ATLANTIS PRESS

Advances in Economics, Business and Management Research, volume 31

4th International Conference on Management Science and Management Innovation (MSMI 2017)

An Easy, Quantitative Method to Evaluate Replica Placement Policies in Distributed Storage Systems

Chang LIU
Department of Logistics and Information Management
Jianghan University
Wuhan, China
e-mail: liuchang@jhun.edu.cn

Duanming ZHANG
Department of Physics
Huazhong University of Science and Technology
Wuhan, China
e-mail: zhangduanming@163.com

Zhao WANG
Big Data Research Center
Jianghan University
Wuhan, China
e-mail: 13971164218@163.com

***Abstract*—In large scale distributed systems, data blocks are usually replicated and distributed across the storage nodes to achieve high availability and reliability. The placement of blocks and their replicas has a marked impact on the data reliability and system performance. Various replica placement policies have been proposed in the past decades, and some of them have been implemented in enterprise storage systems. However, it is difficult to verify that a selected placement scheme yield near-optimal performance or reliability in the real world due to several reasons. First, it is not feasible to build up a physical testbed with hundreds of storage nodes; and constructing a simulation model is also costly and hard to catch up with the fast-paced changes in industry. Second, previous studies usually carry out the analysis to address only one particular facet of schemes. Third, most algorithms are too complicated for programmers to adopt. Hence a synthetic and cost efficient method is needed. Here we use an analytic model to assess reliability and performance of replica placement policies by using a combination of quantitative metrics. We claim that this method is useful for both the analysis in the stage of architecture design and the parameter selection of existing systems. Four replica placement policies with deterministic declustering are under consideration.**

Keywords-replica placement; declustering; reliability; load distribution; distributed storage system

I. INTRODUCTION

Large scale distributed storage systems aggregate the disks of interconnected nodes or the nodes on internet. The input data are divided into blocks and each block is replicated and distributed among the nodes. The block-node mapping information is maintained in the memory either on a master node or on distributed nodes. Such architecture contributes to various advantages such as parallel processing, load balancing, reduced access time (high availability) and increased fault tolerance (high reliability).

Studies have shown that the placement strategy for the blocks and their replicas significantly affects system reliability and performance [1–3]. Numerous replica placement schemes have been proposed. Here we classify them according to the algorithm being used.

Hashing-based Hashing is widely adopted by block placement schemes due to its random nature. These schemes have demonstrated remarkable load balancing ability. The representative architectures include Oceanstore [4, 5] which places the replica based on a tree structure [6] and PAST [7] which places replica on the numerically closest nodes based on the hash values of the nodes and data.

Statistics-based Statistics-based schemes select the node to place replica based on the statistic results generated from various perspectives. Google File System (GFS) [8] considers disk capacity utilization, client traffic and load distribution. In [9] *Peer Availability* Table (PAT) is maintained to monitor the peers' availability over time. The combination nodes which yield the highest data availability are selected to store the replica. Tung et al. [10] propose a swap-based replication scheme that takes data popularity and bandwidth capability into account. [2] proposed a self-stabilizing algorithm which periodically checks the load distribution: one of multiple replicas on a node will be shifted to its direct neighbor node if it can minimize the variance of the number of replicas on each node.

Deterministic declustering Declustering technique stripes data over multiple disks to render parallel processing and load balancing. Deterministic declustering, as defined in [11], calculates the node location of each block based on a mathematical function. Well-known declustering schemes include interleaved declustering [12, 13], chained declustering [14], group-rotate declustering [15] and shifted declustering [16].

Studies have shown that the choice of the replica placement policy and the inaccuracy of the selected parameter values have a significant impact on the overall system results. A simple yet efficient method is needed and becomes the motivation behind our work.

