paper, we develop a new reliability model by incorporating the probability of replica loss to investigate the system reliability of multi-way declustering data layouts and analyze their potential parallel recovery possibilities. Our comprehensive simulation results on Matlab and SHARPE show that the shifted declustering data layout outperforms the random declustering layout in a multi-way replication scale-out architecture, in terms of data loss probability and system reliability by up to 63% and 85% respectively. Our study on both 5-year and 10-year system reliability equipped with various recovery bandwidth settings shows that, the shifted declustering layout surpasses the two baseline approaches in both cases by consuming up to 79 % and 87% less recovery band-

[1]University of Central Florida, Orlando, Florida
[2]Shanghai Maritime University
* Corresponding Author

width for copyset, as well as 4.8% and 10.2% less recovery bandwidth for random layout.

Keywords: Reliability, Random declustering, copyset, Multi-way replication, Continuous Time Markov Chains (CTMC)

1. Introduction

Today, an increasing number of big data storage systems are turning to use multi-way replication based storage architecture for the sake of high availability and reliability [18, 16, 3, 10].

Reliability is increasingly important in big data storage systems built from thousands of individual components. As the storage system scale up, it increases both capacity and bandwidth, but it also makes disk failures more common [35]. In petabyte-scale file systems, disk failures will happen in a daily bases. In the system with thousands of machines and storage devices, even in-frequent failures become likely. Therefore, analyzing the system reliability through an analytic method is useful since they can provide a reference for developers to choose the best layout catering their requirements.

With regards to the modeling of system reliability, it should considers all the situations that will make the system unreliable. Intuitively, when people think about system reliability, they will tend to consider the data loss probability and the system recovery bandwidth [5, 6, 37]. However, in a replication-based storage systems, the loss of replicas will also have an influence on the system reliability which should not be neglected. This motivates us to create an effective and comprehensive analytic model to obtain an accurate reliability results.

Many replication schemes are proposed to improve the reliability without taking the impacts on performance into account. For example, some schemes were designed to improve the reliability by concentrating the replicas in a small subset. These plans may give rise to performance degradation because the redundancy is not only used to improve the reliability but also used to serve the normal read to enhance the throughput. Increasing the scatter width can alle-

viate this problem by spreading the replicas evenly in a larger subset. Thus, it is important to improve reliability while minimizing the impacts on the performance.

It may be noted that it is not feasible to build a real testbed to evaluate the reliability of a large-scale redundancy storage system with different layout schemes. Once the block distribution scheme is selected, the layout can no longer be changed after the production storage system is up to work. Moreover, there is no systemic method which can be used to quantify the reliability for multi-way replication based data placement methods. This is especially true when the number of copies exceeding two. This is because the replicas of data on one failed disk are distributed among multiple disks in the declustered storage system, if either one of the disks fails before the failed disk completely recovered, the data block will lost permanently. As a result, the data loss probability has increased. This motivates us to exploit the impact of data layouts on reliability in multi-way replication storage systems.

In this paper, we propose a comprehensive reliability model to investigate the system reliability of multi-way declustering data layouts and analyzing their potential parallel recovery possibilities.

This paper makes the following contributions:

1. Propose a new reliability model to investigate the system reliability of multi-way declustering data layouts.
2. Utilize the reliability model for analyzing the data loss probability and system repair rate with different data layout schemes;
3. Quantify the most important parameter used in the model, $P^{(l)}$, which is the probability that the disk does not lose data with l failed disks in the system with either mathematical proof for copyset replication layout [1] or reasonable sampling techniques for shifted declustering [36] and random declustering[3];

[3]Random declustering layout distributes data blocks according to given randomization algorithms, which map the key (or the index) of a data block to a position in the storage system.

4. Analyze the possible optimal parallel recovery schemes for copyset replication, shifted declustering and random declustering layouts;
5. Make specific comparison between these three layouts, which are the most widely used in current enterprise large-scale multi-way replication storage systems;
6. Simulate the reliability model, compare and explain the system reliability with considering various recovery bandwidths.

In the remainder of the paper, we use n for the number of disks in a system, and k for the number of ways of replication. This paper is organized as follows: Section 2 introduces prior research related to the reliability study of large-scale multi-way replication storage systems. Section 3 gives the continuous time Markov chains (CTMC) model for system reliability, and quantifies the parameters used in the model. In Section 4, we present, quantify and compare the key parameters for the three different data layouts. In Section 5, we simulate the reliability using the Matlab software [25] and the SHARPE package [27] and explain the results. The conclusion is discussed in Section 6.

2. Related Works

2.1. Copyset replication

Copysets [1] replication technique is proposed to reduce the data loss frequency by introducing a near optimal trading off between number of nodes on which the data is scattered and the possibility of data loss. The problem is defined by several parameters R,N,S, which stand for number of replicas, number of nodes and the scatter width respectively. There are two phases of the copyset application technique, which are permutation and replication. The first phase refers to an offline activity that creates a number of permutations by randomly permuting the nodes into the system. The second phase executes when a chunk needs to be replicated. The number of permutation depends on scatter width S, which is equal to $S/(R-1)$. The primary replica can be placed on any node of the system, while others are placed on the nodes of a randomly chosen copyset

that contains the first node. This design provides an optimal tradeoff between the scatter width and the number of copysets. Comparing with random replication method, copysets is able to reduce the probability of data loss form 99.99% to 0.15% under power outage in a 5000-node RAMCloud cluster.

2.2. Random and shifted declustering approach

Recently, a general multi-way replication-based declustering scheme has been widely used in enterprise large-scale storage systems to improve the I/O parallelism. Specifically, a random replication method [3] is widely used in large-scale storage systems due to its simple replication technique and strong protection against uncorrelated failures. Unfortunately, its high data loss rate and poorly handling of common correlated failures make the random replication method lose its generality in large-scale storage systems. There is also another copyset replication method [1] that has been proposed to obtain lower data loss frequency. Although the copyset replication technique is able to greatly reduce the probability of data loss, however it trades-off the data loss frequency with the amount of lost data in each incident. Thus reliability issue remains a main concern. To improve the system reliability, a placement-ideal data layout–Shifted declustering [36] has been created to obtain optimal parallelism in wide variety of configurations and load balancing in both fault-free and degraded modes.

2.3. Muti-way Replication

RAID is first introduced in the mid 80's for improving both the performance and reliability of storage systems by providing redundancy through replication or parity schemes. RAID-1 applies mirroring, the simplest policy for replication based redundancy. Declustered layout schemes for replication based disk arrays include chained declustering [23], group-rotational declustering [14], and interleaved declustering [15]. Among them, chained declustering can be generalized to multi-way replication, but to the best of our knowledge, no such implementation exists. Group-rotational declustering is used in media streaming servers configured with 3-way replication to improve the quality of Video On Demand

services [13]. Interleaved declustering is difficult to leverage up to multi-way replication. shifted declustering is a scalable solution for multi-way replication storage over any number of disks achieving optimal parallel performance [36].

Due to the availability benefits and the ease of maintenance, the switch from parity to multi-way replication is being further encouraged by the wide adoption by such mission critical systems as Google's File System (GFS) [18], Amazon's Dynamo [16] used to power Amazon.com and Amazon's S3 service, Microsoft's FARSITE [10], projects using Apache's Hadoop File System (HDFS) [3], video on demand services [13], and Geographic Information Systems (GIS) [30].

There are also theoretical methods for replica distribution in a large data cluster. RUSH [22] was introduced for the replica placement issues in large distributed storage systems. RUSH is a family of algorithms that has excellent randomization performance and low conflict probabilities, so it can be used in distributed storage systems which allowed each node to distribute data objects. Tosun compared a series replicated declustering schemes for spatial data systems such as GIS, where majority queries obey some explicit patterns [30]. For example, range queries cover a geometric shape of a spatial data set. However, decentralized systems have not been used in reality. Current enterprise systems such as the Google File System [18], are still centrally controlled due to the ease of maintenance.

2.4. Existing Reliability Models

Xin *et al.* [34, 35] analyzed the reliability in terms of MTTDL (mean time-to-data-loss) of large-scale distributed storage systems, configured with two-way and three-way replication. They apply Markov chains, where the state represents the failed number of data objects in a single redundancy group[4]. This model is based on the assumption that the data objects are independent,

[4]For replication based redundancy, a *redundancy group* includes all copies of a data unit. For parity based redundancy, a redundancy group includes all data stripe units as well as parity units of a data stripe.

as well as redundancy groups. With this assumption, it makes sense to only model one redundancy group is enough, and if at time t, the reliability of one redundancy group is $R(t)$, then the reliability of the whole system is simply $1-(1-R(t))^r$, where r is the number of redundancy groups in the system. In this model, single data object failure should be allowed. However, they take disk failure rate as the data object failure rate. With this failure rate, all data objects on a disk fail when the disk fails. Accordingly the failure of data objects are no longer independent, and it is contradictory to the assumption. As long as the events of disk failure exist, it is not reasonable to view the redundancy groups as independent. We will also explain that disk failure is the dominant factor rather than data block failure to evaluate the impacts of data layouts on system reliability in Section 4. Gafsi *et al.* applied continuous time Markov chains (CTMC) to model the reliability of distributed video servers [17]. Their analysis is based on parity redundancy and two-way replication redundancy. They categorized the redundancy schemes into one-to-one, one-to-all and one-to-some, where one-to-all and one-to-some are declustering layout policies.

Currently, there is no systematic research on the reliability issues for redundancy systems with multi-way replication. Among reliability metrics, the most straightforward one to understand is the failure free period, during which the system provides services without any data loss.

3. Extended Reliability Model

The uncorrelated failures, such as failures of individual servers or disks, happen thousands of times a year. For example, In Google File System, the annual disk failure rate is 2.88%, which means you were expected to see 8,760 disk failures in a year. Concurrent failures, such as failures of servers in a rack or larger domain, happens dozens of a year. Large-scale correlated failures, such as the entire cluster loses power, happens once or twice every year. Our model is evaluated based on the real-life enterprise observations. We also compare the applicability of these data layout schemes. While the random declutter-

ing and copy set replication can be used both for block-level and object-based storage system, the shifted declustering cannot be applied to the object-based storage system. Continuous time Markov chain (CTMC) has been utilized for the modeling of multi-way replication declucstered storage system [37]. In this section, we will further extend the model by abstracting the CTMC model to an ordinary differentiate equation (ODE) group.

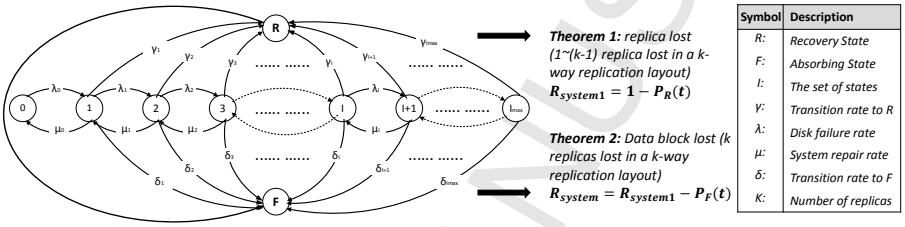


Figure 1: State transition in declustered layouts

In order to precisely analyze the reliablity of the storage system, we made the following assumptions:

- The state space I is the set of states with a certain number of failed disks.
- A disk failure means that the data of the failed disk is unreachable and unrecoverable in place permanently. Other types of failures such as problems in cables, network, air conditioning and power are not considered.
- All the disks are homogeneous with the same failure rate.
- Storage systems with different layouts store the same amount of data per disk.
- Recovery procedure starts right after disk failure.
- The recovery is at block level, and the switch overhead (seeking and rotating time) between normal service and recovery is not considered.
- The storage network bandwidth is not a bottleneck in parallel recovery.

- The workload per disk in the storage system follows an even distribution. The storage system is not saturated at any time point.

With these assumptions, CTMC is the desired model to describe the development of a storage system. Define $p_i(t)$ as the probability that the system is in the state i at time t. The graphical representation of a CTMC was presented in Figure 1 and can be abstracted to an ordinary differentiate equation (ODE) group as shown in equation 1. With regards to the system reliability, there are two situations we should take into consideration. Take 3-way replication method as an example. The first situation happens when one or two replica of the data blocks have been lost due to node failures, disk failures or system unrecoverable errors. In this case, the data lost instance has not happened but the system reliability was degraded because of only one or two copies of the data block remaining functionally. The second situation happens when the data lost instance was happened, which means 3 replica of the data blocks have been lost. In this case, the data blocks will lost permanently, which should be avoided in the storage system design. We will analysis and formulate the above two cases in details in Sections 3.1 and 3.2.

$$\frac{d}{dt}\mathbf{P} = Q\mathbf{P} \tag{1}$$

where Q is the generator matrix [26], $\mathbf{P}$ is defined $\mathbf{P} = [p_0(t), p_1(t), \ldots, p_R(t), p_F(t)]'$ as the vector of $p_i(t)$'s, and $\frac{d}{dt}\mathbf{P}$ is defined $\frac{d}{dt}\mathbf{P} = [\frac{d}{dt}p_0(t), \frac{d}{dt}p_1(t), \ldots, \frac{d}{dt}p_R(t), \frac{d}{dt}p_F(t)]'$ as the vector of the first derivative of $p_i(t)$'s for all i.

The element of Q on the i-th ($0 \leq i \leq l_{max} + 1$) row and the j-th ($0 \leq j \leq l_{max} + 1$) column (represented by q_{ij}) is the transition rate from the state i to the state j. The elements of the generator matrix of the CTMC in are:

$$q_{ij} = \begin{cases} \lambda_i, \text{ if } 0 \le i \le l_{max-1} \text{ and } j = i+1 \\ \mu_i, \text{ if } 1 \le i \le l_{max} \text{ and } j = i-1 \\ \delta_i, \text{ if } 1 \le i \le l_{max} \text{ and } j = F \\ \gamma_i, \text{ if } 1 \le m \le k-1,\ 0 \le i \le l_{max-1} \text{ and } j = R \\ -\sum_{i \ne j} q_{ij}, \text{ if } i = j \end{cases} \tag{2}$$

Equation Group (1) can be expanded in terms of Equation Group (3) using q_{ij}.

$$\begin{cases} \frac{dp_0(t)}{dt} = -\lambda_0 p_0(t) + \mu_0 p_1(t) \\ \frac{dp_i(t)}{dt} = -(\mu_{i-1} + \lambda_i + \delta_i) p_i(t) + \lambda_{i-1} p_{i-1}(t) + \mu_i p_{i+1}(t), \\ \qquad 1 \le i \le l_{max} \\ \frac{dp_R(t)}{dt} = \sum_{i=0}^{l_{max-1}} \sum_{j=1}^{k-1} \gamma_i p_{b_j}(t) \\ \frac{dp_F(t)}{dt} = \sum_{j=1}^{l_{max}} \delta_j p_j(t) \end{cases} \tag{3}$$

The functions $p_i(t)$'s ($0 \le i \le F$) solved from the ODEs (3) with the initial condition: $p_0(0) = 1$, $p_i(0) = 0$ ($\forall i > 0$) is the probability that the system is in state i at time t. The initial condition means at the initial time, the probability of the system without disks failure (at state 0) is 100%. $p_{b_j}(t)$ represents the probability of replica lost p_b in state j at time t.

3.1. Case 1:

In this section, we will analyze and formulate the relationship between system reliability and the probability of replica lost. The state R represents the recovery state, which means 1 to $k-1$ replicas have been lost in a k-way replication approach. The system reliability has been degraded and is defined in equation (4):

$$R_{system1}(t) = 1 - p_R(t) \tag{4}$$

R represents the recovery state, the function $p_R(t)$ is defined as the system unreliability under recovery mode. We will discuss the aggressive parallel recovery in Section 4.4. As we discussed in the above paragraph, $p_R(t)$ is closely related with the probability of replica lost p_b. Now, we further investigated and computed the probability of replica lost defined in equation (5).

$$p_b(m) = \begin{cases} \dfrac{m}{C(N,1) \cdot C(k,m)}, \\ \text{m: } 1 \leq m \leq k-1, \end{cases} \tag{5}$$

where m stands for the number of replicas, k is defined as the total number of replicas for each data block, and N represents the total number of data blocks per disk. Based on the principle of permutation, the equation (5) can be further expanded and shown in the following equation (6).

$$p_b(m) = \begin{cases} \dfrac{m \cdot m!}{N \cdot k \cdot (k-1) \cdot \ldots \cdot (k-m+1)}, \\ \text{m: } 1 \leq m \leq k-1, \end{cases} \tag{6}$$

Apparently, $p_b = 0$ if $m = 0$. If the replica lost after one more disk failure, the system transits from the state l to state R, and the transition rate γ_l is defined as:

$$\gamma_l = (n-l) \cdot \lambda \cdot (1-p_b) \tag{7}$$

Where n represents the total number of disks in the storage system. The transition rate γ_l is related to the number of failed disks, the disk failure rate and the probability of replica lost. In the next section, we will investigate the other situation that will affect the system reliability mentioned above.

3.2. Case 2:

Since the state F represents absorbing state, the function $p_F(t)$ is usually defined as the system unreliability [31]. Meanwhile, the reliability of the system is defined in equation (8):

$$R_{system}(t) = R_{system1}(t) - p_F(t) \tag{8}$$

The mean time to failure (MTTF) of the system described by an absorbing CTMC is defined in Equation (9) [31]. In this particular model for a storage system, F is the state of losing data, and the MTTF of the system is actually the mean time to data loss (MTTDL).

$$MTTDL = \int_0^\infty R_{system}(t)dt \tag{9}$$

In our model, λ_l and δ_l are functions of $P^{(l)}$, which is defined as **the probability that the system does not lose data after the l-th disk failure**. Apparently, $P^{(l)} = 1$ if $l < k$, and $P^{(l)} = 0$ if $l > l_{max}$. With l failed disks, there are $n-l$ disks which may fail, so the rate of another disk failure is $(n-l)\lambda$, where λ represents the disk failure rate. If there is no data loss upon l disk failures, and no data loss after one more disk failure, the system transits from the state l to $l+1$, and the transition rate is $(n-l)\lambda P^{(l+1)}$. On the other hand, if there is data loss after one more disk failure, the system transits from the state l to the state F, and the transition rate is $(n-l)\lambda(1-P^{(l+1)})$. Accordingly,

$$\lambda_l = (n-l)\cdot\lambda\cdot P^{(l+1)} \tag{10}$$

$$\delta_l = (n-l)\cdot\lambda\cdot(1-P^{(l+1)}) \tag{11}$$

The value of $P^{(l)}$ and the repair rate μ_l when the system has l failed disks are also different from layout to layout. In the following section, we will investigate the above two values in our shifted declustering layout as well as random declustering layout in the later sections.

4. Reliability Analysis

4.1. Shifted Declustering Layout

Shifted declustering [36] is a layout scheme that distributes the replicas within one redundancy group with a certain distance. The distance increases in each iteration by one. Depending on the number of disks n, shifted declustering has two solutions:

1. Basic solution

The basic solution is applied to the situations when n is odd or $n = 4$ for $k = 3$, and n is prime for $k \geq 4$.

For k-way replication ($k \geq 3$), the k replicas within one redundancy group are distributed to k disks, and from the first disk in these k disks onwards, each disk has a fixed distance after the previous one.

For example, assuming that the disks are labeled from 0 to $n-1$, if $k = 3$, and one redundancy group is distributed to disks d_1, d_2 and d_3, then $d_2 = (d_1 + y) \bmod n$, and $d_3 = (d_2 + y) \bmod n$, where y is the distance between the replicas of this redundancy group. The layout is illustrated in Figure 2 and explained in [36].

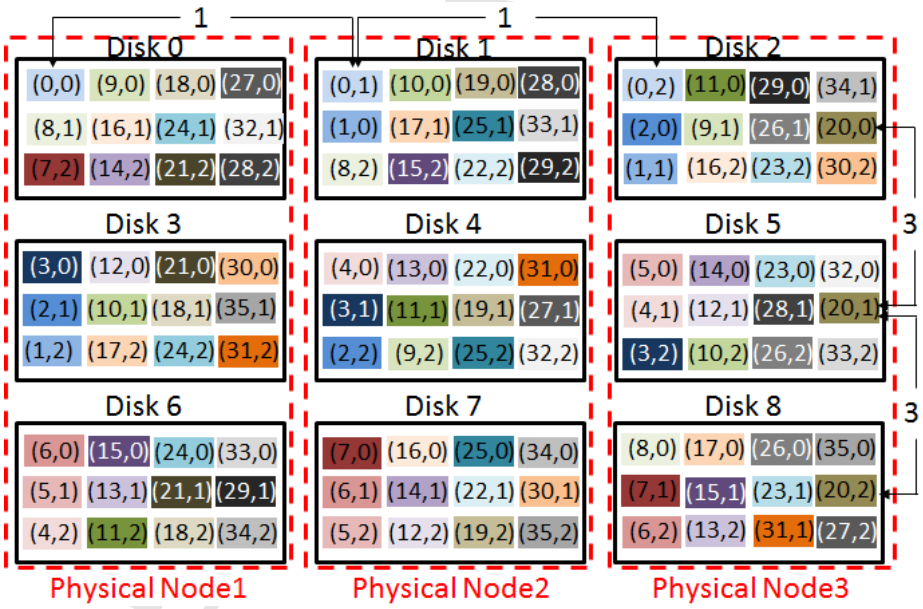


Figure 2: Shifted declustering layout

Due to the layout feature, data will be lost upon l disk failures if there exists a set of k failed disks with a fixed distance between neighboring disks. Otherwise, there is no data loss. **The value of $P^{(l)}$ (no data loss after l disk failures) is the probability that given l failed**

disks, for any k out of the l disks, there is no fixed distance between neighboring ones. Therefore, we are looking for the sub-suite $\{i_0,\ i_1,\ \ldots,\ i_{l-1}\} \subseteq [0,\ \ldots,\ n-1]$ such that $\forall\{j_0,\ j_1,\ \ldots,\ j_{k-1}\} \subseteq \{i_0,\ i_1,\ \ldots,\ i_{l-1}\}$, where $\{j_0,\ j_1,\ \ldots,\ j_{k-1}\}$ is a combination of k elements out of the sub-suite $\{i_0,\ i_1,\ \ldots,\ i_{l-1}\}$, and d does not exist for all $m \in \{0,\ 1,\ \ldots,\ k-1\}$, $j_{(m+1)} \bmod {}_k = (j_m + d) \bmod n$.