In our method, we use an analytic model and a combination of three quantitative metrics: workload

distribution, the maximum number of simultaneous disk failures the system can sustain without data loss, and the probability of no-data-loss on a given number of disks. The latter two are intuitive reliability metrics. The reason we involves the load distribution is that an optimized distribution can result in maximized system throughput and thus improved performance, and reduced recovery time in degraded modes and thus enhanced reliability.

Using this method, we evaluate deterministic declustering rather than hashing-based or statistics-based algorithms due to three reasons. First, hashing-based and statistics-based algorithms organize storage nodes in a pseudo-random manner, and random placement tends to cause data loss from multiple and concurrent random node failures. Our method, however, takes block permutation into account as described in section II (B), and can distinguish policies on any given number of disk failures. Second, in large-scale systems, declustering gains much attention by yielding good performance for system in degraded modes and enhanced reliability by significantly reducing data recovery time [17]. Third, with the increasing volume of data, deterministic declustering pre-defines the replica layout and thus eliminates the problems of memory and bandwidth consumption caused by maintaining growing node-block mapping tables (for hashing-based algorithm) or calculating block location on-the-fly (for statistics-based algorithm) [11, 18]. Four deterministic declustering configurations are under consideration: basic mirroring, interleaved declustering, chained declustering and group-rotate declustering.

The remainder of paper is organized as follows. In Section II we present the analytic model for the study of load balancing and reliability, and describe deterministic declustering configurations. In Section III and IV we describe the quantitative results. Conclusions and future work are given in Section V.

II. MODEL AND SYSTEM DESCRIPTIONS

This section presents the model of the distributed storage system, specifies the parameters used in analysis, and describes various declustering configurations with a fixed distributed storage framework.

A. Model Definition

A system with N disks is under consideration. The disks are further divided into one or more non-overlapped configurations concerning clusters with the unit size n, and the number of clusters $c = N/n$. To begin with, we analyze the system with two physical copies of data maintained: primary data copy *X* and secondary data copy X′ (also stands for a "backup copy" in this study). In normal mode, X′and X are accessed with a certain distribution ratio α. The arrival rates of read requests and write requests are λ_R and λ_W respectively. Total load of each disk is denoted as λ_{disk}, which consists of the load of read requests at each disk, denoted as Λ_R , and the load of write requests at each disk, denoted as Λ_w, i.e., $\lambda_{disk} = \Lambda_R + \Lambda_w$.

B. Various Replica Declustering Configurations

Basic mirroring: Disk 2i mirrors disk 2i - 1, $1 \le i \le n$ as shown in Figure 1. The Read load is assumed to be evenly distributed to either disk, i.e., $\alpha = 1/2$.

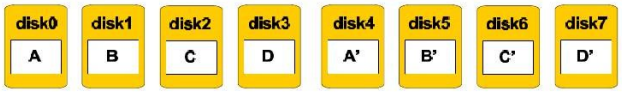

Figure 1. Basic mirroring.

Interleaved declustering [12, 13]: The primary and secondary copies of data are stored within an n-disk ($n \ge 2$) cluster. The secondary data copy is evenly striped across all the remaining disks in the cluster. Figure 2 illustrates the case for n=8. α of Reads are processed at the disk which maintains the primary data copy, and (1-α) of Reads are evenly routed to the other disks in the cluster.

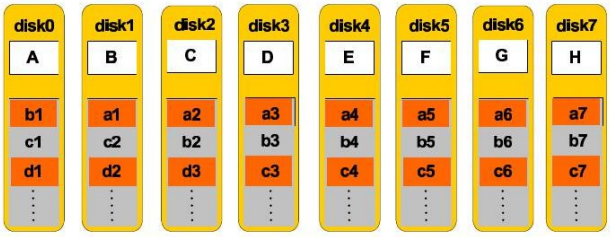

Figure 2. Interleaved declustering, $n = 8$.

Chained declustering [14]: As shown in Figure 3, the data on each disk is replicated on the next disk (modulo the number of disks). In normal mode, Reads are routed to the primary data copy and write operations update both copies.