Denote $S_{sd}(n,k,l)$ as the number of choices to select l disks from n disks with k-way replication configuration, obeying the conditions shown above. Therefore,

$$P^{(l)} = \frac{S_{sd}(n,k,l)}{C(n,l)} \tag{12}$$

In particular, if $l = k$, $P^{(k)} = 1 - n(n-1)/2C(n,k)$, because if first failed disk is chosen, there are $(n-1)/2$ types of distances to decide the other $k-1$ failed disks to cause data loss. There are n ways to select the first disk, so there are in total $n(n-1)/2$ ways to select k failed disks to lose data.

2. Extended solution

The extended solutions are applicable to the cases when n is even and $n > 4$ for $k = 3$, and n is non-prime for $k \geq 4$.

For the extended, solution replaces n in the numerator in Equation 12 with n', where n' is the maximum number that is smaller than n and there exists a basic solution for n'-disk, k-way replication configuration. If $k = 3$, $n > 4$ and n is even, $n' = n - 1$, because $n - 1$ is odd and there is basic shifted declustering layout for $(k, n-1)$ configuration. If $k > 3$, n is non-prime, then n' is the largest prime number smaller than n.

Now the value of $P^{(l)}$ is

$$P^{(l)} = \frac{S_{sd}(n',k,l)}{C(n,l)} \tag{13}$$

Similar to the basic solution, if $l = k$, $P^{(k)} = 1 - n(n'-1)/2C(n,k)$. Because if first failed disk is chosen, there are $(n'-1)/2$ types of distances to decide the other $k-1$ failed disks to cause data loss.

When $l = k$, the value of $P^{(k)}$ can be explicitly given:

$$P^{(k)} = 1 - n(n' - 1)/2C(n, k) \tag{14}$$

where $n' = n$ for the basic solution; $n-1$, for the extended solution if $k = 3$; n' is the maximum prime number less than n for the extended solution if $k > 3$.

In shifted declustering layout, the equations for $P^{(l)}$ are difficult to summarize. This is because the combination behavior is evaluated by "fixed distances", which varies from 1 to $\lfloor n'/2 \rfloor$. So it is not easy to distinguish independent subsets for the purpose of generalizing the behavior. For this reason, we have difficulty in abstracting the expression of $P^{(l)}$ explicitly.

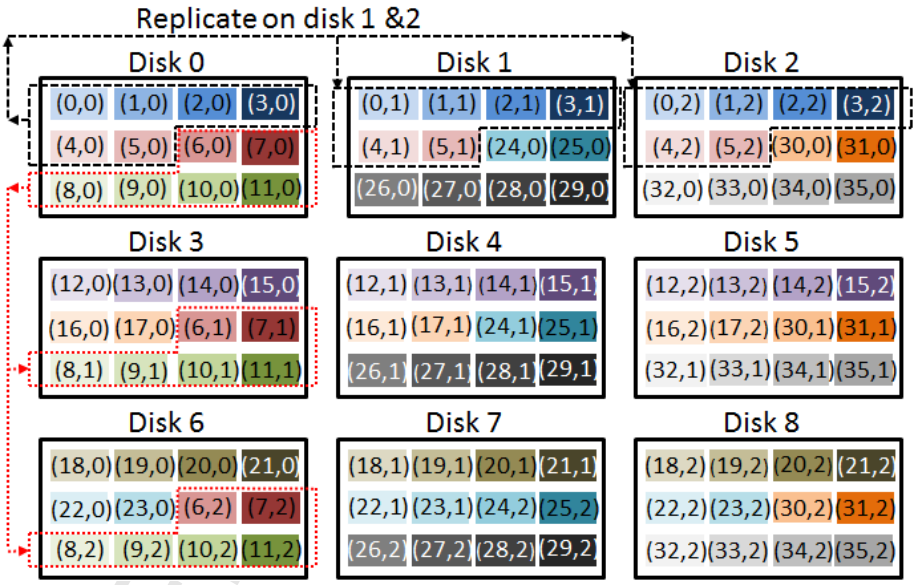


Figure 3: Copyset Replication layout

4.2. Copyset Replication Layout

Copyset replication is a novel technique that provides a near optimal solution between scatter width and the number of copsets [1]. Figure 3 is an example of copyset replication technique with a scatter width equals 4 and the number of replicas for each chunk is 3. In this example, there are 12 chunks

in each disk and 3 replicas for each chunck. Take chunk (0,0)in disk 0 as an example, the other two replicas (0,1) and (0,2) are either in disk 1 or in disk 2. Figure 3 illustrates that the chunks of disk 0 is either replicated on disk 1 and disk 2 or replicated on disk 3 and disk 6. In this case disk {0,1,2} and disk {0,3,6} are formed as copysets. Based on this regulation, the total combination of copysets of these nine disks are {0,1,2}, {0,3,6}, {1,4,7}, {3,4,5}, {6,7,8} and {2,5,8}. Accordingly, the system will only lose data if and only if it loses a whole copyset. However, copyset replication method trades off the probability of data loss with the amount of lost data in each incident. In other word, the storage system may not lose data frequently as comparing with shifted or random declustering layout but may lose a larger amount of data if node failure happens.

Therefore, if three nodes fail at the same time, the probability of data loss in this specific example is:

$$\frac{number of copysets}{total number of copysets} = \frac{6}{84} = 0.07 \tag{15}$$

Explicitly, if k-way replication has been used in this copyset technique, each copyset group must contain at least k nodes to ensure that all replicas of the primary chunks will be included. As we mentioned before, the data loss will only happen when a whole copyset has been lost. Thus, if l disks fail and l is an integer multiples of k,the probability of no-data loss after the l-th disk failure is:

$$P^{(l)} = 1 - \frac{2\frac{n}{k}}{C(n,k)}\frac{l}{k} \tag{16}$$

Otherwise, the probability of no-data loss after l disk fails will be:

$$P^{(l)} = 1 - \frac{2\frac{n}{k}}{C(n,k)}\lfloor\frac{l}{k}\rfloor \tag{17}$$

4.3. Random Declustering Layout

Random declustering layout distributes data blocks according to given randomization algorithms, which map the key (or the index) of a data block to

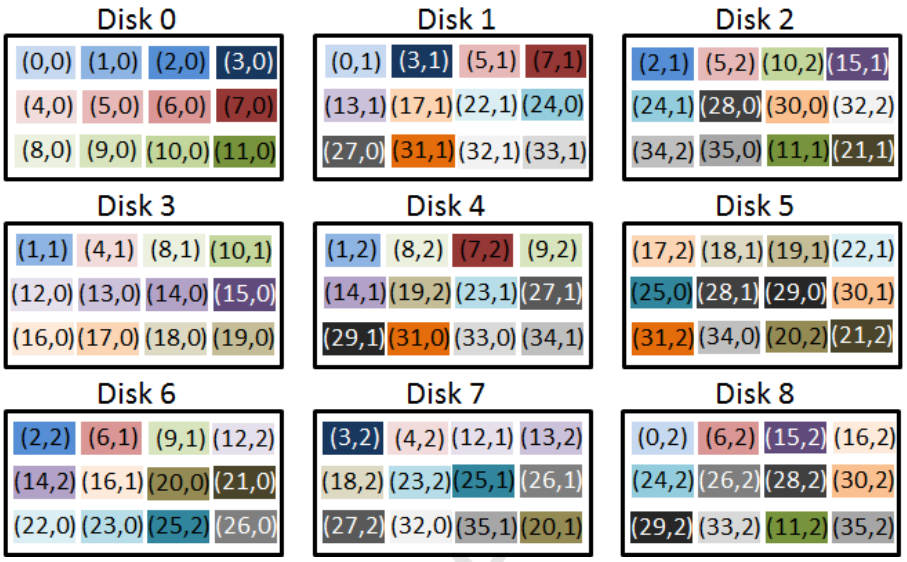


Figure 4: Random Declustering layout

a position in the storage system. Assume that each data block has a unique key, and the blocks within a redundancy group have different keys. Ideally, a good randomization algorithm distributes data in a balanced manner. If a disk fails, all other disks should share the same amount of traffic redirected from the failed disk. Figure 4 illustrates an example of random declustering layout.

The probability of losing (or not losing) data upon l failed disks depends not only on l, the number of disks (n), and the number of replicas in each redundancy group (k), but also on the number of redundancy groups (r). The number of redundancy groups is a function of n, k, the used capacity of disks (S), and the size of a data unit (s). We assume that the used space of each disk is the same, and the size of a data unit is fixed. The number of redundancy groups can be derived as $r = \frac{S \times n}{s \times k}$, where $S \times n$ means the total used space for data storage, and $s \times k$ is the space used by one redundancy group.

There are $C(n, k)$ ways to distribute a redundancy group. We assume that an ideal random declustering algorithm has the ability to distribute redundancy groups to at most combinations of disks possible. With this assumption, if

$r < C(n,k)$, the redundancy groups are distributed to r combinations of k disks. If $r \geq C(n,k)$, all $C(n,k)$ combinations are used for distributing redundancy groups. Upon l disk failure, the probability of losing data is the probability of that the k failed disks is one combination of the r combinations used for distributing redundancy groups. When $l = k$, the probability of not losing data is:

$$P^{(k)} = \max(1 - r/C(n,k), 0) \tag{18}$$

For $l > k$, due to the lack of mathematical regulations to describe the behavior of random declustering, we use sampling techniques to obtain the values of $P^{(l)}$.

4.4. Aggrestive parallel recovery and the repair rate μ_l

Recovery is commonly performed in two different ways: off-line and on-line [20]. In large-scale storage systems, on-line recovery is more acceptable, because the system will not pause service while the recovery process is undergoing. In this paper, we assume that on-line recovery is used and each disk will dedicate a certain amount of bandwidth for recovery if the disk is involved in a recovery process. This assumption is reasonable, because such bandwidth allocation schemes are widely used in storage quality of service (QoS), and can cooperate with other mechanisms like latency control, burst handling, etc. to guarantee a minimum compromise of storage performance[19]. If recovery is considered as an event that requires a priority in a storage system, similar policies can be used to provide a certain amount of sources for this process.

For simplicity, we assume a fixed bandwidth per disk in recovery (*defined as recovery bandwidth per disk*), optimizing the storage QoS is out of our research emphasis of this work. Since in current disk drive architectures there is only one read/write channel, at any time a surviving disk is either under normal service or under recovery. The concurrency of normal service and recovery at a course grained time scale can be obtained in a time sharing manner at a fine grained time scale. For example, we are able to get a 10 KB/sec recovery bandwidth

usage by controlling a disk to transfer 10 KB recovery data per second, and to serve normal requests in the remaining time.

Let $\mu = b_r/S$, where S is the capacity of a disk used for storage, and b_r is the recovery bandwidth per disk, so μ is the rate of sequentially recovering one failed disk. We assume an aggressive recovery scheme: for a failed disk, its data replicas can be found from multiple other disks, and as many as possible disks sharing redundant data with the failed disk will be involved in the recovery process. When a disk fails, a new disk is added, but the aggressive recovery does not recover the failed disk's data to the new disk sequentially. Instead, the system will take advantage of spare space on all surviving disks and the new disk to recover the data as fast as possible. First the data is reconstructed and written to the available spare space, then the recovered data is moved to the new disk in the background to restore the system to the original layout before failure. As long as the data of the failed disks is restored somewhere in the system, the failed disks are considered as recovered.

With the above assumption, the recovery rate upon l-disk failures (μ_l) is proportional to the number of disks which can provide data for recovery. We define the disks providing data for recovery as *source* disks, and the disks which the recovered data is written to as *target* disks. At any moment, a disk can only be a source disk or a target disk. If the number of source disks surpasses half of the total number of disks, we consider $\lfloor n/2 \rfloor$ as the number of source disks, because at most $\lfloor n/2 \rfloor$ source-target disk pairs can be formed. Upon l disk failures, we assume l blank disks are added, when $l < n/2$, the possible maximum source disks is still$\lfloor n/2 \rfloor$; when $l \geq \lceil n/2 \rceil$, the possible maximum source disks is $n - l$, which is the number of all surviving disks. Accordingly, we can get an upper bound of recovery rate: $\min(\lfloor n/2 \rfloor \mu, (n-l)\mu)$.

However, the upper bound recovery rate may not be achievable in non-ideal layouts, because the replicas of data on the failed disk may be limited to a small number of surviving disks. As a result, the transfer rate can be expressed as the product of a factor x and the value μ, where x is the smaller one among $\lfloor n/2 \rfloor$, $n - l$ and the number of source disks. In addition, with different failed

disk combinations, the number of source disks differs. With this assumption, we can conclude the ranges of μ_l for each layout:

In shifted declustering and random declustering layouts, all surviving disks can provide data for recovery, so the recovery rate is always $\min(\lfloor n/2 \rfloor \mu, (n - l)\mu)$.

For random declustering, we make the assumption that all disks across all nodes have an equal probability of being selected to become a replica holder. This is a slight deviation from the Google File System [18] (and Hadoop's HDFS [3]) which incorporates rack position awareness and thus limits the nodes and drives that could potentially be selected by the random placement algorithm. This assumption simplifies the analysis of the system while still providing a model that could be used within the locality group specified in these real world systems.

For copyset replication,$\mu_l \leq \min((k-1)l\mu, \lfloor n/2 \rfloor \mu, (n-l)\mu)$, where n must be a multiple of k.

The highest recovery rate is achieved if the l failed disks are in as many redundancy disk groups as possible. This includes the following situations:

- If $l \leq n/k$, the highest recovery rate is achieved if the l failed disks are in l different redundancy disk groups, because for each failed disk, $k-1$ surviving disks within its redundancy disk group are its source disks. In this case, the recovery rate is $(k-1)l\mu$. In addition, if $(k-1)l\mu > \lfloor (n-l)/2 \rfloor \mu$, the highest recovery rate is $\lfloor (n-l)/2 \rfloor \mu$, since it is the upper bound as analyzed above.

- If $n/k < l \leq n(k-1)/k$, the highest recovery rate is achieved if the l failed disks are in all n/k redundancy disk groups, so all surviving disks can be source disks. Therefore, in this case, the recovery rate reaches the upper bound, which is $\min(\lfloor n/2 \rfloor \mu, (n-l)\mu)$.

As a result, the highest recovery rate is $\min((k-1)l\mu$, $\lfloor n/2 \rfloor \mu, (n-l)\mu)$, so $\mu_l \leq \min((k-1)l\mu, \lfloor n/2 \rfloor \mu, (n-l)\mu)$.

In contrast, the lowest recovery rate is achieved if l failed disks are in as few redundancy disk groups as possible. This happens when the l failed disks are limited within $\lceil l/(k-1) \rceil$ redundancy disk groups. In each of $\lfloor l/(k-1) \rfloor$ redundancy disk groups, $k-1$ disks fail, and the failed disks can be recovered by only one surviving disk. In the remaining redundancy disk group (if $\lceil l/(k-1) \rceil \neq \lfloor l/(k-1) \rfloor$), $l \bmod (k-1)$ disks fail, and they can be recovered by $k - (l \bmod (k-1))$ disks.

4.5. Comparison between Copyset,Shifted and Random Declustering

Some current enterprise-scale storage systems adopt random data layout schemes to distribute data units among storage nodes [18, 3]. There is also theoretical research on algorithms for random data distribution in large-scale storage systems [22]. Random declustering attracts people's interests, because it brings a statistically balanced distribution of data, thus providing optimal parallelism and load balancing in both normal and degraded mode[5].

Compared to random declustering, Shifted declustering layout deterministically guarantees the optimal parallelism and load balancing for replication based storage systems. For a large data set, both random and Shifted Declustering deliver comparably high performance due to optimal parallelism, but they may provide different reliability. In Shifted Declustering layout, $P^{(k)}$ (probability of not losing data when the system has k failed disks) is independent from the number of redundancy groups. It is given as Equation (14) in Section 4.1. We also find that $P^{(k)}$ in random declustering layout is closely related to the number of redundancy groups (r) in the system, mentioned in Equation (18) in Section 4.3.

Copyset replication is able to achieve the best probability of no data loss comparing with random and Shifted Declustering layout due to its carefully

[5]If there are disk failures in the system, but the data set is still complete, the service work load which is supposed to be processed by the failed disks is directed to surviving disks. This system status is called degraded mode.

replication of the data blocks which largely reduce the number of redundancy groups to minimize the probability of data loss.

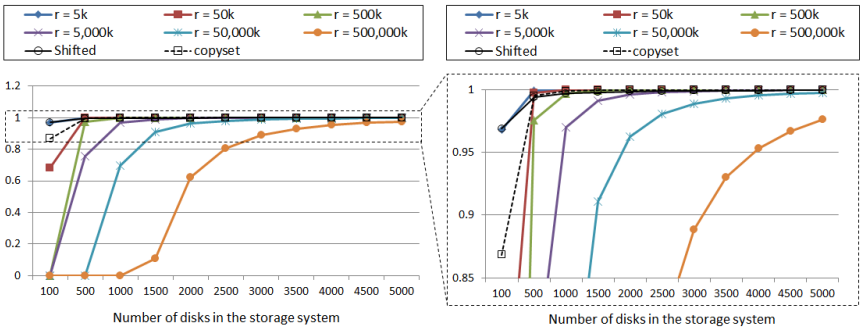


Figure 5: The comparison of $P^{(k)}$ between copyset,shifted and random declustering

From Equations (14) and (18), we can see that as long as n and k are determined, $P^{(k)}$ of Shifted Declustering is fixed. While $P^{(k)}$ of random declustering is negatively linear with r. This indicates that the more the redundancy groups in random declustered storage systems, the higher the probability of data loss upon k disk failures. Additionally, in Section 4.4, we show that these two layouts have the same potential for parallel recovery, so the reliability can be directly reflected by the value of $P^{(k)}$.

In Figure 6, we quantitatively compare the change of $P^{(l)}$ with the number of disks (n) and the number of redundancy groups (r). We assume $k = 3$, the number of disks varies from 100 to 5,000, and the number of redundancy groups varies from 5,000 to 5×10^9. The left diagram demonstrates the dependence of $P^{(k)}$ on n and r in random declustering. Each line reflects the value change of $P^{(k)}$ for a certain r and a varying n. We can see that with a fixed number of disks, the more redundancy groups the system has, the smaller is $P^{(k)}$, thus the easier the system lose data upon k disk failures. We enlarge the portion near $P^{(k)} = 1$ to the right diagram, and add the $P^{(k)}$ values of both shifted declustering and copyset. When the number of disks in the storage system is larger than 100, we can see that $P^{(k)}$ of shifted declustering and copyset are constantly approaching 1 with the increase in the number of disks, so the larger

the system scale, the lower the probability that the system will lose data upon k disk failures. However, when the storage system only has 99 disks, the $P^{(k)}$ of copyset is worse than shifted declustering because it is sensitive to the number of disks as well as the scatter width.

There is a break-even point of r (denoted by r^*) that random and shifted declustering have the same value of $P^{(k)}$. It can be found by making Equations (14) and (18) equal to each other, and solving the following equation:

$$1 - r^*/C(n,k) = 1 - n(n'-1)/2C(n,k) \tag{19}$$

Equation 19 yields $r^* = n(n'-1)/2$. For a given number of disks (n), if $r > r^*$, shifted declustering has higher reliability, and vise versa.

Considering a storage system with 1,000 disks, configured as in the Google File System: each data chunk has three copies, and each copy is 64 MB. The break-even number of the redundancy groups $r^* = 499,000$, this only allow the system to store at most about 96 TB (96 GB per disk) data when using the random declustering layout, otherwise the system is less reliable than with shifted declustering layout. Notice that 96 GB per disk is a small amount of capacity in today's hard drives. For larger storage system with more disks, the data size per disk will be even smaller for the random declustering layout to provide the same reliability as achieved with the shifted declustering.

4.6. Performance Analysis and Comparison

For sequence read accesses, we use a metric named maximum parallel read counts. For M concurrent reads and N storage nodes, if M is less than N, the maximum parallel read counts is M, otherwise the maximum parallel read counts is N. For shifted declustering layout, it is easy to achieve maximum parallel read counts. For random declustering and copyset replication, the primary data are distributed randomly, thus, these two schemes could satisfy this metrics sometimes. However, for small random read accesses, the primary data in shifted declustering layout is organized in a hash function. The hash function may map

the popular data to a small subset of overall disks, which may cause an unbalanced workload distribution. Thus, for random-dominated request distribution, shifted declustering will perform worse than the other two schemes.

For write performance, the write performance of these configurations are based on the load-balancing capacity. If the workloads are evenly distributed all the time, the write performance of these configurations will be almost the same. However, if the workload is unevenly distributed, the number of disks, which can host the copies of hot-spot node, are different for different data layout schemes. The copyset replication has worse write performance than other two data layout schemes, because of smaller scatter width.

In addition to the performance comparison in normal mode, we should also discuss the performance comparison in degraded mode. The degraded mode means the scenario when partial nodes failed or powered down. In degraded mode, the shifting declutering will perform better than the random declutering, because shifting declustering scheme spread the data more evenly. At the same time, the random declustering performs better than copyset replication. As the scatter width of copyset replication is much smaller than the other two schemes, the copyset replication is the worst scheme in the degraded mode.

In conclusion, for sequence read accesses, shifted declustering outperforms the other two schemes. However, for random read accesses, shifting declustering is the worst scheme in the case when the popular blocks are mapped to a small subset of overall disks. Random replication will achieve the best performances among these three data layout schemes. The write performance of these configurations are the same under evenly distributed workload mode. Under unevenly distributed workload, copyset will be the worst case due to the smallest scatter width. In addition to the performance impact, we compare the applicability of these data layout schemes. While the random declustering and copyset replication can be used both for block-level and object-based storage system, the shifted declustering cannot be applied to object-based storage system.