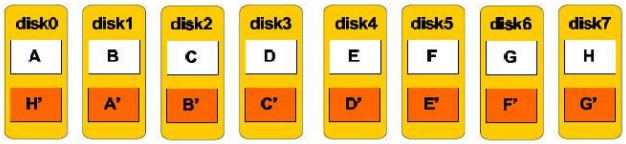

Figure 3. Chained declustering

Group-rotate declustering [15]: The primary and secondary data copies are maintained in separated clusters of disks, as shown in Figure 4. The primary data copy is striped across n disk in a cluster, and these striped data are duplicated as the secondary data copy and stored in another n-disk cluster in a rotated manner.

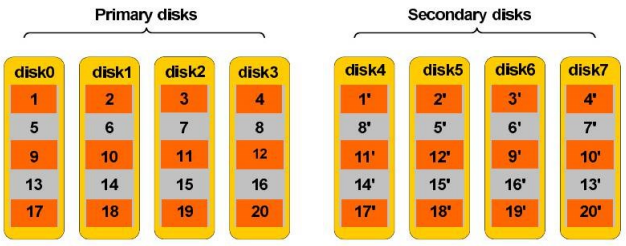

Figure 4. Group-Rotate declustering.

X = Primary data, X'= Duplicated data.

III. LOAD DISTRIBUTION COMPARISON

The load distribution comparison is carried out for the variations with the following assumptions: (i) each disk maintains both primary and secondary data copies, unless otherwise noted; (ii) n is half of the total number of disks N in group-rotate declustering due to the fact that primary and secondary data copies are placed in separate clusters, and equals N in other two variations.

A. Basic Mirroring

Normal Mode

Referring to Figure 1, the load of each disk comprises half of the Read load and the complete Write load:

$$\lambda_{\text{disk}} = \left(\frac{\lambda_R}{2} + \frac{\lambda_R}{2}\right) + (\lambda_W + \lambda_W) = \lambda_R + 2\lambda_W. \quad (1)$$

Degraded Mode

Only for the surviving disk corresponding to the failed one, the Read load is doubled and Write load remains:

$$\lambda_{\text{disk}} = 2\lambda_R + 2\lambda_W \quad (2)$$

The load of the disks in other mirrored pairs remains unchanged, and hence results in unbalanced workloads across all of the remaining N - 1 disks.

B. Interleaved Declustering

Normal Mode

On the assumption that α of Reads access primary data copy (see Figure 2), the Read load of each disk Λ_R comprises two parts: the access to the primary data; and the accesses to the secondary data that corresponds to the primary data stored on the other disks within the same cluster, which is $\alpha\lambda_R + (n-1)\frac{(1-\alpha)\lambda_R}{n-1} = \lambda_R$. Similarly, the Write load of each disk Λ_W is $\lambda_W + (n-1)\frac{\lambda_W}{n-1}$.

The total load of each disk is the sum of the above two parts:

$$\lambda_{\text{disk}} = \Lambda_R + \Lambda_W = \lambda_R + 2\lambda_W \quad (3)$$

Degraded Mode

Interleaved declustering allows only one disk failure within each cluster. For each of the surviving disks, the Read load comes from four sources: (i) accesses to the secondary data corresponding to the failed disk: $\lambda_R/(n-1)$ (ii) accesses to the primary data: $\alpha\lambda_R$ (iii) accesses that cannot be routed to the secondary data copy on the failed disk: $(1-\alpha)\lambda_R/(n-1)$ (iv) accesses routed from other surviving disks: $(n-2)\frac{(1-\alpha)\lambda_R}{n-1}$. The total Read load of each disk Λ_R is the sum of the above four, which yields $\frac{n}{n-1}\lambda_R$. The Write load of each disk Λ_W is: $\lambda_W + (n-2)\frac{\lambda_W}{n-2} = 2\lambda_W$.