In addition to the response time, the recovery time is also an important metric to measure the data layout schemes. Firstly, we make comparison between

the copyset replication and random declustering layout. The difference between these two layout schemes are the number of disks to host the R-1 secondary replicas. Thus, the parallelism of recovery under two schemes are different. Assume the number of nodes is N, the replication factor is R, and the scatter width is S. For the random scheme, the value S is designated as $N-1$. Thus the $R-1$ secondary replicas are randomly distributed in $N-1$ nodes. As the copyset replication is designed to reduce the probability of data loss, the value S is set far smaller than N. We take an example to show that the recovery rate under random replication will be larger than copyset replication. For example, assume that N=9, R=3 and S=3. For copyset replication, the R-1 secondary replicas are held in 3 disks. For random replication, the R-1 secondary replicas is distributed in 8 disks. Secondly, we compare shifted declustering and random declustering. The shifted declustering will be better than Random scheme, because the R-1 secondary replicas within shifted declustering are evenly distributed in the N-1 nodes. In conclusion, the recovery time of shifted declustering will be the best one and Copyset replication is the worst one.

5. Simulation Results

5.1. Methodology

We use the SHARPE package [27] for the simulation of system reliability. The simulation time is 10 years, in steps of a month. We simulate a storage system of 999 disks with both of the HDDs and SSDs, configured as three-way replication, with shifted ,random and Copyset layouts.

For the disk failure rate λ, we use 3% ARR (annual replace rate), which corresponds to $\lambda = 0.25\%$ per month for hard disk drives. This value is from Schroeder et al.'s observation [28]: in real-world large storage and computing systems, the ARR of hard drives is between 0.5% and 13.8%, and 3% on average. It is much higher than the failure rate provided by the most vendors, a MTTF (mean time to failure) of a single disk from 1,000,000 to 1,500,000 hours, which corresponds to an AFR (annual failure rate) between 0.58% and 0.88% [28].

As it refers to the solid state drive (SSD), we use 1.5% annual disk failure rate, which corresponds to $\lambda = 0.125\%$ per month. This value is from the observations of Enterprise storage forum [9].

The size of a data replica was set to 64MB, which is a typical data block size for Hadoop File system(HDFS). The capacity of a disk used for the storage is 1TB, which is a normal size of a hard disk drive used for storage systems. In our simulation, all the disks are considered to be hard disk drives (HDD). Other storage devices such as SSD are out of the research scope of our paper.

The recovery rate upon l failed disks is determined by the product of the available source disks and the sequential recovery rate offered per disk (μ). We will simulate the reliability for 10 years with different values of μ. In the Section 5.3.2, we give a range of the recovery rate upon l failed disks (μ_l) for standard mirroring and chained declustering, but we only take their highest recovery rate in the simulation.

The values of parameters in the simulation are listed in Table 1.

To evaluate the impacts of these data layout schemes on performance, we make an experiment in a 3-server cluster with 15 storage nodes. While the storage nodes can be server or storage-devices, we use one HDD as one storage node. These storage servers are connected by a gigabit ethernet switch, and each storage server has a 4-core CPU, a 32GB RAM memory and 8 storage device interfaces. The disks we used for evaluation are of the type IBM_DNES-309170W. The number of data surfaces is 5 and the number of cylinders is 11474. While the rotation speed (in rpms) is 7200, head switch time is 0.062000.

In order to compare the impacts of different data placement policy on the performance of storage system, we modified an open-source block-level distributed storage system called Sheepdog. This distributed storage system is designed for the KVM/QEMU. Sheepdog exploited a Consistence Hash Table as the default data placement. While the primary data placement of random declustering and copyset replication follow the same policy, the differences are the secondary and tertiary data placement schemes. For random declustering, the secondary and tertiary replicas are randomly distributed. We use the

Table 1: Parameters in Simulation

Parameter and description	**Value**
n: Number of disks in the system	999
k: Number of ways of replication	3
l: Number of failed disks	3 to 666
$P^{(l)}$: Probability of no data loss upon l-disk failure	See Section 3
λ: Disk failure rate (HDD)	0.25% per month
Disk failure rate (SSD)	0.125% per month
λ_l: Transition rate from l-disk failure to $l+1$-disk failure **without** data loss	$(n-l)\lambda P^{(l+1)}$
γ_l: Transition rate from l-disk failure to $l+1$-disk failure **with** replica lost	$(n-1)\lambda(1-p_b)$
δ_l: Transition rate from l-disk failure to $l+1$-disk failure **with** data loss	$(n-l)\lambda(1-P^{(l+1)})$
b_r: Reserved bandwidth for recovery per disk	0 KB to 10 MB
S: Capacity of a disk used for storage(HDD and SSD)	1 TB
μ: Sequential recovering rate	b_r/S
μ_l: Recovery rate from $l+1$-disk failure to l-disk failure	See Section 3
s: Size of a data replica	64 MB
r: Number of redundancy groups	5,455,872
N: Number of the total data blocks per disk	16,384

chained declustering to implement the copyset replication, which means the scatter width is 3. To implement shifted declustering, we modified the data placement based on the formulas in the paper[].

The real-life workload traces used in our evaluation are collected from 36 volumes within 13 enterprise servers by researchers in Microsoft Research Cambridge. The data of these traces are one-week real-life formatted IO data including timestamp, type, request size and so on. We use a tool called btrepay to replay these IO traces. We choose 4 representative traces from these real-life traces. The characteristics of these traces are as shown in Table 2. As shown in Table 2, trace *usr* has the largest IOPS and average request size.

Table 2: The Features of Traces

Traces	Write Ritio	IOPS	Avg. Req	Function
usr	59%	83.87	22.66KB	User home directories
web	70%	50.32	14.99KB	Web SQL server
mds	88%	18.41	9.19KB	Media server
rsrch	91%	21.17	8.93KB	Research projects

5.2. Sampling Procedures

The sampling procedures for a storage with 999 disks and three-way replications with shifted and random declustering are as follows:

- For shifted declustering, the values are obtained by the following simulation procedure: for any l between 3 and 666,[6] randomly generate 10,000 vectors of l numbers between 0 and 998 as failed disks; for each vector, check whether this particular failed l disks causes data loss according to the criteria discussed in Section 4.1. $P^{(l)}$ is estimated by the result of dividing the number of vectors that do not cause data loss by 10,000. This process is repeated 5 times, and the average $P^{(l)}$ is used as the final value.

- For copyset replication, we assume the capacity for storage is 1 TB per disk, the data replica size is 64 MB, so the 999-disk system can store 5,455,872 redundancy groups in total. We generate 5,455,872 random vectors with three numbers between 0 and 998 to represent where the three replicas in each redundancy group are distributed.

- For random declustering, we assume the capacity for storage is 1 TB per disk, the data replica size is 64 MB, so the 999-disk system can store 5,455,872 redundancy groups in total. We generate 5,455,872 random vectors with three numbers between 0 and 998 to represent where the three replicas in each redundancy group are distributed. The following

[6] For 999-disk three-way replication storage, if the number of failed disks is more than 2/3 of the total number of disks, data loss becomes a definite event.

procedure is similar to that used for shifted declustering, for any l between 3 and 666, randomly generate 10,000 vectors of l numbers between 0 and 998 as failed disks; for each vector, check whether this particular failed l disks cause data loss by checking whether it contains three elements of any vector that is used to store a redundancy group. $P^{(l)}$ is estimated by the result of dividing the number of combinations that do not cause data loss by 10,000. Repeat the process 5 times, and compute the average $P^{(l)}$.

5.3. Probability of No-data-Loss

With the methodology introduced in Section 5.1, we calculate the value of $P^{(l)}$ (the probability of no data loss in case of l disk failure) in a 999-disk system ($n = 999$) with 3-way replication ($k = 3$), configured with shifted and random declustering layout schemes by Matlab [25]. The value of $P^{(l)}$ is illustrated in Figure 6.

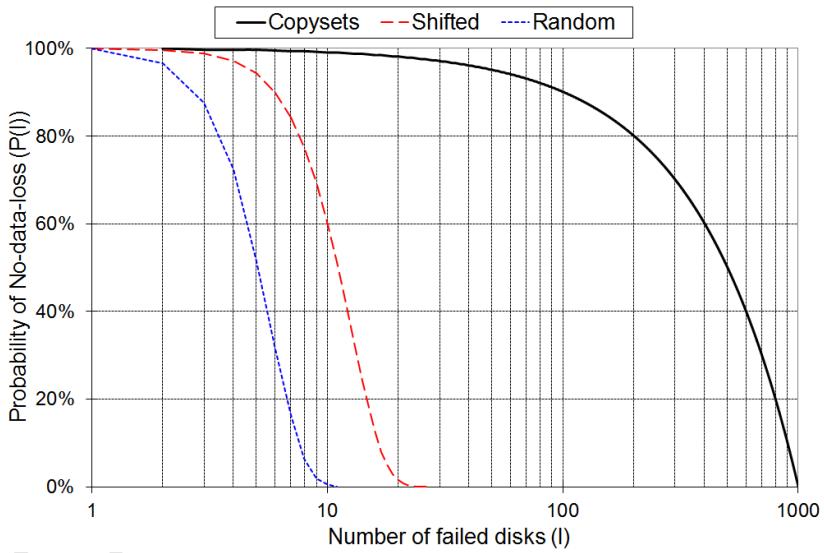


Figure 6: Probability of no data loss in a 999-disk system ($P^{(l)}$)

Theoretically, a 999-disk 3-way replication system can tolerate as many as 666 failed disks, but $P^{(l)}$ drops close to 0 much earlier than l approaching 666, as shown in Figure 6. For shifted declustering and random declustering layouts,

$P^{(l)}$ drops to 0 at 28 and 13, respectively. For the same number of failed disks, We can see that our shifted declustered layout always has the higher possibility of not losing data comparing with random declustering.

5.3.1. System Reliability without Recovery

If the system does not apply any recovery mechanism, the system reliability can be obtained by assigning $b_r = 0$. The system reliability of shifted declustering and random declustering layouts are illustrated in Figure 7. The physical meaning of system reliability at time t (represented by $R_{\text{system}}(t)$) is the probability of the system surviving until the time point t. The reliability without recovery is a direct reflection of $P^{(l)}$. Thus, a higher probability of the system not losing data results in a higher overall system reliability rating due to it being more difficult to enter the failed state. We can see that after the third month, the reliability is almost 84% and 60% for shifted declustering and random declustering respectively. This reliability results has the same tendency of the $P^{(l)}$ of these two layouts. Without considering the recovery, the Copyset layout achieve the best reliability result because of its high probability of no-data-loss.

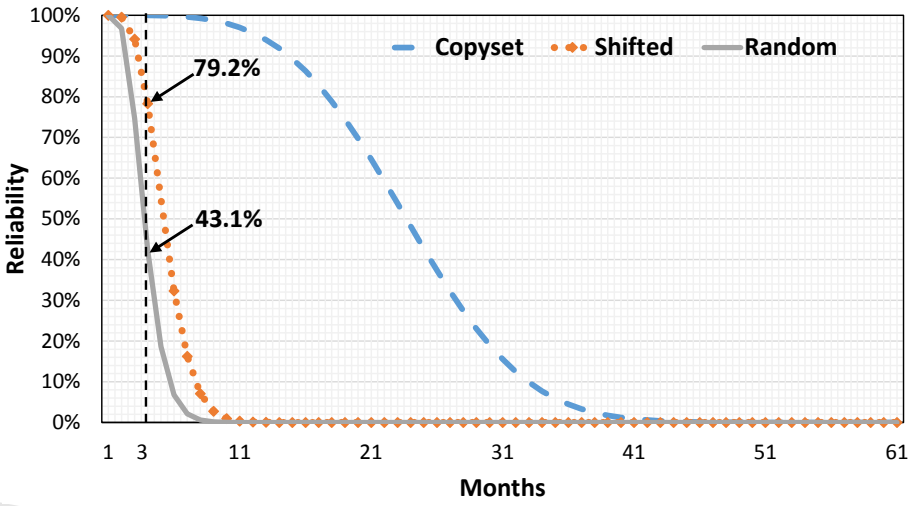


Figure 7: System reliability of a 999-disk system without recovery

With this specific configuration, shifted declustering has a significant advantage in terms of reliability to random declustering. This result is consistent with our analysis in Section 5.3.

5.3.2. System Reliability with Recovery

The previous section discusses the storage system reliability without data recovery in place. It reflects how reliable the system will be with different data layouts, when disk failures are not detected, or replicas on failed disks are never recovered. In this section, we simulate the system reliability of different layouts with the aggressive recovery schemes introduced in Section 4.4.

With the aggressive recovery schemes, shifted declustering layout has the best reliability for a given recovery bandwidth per disk for recovery (b_r) among all layouts. The simulation results in Section 5.3.1 show that with recovery, shifted declustering layout's system reliability is better than the random declustering and the Copyset. This is reflected in the fact that the shifted declustering layout is slower to transmit from non-data-loss states to the data-loss state than the other two. Additionally, although the Copyset layout has a higher probability of no-data-loss, the amount of data lost is huge compared with the other two layouts. Therefore, when we take data recovery into consideration, the system reliability will decrease. As a result, even these three layouts have the same potential of optimal parallel recovery, the reliability with recovery of shifted declustering exceeds the other two.

We can see that from Figure 8, shifted declustering with 10 KB/sec recovery bandwidth per disk (5 MB/sec accumulative recovery bandwidth for 999-disk system) obtains more than 99% reliability after 5 years, while the Copyset layout requires more than 100 KB/sec recovery bandwidth per disk to achieve the same reliability. Similarly, to reach the same reliability, the random declustering layout has to use 20 KB/sec. It is true that the reliability can be improved by increasing the recovery bandwidth to obtain a higher recovery rate. Nonetheless, as long as the disk transfer bandwidth is fixed, higher recovery bandwidth means lower bandwidth for normal services, so higher recovery bandwidth will

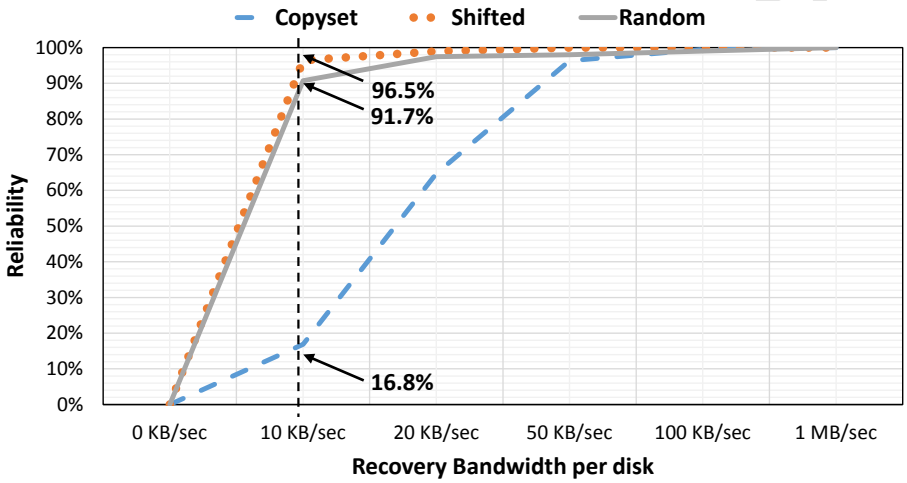


Figure 8: Reliability after 5 years

drag down the normal service performance of the disks involved in recovery. Obviously, for the same reliability goal, the burden on surviving disks of shifted declustering is the lightest, because the service load is balanced among all surviving disks, and the bandwidth used by each disk for recovery is low. In comparison, for the Copyset, the service load on the source disks is heavier than the other surviving disks, because they need to serve requests which are supposed to be served by the failed disks besides normal service; the recovery bandwidth per disk is also higher. These two reasons result in a load imbalance more severe in Copyset. If we want to achieve the same recovery bandwidth of 120 MB/sec, which is the highest recovery bandwidth obtained in the Panasas parallel file system [32] with around 120 disks, corresponding to 999 MB/sec for a 999-disk system three-way replication, for shifted declustering, about 2 MB/sec recovery bandwidth per disk is enough. While for Copyset, about 500 MB/sec recovery bandwidth is needed, which is almost impossible to obtain with current storage technology.

Figure 9 shows the system reliability during the first ten years, with the recovery bandwidth per disk set to 10 KB/sec (b_r = 10 KB/sec). Currently

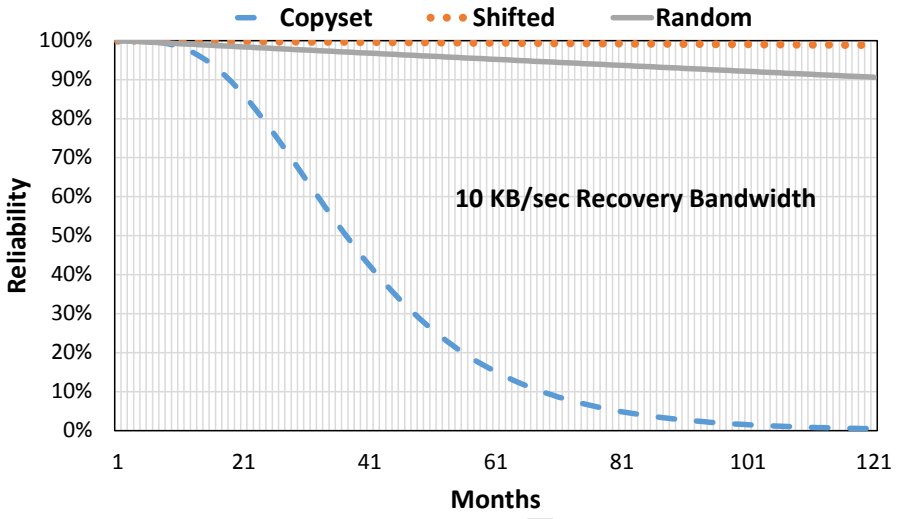


Figure 9: System reliability with 10 KB recovery bandwidth per disk

the sustained transfer rate of hard drives can be higher than 100 MB/sec[7], so 10 KB/sec dedicated to recovery has little impact on the regular service workloads. We can see that with such a small recovery bandwidth per disk, the shifted declustering layout obtains a near 99% reliability even after ten years (Figure 10). With the same recovery bandwidth, the reliability of other layouts is lower. To reach the same reliability as shifted declustering, other layouts need more recovery bandwidth per disk.

5.3.3. System reliability with and without considering probability of replica lost

In order to show the accuracy of our proposed reliability model, we conduct a comparison between the system reliability with and without considering the probability of replica lost. We run the simulation to obtain the 5 years system reliability results for all the three replication methods by setting the system recovery bandwidth to 10KB/sec. In the first round of simulation, we do not considering about the probability of replica lost with setting the initial system

[7]The Barracuda 7200.11 Serial ATA (released May 2008 by Seagate) is specced to have a sustained transfer rate of 105 MB/sec [7].

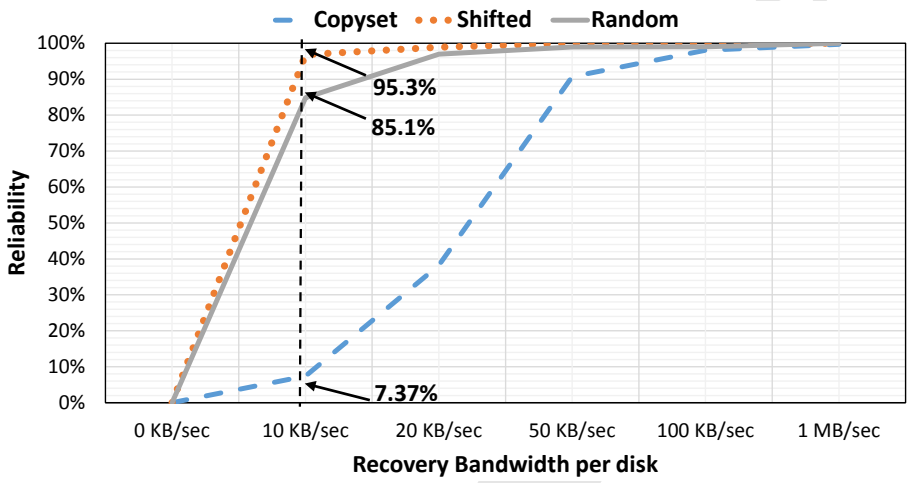


Figure 10: Reliability after 10 years

reliability equals to 1. In the second round of simulation, we have taken probability of replica lost into account by utilizing our proposed model and obtained a more precise outcomes.

The results have been illustrated in Figure 12. Based on the results, we can see that the probability of replica lost do have an influence on the overall system reliability. For shifted, random and copyset replication methods, the system reliability result with considering the probability of replica lost are 96.5%, 91.7% and 16.8% respectively. However, we obtain 99.4%, 94.2% and 18% for the above three replication methods by introducing the probability of replica lost. The differences among these two set of experimental results are 2.9%, 2.5% and 1.2% respectively. The reason that the overall system reliability is low for copyset replication approach is because of the selected low recovery bandwidth (10KB/sec), which was discussed in Section 5.3.2.

5.3.4. The comparison between System reliability of HDD and SSD

A typical SSD uses NAND-based flash memory which is a non-volatile type of memory. With this essential characteristic, you can read and write to an SSD all day long and the data storage integrity will be maintained for well over

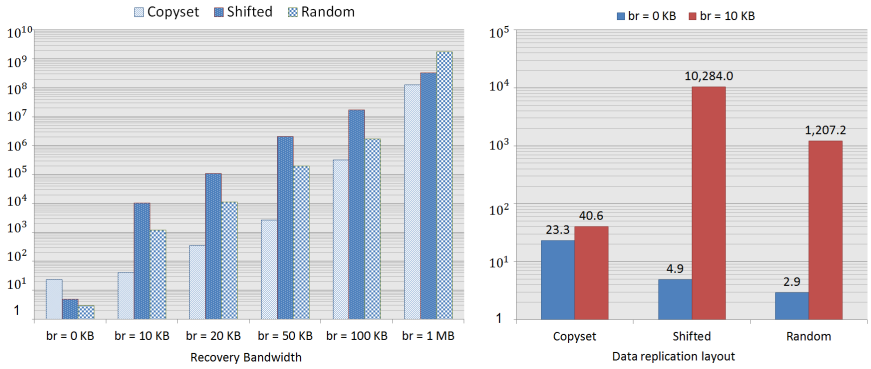


Figure 11: Mean Time to Data Loss

200 years. An SSD does not have a mechanical arm to read and write data, it instead relies on an embedded processor called a controller to perform a bunch of operations related to reading and writing data. An example of a fast controller today is the SandForce SATA 3.0 (6GB/s) SSD controller that supports burst speeds up to 550MB/s read and write speeds. Due to its great performance compared to traditional hard disk drive, lots of enterprise storage systems are seeking to use SSDs as the major storage device if the faster performance is their major consideration. Conversely, an HDD might be the right choice if you need a large amount of storage capacity, spend less money and do not care too much about the boost up speed. Since SSD gain an increasingly popularity in future's storage systems, we carried out an comparison of the reliability analysis between HDD and SSD to give the system designers an overview about these two types of storage systems.