The total load of each surviving disk within a cluster is:

$$\lambda_{\text{disk}} = \Lambda_R + \Lambda_W = \frac{n}{n-1}\lambda_R + 2\lambda_W. \quad (4)$$

C. Chained Declustering

Normal Mode

In normal mode, Reads are routed to the disk that holds primary data, and Writes are required to update both data copies. The load of each disk is:

$$\lambda_{\text{disk}} = \Lambda_R + \Lambda_W = \lambda_R + 2\lambda_W. \quad (5)$$

Degraded Mode

As shown in Figure 3, chained declustering simply backups the data and stores it on the disk next to which the primary data is stored. Therefore, in the degraded mode, a disk failure does not cause the change in Write load of surviving disks. For simplification, a read-only workload is assumed in the analysis. Denote λ_i to be the load of disk_i in normal mode, and λ_i' in degrade mode. Assume the load of each disk is identical: $\lambda_0 = \lambda_1 = \lambda_2 = \cdots = \lambda_{n-1}$.

Chained declustering does not tolerate contiguous disk failures, and the maximum number of failed disks, which will not cause data loss, is $\lfloor n/2 \rfloor$. Load balancing is not achievable in all the cases when system sustains in degraded mode. The case-by-case discussions are as follows:

Single disk failure: Load balancing is easily achieved across all the surviving disks. Each of the disks serves N/(N − 1)(N = n in this case) of the total load, so that the load of disk_i is:

$$\lambda_i = (\Lambda_R)_i = \frac{n-i}{n-1}\lambda_{i-1} + \frac{i}{n-1}\lambda_i \quad (6)$$

Double disk failures:

Due to the fact that two consecutive disk failures will cause data loss, the analysis assumes disk_0 and disk_k fail, where $k \neq 1$ and $n-1$.

However, load balancing cannot be achieved in all the cases without data loss. For example, if $\text{disk}_0$and disk_2 fail, the load of disk_1 will be doubled while not necessary for other surviving disks. This is because disk_1 maintains the data copies B and A', both of whose counterparts are on failed disks.

Load balancing with two disk failures occurs only when each of the remaining disks serve $n/(n-2)$ of the load: $\lambda_1' = \lambda_0 + \frac{2}{n-2}\lambda_1$, $\lambda_2' = \frac{n-4}{n-2}\lambda_1 + \frac{4}{n-2}\lambda_2, \ldots$; however, because disk_k fails and consequently can not take over the load of disk_{k-1}, disk_{k-1} services all the requests accessing its primary data:$\lambda_{k-1} = \frac{2}{n-2}\lambda_{k-2} + \frac{n-2}{n-2}\lambda_{k-1}$. In other words, load balancing is achieved only when the surviving disks between disk_0 and disk_kcan accomplish the load of k disks: $\lambda = (k-1)\frac{n}{n-2}$, and thus $k = n/2$.

In such a load-balancing case, the load of each surviving disk is $\lambda_{disk} = \Lambda_R = \frac{n}{n-2}\lambda_0$; the load of $disk_i$ $(0 \le i \le k)$ is:

$$\lambda_i = (\Lambda_R)_i = \frac{n-2i}{n-2}\lambda_{i-1} + \frac{2i}{n-2}\lambda_i. \quad (7)$$

Multiple disk failures:

For multiple k disk failures, $k \ge 2$, a conclusion can be easily derived from the above analysis that, load balancing is achieved only when $k = 2^m, (m \ge 1)$. For the example shown in Figure 3, the extreme load-balancing case is four disk failures, in which the load of each surviving disk doubled.