From the Figure 13, we can see that the storage systems using SSD have a better system reliability for all three data replication methods. This is because SSD has a lower disk failure rate compare to the HDD. The comparison of average 5 year's reliability between HDD and SSD that is shown in Figure 14 illustrates that the reliability of SSD outperform HDD for all three data replication methods, Copyset, Shifted and Random by 8.1%,4.13% and 3.7% respectively.

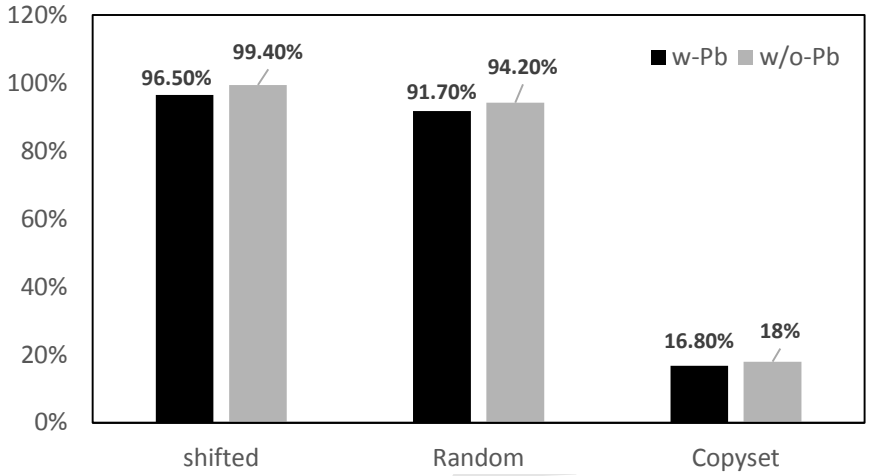


Figure 12: Average system Reliability with and without considering probability of replica lost, recovery bandwidth = 10KB/sec

5.4. Mean Time to Data Loss

Mean time to data loss (MTTDL) is the expected duration between the state of all disks functioning in a storage system (state 0 in Figure 1) and the state of data loss (state F in Figure 1). Figure 11 demonstrates the simulation results of mean time to data loss based on the parameters listed in Table 1.

The left bar in each group is the MTTDL without recovery. The Copyset lasts the longest, 23.3 months. We can see that without recovery, even the most "reliable" system is unacceptable, because data in that system will start to lose approximately two years. The right bar in each group is the MTTDL with 10 KB/sec recovery bandwidth per disk. With the advantage of parallel recovery, shifted declustering and random layouts have higher accumulative recovery bandwidth than the Copyset layout. As a result, the MTTDL is improved by 253 times and 29 times in the shifted declustering and random layout respectively.

In addition, the difference in reliability of shifted declustering and random declustering layouts are not very significant. For example, after 5 years, shifted declustering layout has a 99% reliability, and random declustering layout has

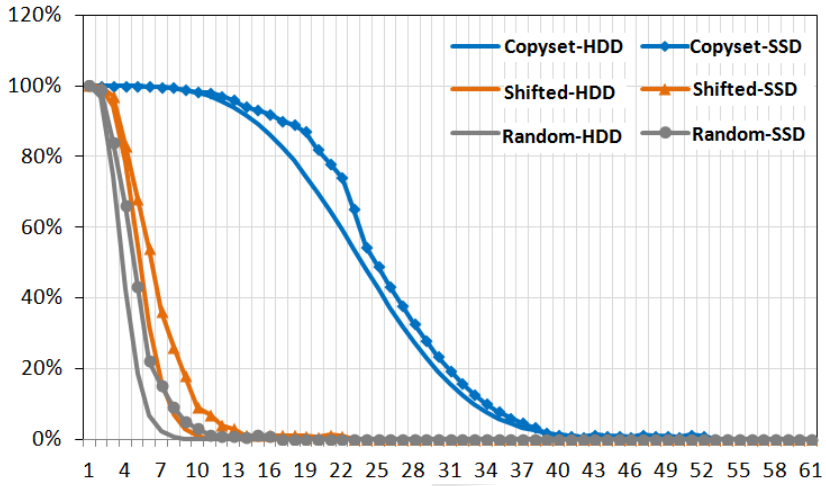


Figure 13: The comparison of 5 year's reliability between HDD and SSD

a 95% reliability; after 10 years, the values becomes 98% and 90%. However, the difference of system reliability yields significant improvement in terms of MTTDL in the shifted declustering layout compared to the other two layouts. In the shifted declustering layout, an almost 8.5 times higher MTTDL than the second ranked layout, random declustering layout, is observed in our simulation environment. Tt is also 254 times higher than the Copyset. These result indicates that the shifted declustering layout achieves the highest reliability given a fixed resource budget among all layouts in comparison. On the other hand, it has the maximum flexibility to be tuned to a desired reliability goal.

5.5. The performance impact

Figure 15 shows the average response time under various traces. First, the random declustering outperforms other two schemes. Hash functions used in the shifted declustering may map the popular data to a small subset of overall disks. This will give rise to a unbalanced workload distribution resulting in performance degradation. The copyset replication has a worse performance than random declustering because only a small set of disks host the replicas of hot-spot node. Second, the average response time of random and Shifting

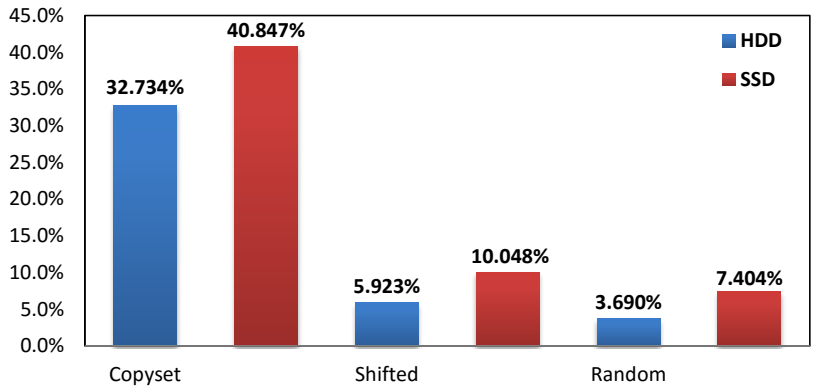


Figure 14: The comparison of average 5 year's reliability between HDD and SSD

Declustering are almost the same under traces mds and rsrch. This is because the read/write ratios is so small that the write performance of two schemes are almost the same. Third, the copyset replication has the largest average response time because of the placement of the secondary and tertiary replicas, which are concentrated in a small subset of all disks. Last, the trace *usr* has the largest response time among four different traces. This is because the trace *usr* has the highest IOPS and the largest average request size as shown in Table 2.

Percentile latency is an importance metric of the response of a data center. In order to get into the insight of comparison, Figure 16 shows the average response time and 50th/90th/95th percentile latency of three data layout schemes under workload trace *usr*. We can find under the 50th and 90th percentile latency, the response time of different configurations are almost the same. This is because they have same performance in normal mode and under load-balanced scenarios. However, the unbalanced load will increase the gap between these configurations. Thus, the 95th latency of copyset replication is much larger than random declustering and shifted declustering.

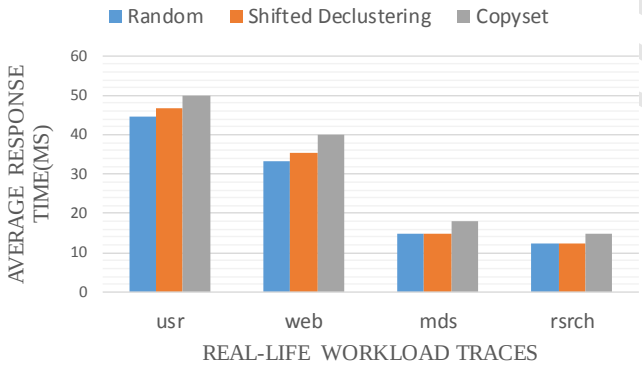


Figure 15: The average response time of three configurations under various workload traces

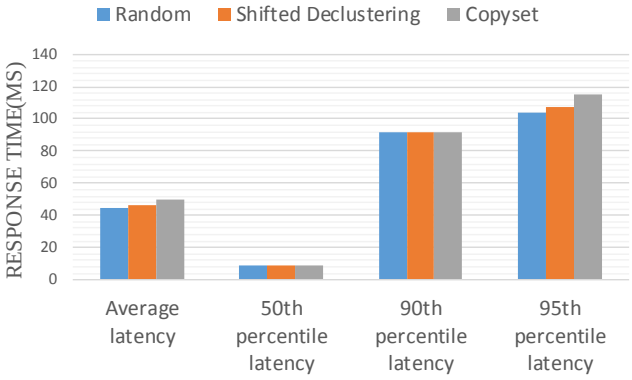


Figure 16: The 50th/90th/95th percentile latency under the trace *usr*

6. Conclusions

In this paper, we have modeled the reliability of multi-way replication storage systems with different data layout schemes. We make the following conclusions:

- Modeling of the system reliability should not only take data loss probability and recovery bandwidth into account, but also need to consider about the probability of replica lost in order to obtain an accurate result.
- The reliability of random declustering layout is highly dependent on the number of redundancy groups in the system. With the increase of redundancy groups and/or the number of disks in the system, the reliability of

random declustering drops.

- The shifted declustering is less sensitive to the scale of the storage system comparing with random declustering. With the same resource provided for recovery per disk, the shifted declustering layout achieves almost 100% reliable that last for 10-years long period. In particular, the data integrity of the shifted declustering layout lasts 85% times longer in our simulation than random declustering layout.

- The Copyset replication method obtained the best system reliability due to its carefully arrangement of the data blocks. However, by considering of the recovery bandwidth, the system reliability has been greatly affected especially when the bandwidth is low.

- Our study on both 5-year and 10-year system reliability equipped with various recovery bandwidth settings shows that, the shifted declustering layout surpasses the two baseline approaches in both cases by consuming up to 83 % and 97% less recovery bandwidth for copyset, as well as 5.2% and 11% less recovery bandwidth for random layout.

- The experiment results of performance impacts demonstrate that random declustering outperforms other two schemes and the Copyset replication is the worst-case scheme. The Copyset performs worse than other replication schemes as we expected because only a small set of disks host the replicas of the hot-spot node. In other words, the placement of the secondary and tertiary replicas within the Copyset replication are concentrated in a small subset of all disks. Hash functions employed in the shifted declustering may map the hot data to a small subset of overall disks. This data layout schemes may give rise to performance degradation due to the uneven workload distribution.

As random declustering layout is widely adopted by enterprise large-scale systems configured with multi-way replication, shifted declustering layout is a

promising alternative for its proved optimal performance and high reliability with low recovery overhead.

[1] A. Cidon, S. Rumble, R. Stutsman, S. Katti, J. Ousterhout and M. Rosenblum Copysets: reducing the frequency of data loss in cloud storage, *Presented as part of the 2013 USENIX Annual Technical Conference*, pages 37–48, 2013

[2] Cisco Nexus 7000 Series 32-port 10Gb Ethernet Module, 80Gb Fabric. http://www.cisco.com/en/US/prod/collateral /switches/ps9441/ps9402/ps9512/Data_Sheet_C78- 437757.html, 2008.

[3] Hadoop project. http://hadoop.apache.org/core/, 2008.

[4] Inclusion-exclusion principle. http://mathworld.wolfram.com /Inclusion-ExclusionPrinciple.html, 2008.

[5] F.Dinu, and T.S. Eugene Ng Understanding the Effects and Implications of Compute Node Related Failures in Hadoop, *Proceedings of the High Performance Parallel and Distributed Computing (HPDC)*, pages 187-197, 2012.

[6] Q. Xin, E.L. Miller, and T.J.E. Schwarz, Evaluation of distributed recovery in large-scale storage systems, *Proceedings of the 13th IEEE International Symposium on High performance Distributed Computing*, pages 172-181, 2004.

[7] Product Manual Barracuda 7200.11 Serial ATA. http://www.seagate.com/staticfiles/support/disc/manuals /desktop/Barracuda%207200.11/100452348e.pdf, 2008.

[8] Seagate Technology - Barracuda 7200.11 SATA 3GB/s 1.5-GB Hard Drive. http://www.seagate.com/ww /v/index.jsp?vgnextoid=511a8cf6a794b110VgnVCM100000f 5ee0a0aRCRD&locale=en-US&reqPage=Model &modelReqTab=Features, 2008.

[9] Enterprise Storage forum. http://www.enterprisestorageforum.com/storage-hardware/ssd-vs.-hdd-performance-and-reliability-1.html, 2014.

[10] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer. Farsite: Federated, available, and reliable storage for an incompletely trusted environment. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, pages 1–14, 2002.

[11] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler. An analysis of latent sector errors in disk drives. *SIGMETRICS Perform. Eval. Rev.*, 35(1):289–300, 2007.

[12] R. E. Bryant. Data-intensive supercomputing: The case for DISC. Technical Report CMU-CS-07-128, Carnegie Mellon University, May 2007.

[13] M.-S. Chen, H.-I. Hsiao, C.-S. Li, and P. S. Yu. Using rotational mirrored declustering for replica placement in a disk-array-based video server. *Multimedia Syst.*, 5(6):371–379, 1997.

[14] S. Chen and D. Towsley. A performance evaluation of RAID architectures. *IEEE Trans. Comput.*, 45(10):1116–1130, 1996.

[15] G. Copeland and T. Keller. A comparison of high-availability media recovery techniques. In *SIGMOD '89: Proceedings of the 1989 ACM SIGMOD international conference on Management of data*, pages 98–109, New York, NY, USA, 1989. ACM.

[16] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazon's highly available key-value store. In *SOSP '07: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, pages 205–220, New York, NY, USA, 2007. ACM.

[17] J. Gafsi and E. W. Biersack. Modeling and performance comparison of reliability strategies for distributed video servers. *IEEE Transactions on Parallel and Distributed Systems*, 11:412–430, 2000.

[18] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. *SIGOPS Oper. Syst. Rev.*, 37(5):29–43, December 2003.

[19] A. Gulati, A. Merchant, and P. J. Varman. pclock: an arrival curve based approach for qos guarantees in shared storage systems. *SIGMETRICS Perform. Eval. Rev.*, 35(1):13–24, 2007.

[20] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 4th edition, 2006.

[21] M. Holland and G. A. Gibson. Parity declustering for continuous operation in redundant disk arrays. In *ASPLOS*, pages 23–35, 1992.

[22] R. Honicky and E. Miller. Replication under scalable hashing: a family of algorithms for scalable decentralized data distribution. *In Proceedings of 18th International Parallel and Distributed Processing Symposium, 2004.*, page 96, 26-30 April 2004.

[23] H.-I. Hsiao and D. J. DeWitt. Chained declustering: A new availability strategy for multiprocessor database machines. In *Proceedings of the Sixth International Conference on Data Engineering*, pages 456–465, Washington, DC, USA, 1990. IEEE Computer Society.

[24] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. In *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 251–262, New York, NY, USA, 1999. ACM.

[25] The MathWorks. Matlab. http://www.mathworks.com.

[26] R. Nelson. *Probability, Stochastic Processes, and Queueing Theory*. Springer-Verlag, 1995.

[27] R. Sahnar, K. S. Trived, and A. Puliafito. Sharpe. http://www.ee.duke.edu/ kst/software_packages.html, 2001.

[28] B. Schroeder and G. A. Gibson. Understanding disk failure rates: What does an mttf of 1,000,000 hours mean to you? *Trans. Storage*, 3(3):8, 2007.

[29] A. Thomasian and M. Blaum. Mirrored disk organization reliability analysis. *IEEE Transactions on Computers*, 55(12):1640–1644, 2006.

[30] A. S. Tosun. Analysis and comparison of replicated declustering schemes. *IEEE Trans. Parallel Distrib. Syst.*, 18(11):1578–1591, 2007.

[31] K. S. Trivedi. *Probability & Statistics with Reliability, Queuing, and Computer Science Applications*. Prentice-Hall, 1982.

[32] B. Welch, M. Unangst, Z. Abbasi, G. Gibson, B. Mueller, J. Small, J. Zelenka, and B. Zhou. Scalable performance of the panasas parallel file system. In *FAST'08: Proceedings of the 6th USENIX Conference on File and Storage Technologies*, pages 1–17, Berkeley, CA, USA, 2008. USENIX Association.

[33] J. M. Wing. Computer science meets science and engineering. HEC FSIO R&D Workshop, NSF, August 2007.

[34] Q. Xin. Understanding and coping with failures in large-scale storage systems. Technical Report UCSC-SSRC-07-06, Storage Systems Research Center, Baskin School of Engineering, University of California, Santa Cruz, May 2007. This Ph.D. thesis was originally filed in December, 2005.

[35] Q. Xin, E. L. Miller, T. Schwarz, D. D. E. Long, S. A. Brandt, and W. Litwin. Reliability mechanisms for very large storage systems. In *MSS '03: Proceedings of the 20 th IEEE/11 th NASA Goddard Conference on Mass Storage Systems and Technologies (MSS'03)*, page 146, Washington, DC, USA, 2003. IEEE Computer Society.

[36] H. Zhu, P. Gu, and J. Wang. Shifted Declustering: a placement-ideal layout scheme for multi-way replication storage architecture. In *ACM International Conference on Supercomputing*, pages 134–144, 2008.

[37] J. Wang, R. Wang, J.Yin, H. Zhu, and Y.Yang. Reliability Analysis on Shifted and Random Declustering Block Layouts in Scale-out storage Architectures. In *IEEE International Conference on Networking, Architecture and Storage*, Tianjing, 2014.

[38] R. J. Chansler. Data Availability and Durability with the Hadoop Distributed File System In *The USENIX Magazine*, February 2012.

[39] J. Dean. Evolution and future directions of large-scale storage and computation systems at Google. In *SoCC*, page 1, 2010.

[40] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan. Availability in globally distributed storage systems. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, pages 1-7, Berkeley, CA, USA, 2010. USENIX Association.

Jun Wang received the BEng degree in computer engineering from Wuhan Technical University of Surveying and Mapping (now Wuhan University), MEng degree in computer engineering from Huazhong University of Science and Technology, and the PhD degree in computer science and engineering from the University of Cincinnati, in 2002. He is a tenured associate professor of computer science and engineering, and director of the Computer Architecture and Storage Systems (CASS) Laboratory, University of Central Florida, Orlando, Florida. He has won UCF Reach For The Stars Award in 2015, and was named Charles N. Millican Faculty Fellow in EECS during 2010–2012. He received the National Science Foundation Early Career Award 2009 and Department of Energy Early Career Principal Investigator Award 2005. He has authored more than 100 publications in premier journals such as the *IEEE Transactions on Computers*, the *IEEE Transactions on Parallel and Distributed Systems*, and leading HPC and systems conferences such as VLDB, HPDC, EuroSys, ICS, Middleware, FAST, IPDPS. He has graduated nine PhD students who upon their graduation were employed by major US IT corporations (e.g., Google, Microsoft, etc.). He has served as numerous US NSF grant panelists and US DOE grant panelists and TPC members for many premier conferences such as IPDPS, ICPP, HiPC. He currently serves in the editorial board for the *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Parallel and Distributed Systems* 2012-2014, and was associate editor for the *International Journal of Parallel, Emergent and Distributes Systems* (IJPEDS) during 2010–2012. He serves in the general chair for IEEE DASC/CyberSciTech/DataCom/PIcom 2017. He serves as a program co- chair for 10th IEEE conference on Network, Architecture and Storage (NAS) in June 2015, and the storage track for 7th NAS in June 2012. He has co-chaired the 1st International Workshop on Storage and I/O Virtualization, Performance, Energy, Evaluation and Dependability (SPEED 2008) held together with HPCA.

Huafeng Wu received the undergraduate degree from Navigation Technology of Jimei University (Junior preparatory classes), in 1997, the master's degree in traffic information engineering and control from Dalian Maritime University, in 2004, and the PhD degree in computer application technology from Fudan University, in 2008. He is a professor, PhD supervisor, executive member of Shanghai Institute of Electronics and Shanghai Shuguang Scholar. In 2008-2009 he was engaged in postdoctoral research with Carnegie Mellon University; then he was study with Shanghai Jiao Tong University as a visiting scholar in 2012-2013; and currently he is the deputy director of Human Resource Division and an associate professor in the Merchant Marine College Shanghai Maritime University. He has enthusiasm for the basic theory of computer, communication and network and its application in the field of transportation and maritime, including wireless sensor network (WSN), radio frequency identification systems (RFID), cloud computing and Internet-of-Things, and so on. He has published more than 70 papers in authoritative journals such as the *International Journal of Distributed Sensor Networks*, *Journal of Communication*, and top International conference such as the IEEE ICDCS and IEEE PerCom, most of them are indexed by SCI, EI. He presides over the national Natural Science Fund of China (NSFC) projects, key project of natural science fund of Shanghai, Shanghai ShuGuang plan talent project, applied basic research project of ministry of transport, Shanghai Education Research and Innovation Key Project, selection and training of outstanding young teachers in universities in Shanghai project, and he also participates in the United States NSF project, national 863 plan project, ministry of science and technology special program, the national natural science fund project, key project of Shanghai science and technology commission and PuJiang talent plan and so on; and he has been authorized 3 national invention patents, 6 utility model patents.

Ruijun Wang received the BS degree in Electrical Information Engineering from the University of Petroleum China (Beijing), in 2007 and the MS degree in information systems from the University of Central Queensland, in 2009. She is currently working toward the PhD degree in the School of EECS, University of Central Florida, Orlando. Her research interests include energy-efficient computing and storage systems.