D. Group-rotate Declustering

Normal Mode

Assume α of the read requests are routed to the primary disks. Similar to interleaved declustering, $\Lambda_R = \alpha\lambda_R + (1-\alpha)\lambda_R = \lambda_R$. The Writes to each disk update both the primary and secondary data copies: $\Lambda_W = \lambda_W + \lambda_W = 2\lambda_W$. The total load of each disk is:

$$\lambda_{disk} = \Lambda_R + \Lambda_W = \lambda_R + 2\lambda_W. \quad (8)$$

Degraded Mode

Group-rotate declustering sustains multiple disk failures only when the failed disks are in one cluster, i.e., either primary disks or secondary disks, but not both. In the analysis, disk failure is assumed to occur at the primary disks.

In order to achieve load balancing across the surviving disks, the load proportion between primary disks and secondary disks, α, is varied along with the number of failed disks. To simplify the calculation, assume each cluster maintains only one type of data copy, i.e., either primary data copy or secondary data copy. A cluster of disks that maintain the primary/secondary data copy are denoted as primary/secondary disks, as shown in Figure 4. The correctness of the assumption is accomplished by the fact that the two clusters are symmetric to each other. The analyses, based on the system with $n = \frac{1}{2}N$, are as follows:

Single disk failure

The load of failed disk is evenly distributed to all the secondary disks. For each secondary disk, the Read load includes the accesses corresponding to the surviving $n-1$ primary disks, and $1/n$ of the accesses from failed disk: $\Lambda_R = (n-1)\frac{(1-\alpha)\lambda_R}{n} + \frac{\alpha}{n}\lambda_R = \lambda_R(1-\alpha+\frac{\alpha}{n})$.

To achieve load balancing, the load between the primary and secondary disks must be equal: $\alpha\lambda_R + \lambda_W = \lambda_R\left(1-\alpha+\frac{\alpha}{n}\right) + \lambda_W$, and thus $\alpha = n/(2n-1)$.

To achieve load balancing, the load between the primary and secondary disks must be equal: $\alpha\lambda_R + \lambda_W = \lambda_R\left(1-\alpha+\frac{\alpha}{n}\right) + \lambda_W$, and thus $\alpha = n/(2n-1)$.

The load of each surviving disk is:

$$\lambda_{disk} = \frac{n}{2n-1}\lambda_R + \lambda_W \quad (9)$$

Multiple disk failures

Similarly, to balance the load between the surviving primary disks and secondary disks, α is adjusted according to $\alpha\lambda_R + \lambda_W = (n-k)\frac{(1-\alpha)\lambda_R}{n} + \frac{k\lambda_R}{n} + \lambda_W, (k \le n)$ and thus $\alpha = n/(2n-k)$.

The load of each surviving disk is:

$$\lambda_{disk} = \frac{n}{2n-k}\lambda_R + \lambda_W \quad (10)$$

In an extreme case in which $k = n$, the Read load of each secondary disk doubled.

IV. RELIABILITY COMPARISON

This section estimates the reliability of various declustering configurations by using the probability of no-data-loss, denoted by P_k, for each level in degraded mode based on the system described in Section III.

A. Basic Mirroring

Basic mirroring can tolerate up to $N/2$ disk failures, as long as the failed disks are in different mirrored pairs.

In general, there are $\binom{N}{k}$ ways for k disk failures; the number of ways that k disk failures do not cause data loss is $\binom{\frac{N}{2}}{k}2^k$. Thus the reliability of basic mirroring is:

$$P_k = \frac{\binom{\frac{N}{2}}{k}2^k}{\binom{N}{k}} \quad (11)$$

B. Interleaved Declustering

Interleaved declustering distributes the load of a failed disk evenly to the remaining disks within a cluster. So each cluster can tolerate up to one disk failure, and its probability is $P_1 = 1$.

When $k \le c$ disks fail, data loss does not occur as long as the failed disks are in different clusters. There are $\binom{N}{k}$ ways for k disk failures; the number of possibilities of no data loss is $(N-n)\cdot(N-2n)\cdots(N-(k-1)n)$. Thus

$$P_k = \frac{N(N-n)\cdot(N-2n)\cdots{}^{(N-(k-1)n)}/_{k!}}{\binom{N}{k}} = \frac{n^k\frac{N}{n}\cdot(\frac{N}{n}-1)\cdot(\frac{N}{n}-2)\cdots(\frac{N}{n}-(k-1))/k!}{\binom{N}{k}} = \frac{n^k c\cdot(\frac{N}{n}-1)\cdot(c-2)\cdots(c-(k-1))/k!}{\binom{N}{k}} = \frac{\binom{c}{k}n^k}{\binom{N}{k}}. \quad (12)$$

C. Chained Declustering

As described in Section III(C), chained declustering can still tolerate one disk failure, so the probability of no-data-loss $P_1 = 1$; chained declustering system can sustain up to $\lfloor 2/n \rfloor$ ($n = N$ in this case) failures.

The probability of k disk failures without data loss will be calculated for each case.

Single disk failure:

$$P_1 = 1 \tag{13}$$

Double disk failures:

There are $\binom{n}{2}$ ways for two disk failures; the number of ways that two contiguous disks can fail is n. Thus

$$P_2 = 1 - \frac{n}{\binom{n}{2}} \tag{14}$$

Three disk failures: There are $\binom{n}{3}$ possible ways for three disk failures. Any two contiguous disk failures would result in data loss, which includes two possible cases: three failed disks are consecutive; two failed disks are consecutive, but the third one is one or more surviving disks apart.

The number of ways that three contiguous disks fail is n; the number of ways that two contiguous disks fail excluding three consecutive failures is n(n−2−2), which is elaborated as follows: the number of two contiguous disk failures is n; the third failed disk is not possible adjacent to those two, so that the third failure is only possible to occur on the remaining n–2−2 disks. Thus

$$p_3 = \frac{\binom{n}{3} - n - n(n-4)}{\binom{n}{3}} \tag{15}$$

Four disk failures:

Similar to the above analysis

$$P_4 = \frac{\binom{n}{4} - n - n(n-3-2) - n\binom{n-2-2}{2} + \frac{n(n-4-1)}{2}}{\binom{n}{4}} \tag{16}$$

D. Group-rotate Declustering

As explained in Section III(D), group-rotate declustering tolerates multiple disk failures, as long as the failed disks are within one cluster, i.e. either primary disks or secondary disks. Given the assumption of disk failure(s) occurring on primary disk(s), the reliability analysis is performed as follows.

The probability that single disk failure does not incur data loss $P_1 = 1$. When $k(k \geq 2)$ disks fail, there are $\binom{N}{k}$ ways for this to happen; while there are only $\binom{n}{k}$ out of them that k disks fail within one cluster, which does not incur data loss. There are $c(c = N/n)$ clusters in total, so the number of ways that k disks can fail within a cluster and hence without data loss is $c\binom{n}{k}$. Thus

$$P_k = \begin{cases} 1 & k = 1 \\ \frac{N\binom{n}{k}}{\binom{N}{k}} = \frac{c\binom{n}{k}}{\binom{N}{k}} & 1 < k \leq n. \end{cases} \tag{17}$$

V. CONCLUSIONS AND FUTURE WORK

It is widely acknowledged that replica allocation significantly impacts the reliability and performance of distributed storage systems. However, their flexible nature brings a wide range of choices for replica placement scheme and parameter values, and thus makes the overall system behavior difficult to predict. In this paper, we propose a simple method to evaluate replica placement schemes by using an analytic model and a combination of three quantitative metrics. The results are summarized as in Table I.

TABLE I. ABILITY OF LOAD BALANCING AND RELIABILITY COMPARISON

Declustering type	**Basic**	**Interleaved**	**Chained**	**Group-Rotate**
Load balanced after one disk failure	✗	✔	✔	✔
Ease of balance	Not applicable	Easy	Limited	Easy
Maximum number of failed disks	N/2	1 per cluster	N/2	N/2
Prob[k disk failures without data loss]	Equation 11	Equation 12	Equations 13-16	Equation 17

As future work, we would like to conduct experiments in large scale systems, comparing the results against simulation results to validate the accuracy of this method. Developing a weighted metrics vector to evaluate performance and reliability, which can be served as a fast selection tool for the policies and their parameters, will also be a valuable topic.

REFERENCES

[1] Q. Lian, W. Chen, and Z. Zhang. “On the impact of replica placement to the reliability of distributed brick storage systems,” 2013 IEEE 33rd International Conference on Distributed Computing Systems, 00:187–196, 2005.

[2] S. KÖhler and V. Turau. “Self-stabilizing local k -placement of replicas with local minimum variance,” Elsevier Science Publishers Ltd., 2015.

[3] T. Zhang. “Data replica placement in cloud storage system,” CCIS-13, 2013.

[4] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, and C. Oceanstore, “an architecture for global-scale persistent storage,” Acm Sigplan Notices, 34(5):190–201, 2000.

[5] Y. Chen, R. H. Katz, and J. D. Kubiatowicz. D. Replica, “Placement for Scalable Content Delivery,” Springer Berlin Heidelberg, 2002.

[6] C. Greg Plaxton, R. Rajaraman, A. Richa, and a W, “Accessing nearby copies of replicated objects in a distributed environment,” In ACM SPAA, pages 311–320, 1999.

[7] P. Druschel and A. Rowstron. Past, “a large-scale, persistent peer-to-peer storage utility,” In Proceedings Eighth Workshop on Hot Topics in Operating Systems, pages 75–80, May 2001.

[8] S. Ghemawat, H. Gobioff, and S. Tak Leung, “The google file system,” Acm Sigops Operating Systems Review, 37(5):29–43, 2003.

[9] G. Song, S. Kim, and D. Seo, “Replica placement algorithm for highly available peer-to-peer storage systems,” In 2009 First International Conference on Advances in P2P Systems, pages 160–167, Oct 2009.

[10] Y. C. Tung, C. J. Lin, and C. F. Chou, “Bandwidth-aware replica placement for peer-to-peer storage systems,” In 2011 IEEE Global Telecommunications Conference-GLOBECOM 2011, pages 1–5, Dec 2011.

[11] X. Zhang, J. Yin, J. Wang, R. Wang, and D. Huang, “Deister: A light-weight autonomous block management in data-intensive file systems using deterministic declustering distribution,” In IEEE International Conference on Smart City/socialcom/sustaincom, pages 598–604, 2015.

[12] DBC/1012 Database Computer System Manual. Release 2. Teradata Corporation, Nov 1985.

[13] G. P. Copeland and T. Keller, “A comparison of highavailability media recovery techniques,” In James Clifford, Bruce G. Lindsay, and David Maier, editors, Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, Oregon, May 31 - June 2, 1989, pages 98–109. ACM Press, 1989.

[14] H. Hsiao and D. J. DeWitt, “Chained Declustering: A new availability strategy for multiprocessor database machines,” In Proceedings of 6th International Data Engineering Conference, pages 456–465, 1990.

[15] S. Chen and D. Towsley, “A performance evaluation of RAID architectures,” IEEE Transactions on Computers, 45(10):1116–1130, 1996.

[16] H. Zhu, P. Gu, and J. Wang, “Shifted declustering: a placement-ideal layout scheme for multi-way replication storage architecture,” Proceedings of Ics’, pages 134–144, 2008.

[17] X. Qin, E. L. Miller, and T. J. E. Schwarz, “Evaluation of distributed recovery in large-scale storage systems,” In Proceedings. 13th IEEE International Symposium on High performance Distributed Computing, 2004, pages 172–181, June 2004.

[18] J. Wang, H. Wu, and R. Wang, “A new reliability model in replication-based big data storage systems,” Journal of Parallel Distributed Computing, 2017.