Shifted Declustering: A Placement-ideal Layout Scheme for Multi-way Replication Storage Architecture

Huijun Zhu
School of EECS, University of Central Florida
Orlando, FL 32816 USA
zhuhj@cs.ucf.edu

Peng Gu
School of EECS, University of Central Florida
Orlando, FL 32816 USA
penggu@cs.ucf.edu

Jun Wang
School of EECS, University of Central Florida
Orlando, FL 32816 USA
jwang@cs.ucf.edu

ABSTRACT

Recent years have seen a growing interest in the deployment of sophisticated replication based storage architecture in data-intensive computing. Existing placement-ideal data layout solutions place an emphasis on declustered parity based storage. However, there exist major differences between parity and replication architectures, especially in data layouts. We retrofit the desirable properties of optimal parallelism in parity architectures for replication architectures, and propose a novel placement-ideal data layout — shifted declustering for replication based storage. Shifted declustering layout obtains optimal parallelism in a wide range of configurations, and obtains optimal high performance and load balancing in both fault-free and degraded modes. Our theoretical proofs and comprehensive simulation results show that shifted declustering is superiour in performance and load balancing to traditional replication layout schemes such as standard mirroring, chained declustering, group rotational declustering and existing parity layout schemes PRIME and RELPR in reference [4].

Categories and Subject Descriptors

B.4.3 [**Input/Output and Data Communications**]: Interconnections(Subsystems)—*Parallel I/O*; D.4.5 [**Operating Systems**]: Reliability—*Faut-tolerance*

General Terms

Design, Performance, Reliability

Keywords

Multi-way replication, data layout, parallel I/O

1. INTRODUCTION

With the introduction of blade servers and the emergence of techniques for spreading applications over clusters to exploit scalability, recent years have seen large-scale data clusters utilizing hundreds of thousands of hard drives [20, 5]. Because disk failures may cause the loss of valuable data, the direct and indirect costs associated with the failures become increasingly important in the deployment and operation of the data clusters. To attack the problem, more and more clusters are turning to multi-way replication based storage architecture due to the requirement of high availability. Fortunately, recently the capacity of a single magnetic disk has reached 1 TB [3], so storage efficiency becomes less considered for building storage cluster than before. Hence, multi-way replication is a more attractive candidate than parity for redundant storage. For example, Google file system uses a three-way replication storage organization [10] by default, and Amazon's Dynamo wide-area storage system employs a customized multi-way replication storage architecture [9].

With the advent of parity or more sophisticated type of parity-based redundancy, replication has come to be considered the simplest case, and thus has not attracted specific research attention in the past decade. However, a new observation is that, replication is a special case of erasure codes [17, 14] just in respect of data coding. For data layout, replication has its own properties that does not fall into the category of parity-based redundancy schemes. More specifically, there are several major limitations in current data layout solutions:

•**Restricted configurations of ideal layouts:** Some researchers have adopted solutions using BIBD (Balanced Incomplete Block Design) [15] theories. Unfortunately, only a small portion of layouts whose parameters agree with *symmetric* BIBD can meet the placement-ideal requirements [4]. On the other hand, previous work developed near-optimal parallelism layout solutions such as declustered parity organizations for small-scale storage architectures [11, 4], but these schemes do not or not always satisfy all of the desirable properties of an ideal layout. In addition, several replication-declustered layouts were developed for RAID-1 [16] systems, such as Interleaved declustering [8], Group-rotate declustering [7] and Chained declustering [12]. All these existed solutions violates the properties of placement-idealness to some extent, and they are explained in Section 2.3.

•**Limited optimal parallelism studies on multi-way replication disk architecture:** Prior studies emphasize exploitation of parallelism for parity disk array architectures [11, 4]. For example, Holland and Gibson [11] solve the placement-ideal layout problem to tolerate one disk failure. Alvarez *et al.* [4] present a set of complete, constructive declustered layouts with near-optimal parallelism that tolerate multiple disk failures for parity-based architectures in a wide range of configuration. Meanwhile, multi-way replica-

ICS'08, June 7–12, 2008, Island of Kos, Aegean Sea, Greece.

tion architecture is often treated as a special case of parity architectures. Therefore, it does not attract enough interests. However, there exist important differences between parity and replication architectures, especially in terms of data layout: *i)* checksum units in parity architectures cannot be directly used to service read requests without data units, while both data units and replica units in replication architectures can service read requests individually; *ii)* writing a data unit in parity architectures leads to an update on its corresponding checksum unit, while writing a data unit leads to updating all other replica units in replication architectures; *iii)* it is write expensive to maintain checksum units rather than simple replica units; *iv)* there are different parallelism exploitation perspectives to perform reconstruction and recovery between two architectures.

This paper is organized as follows: in Section 2, we retrofit six desirable properties of placement-ideal layouts for multi-way replication storage architectures, and analyze the limitations of existing solutions. In Section 3, we develop a novel placement-ideal data layout scheme named *shifted-declustering* by leveraging chained-declustering for RAID-1 layout to a more general case. Our comprehensive simulation experiments given in Section 4 show that the shifted-declustering architecture obtains wide variety of large-scale configurations, and consistently outperforms existing replication and parity schemes under both normal and degraded modes. Finally, we give the conclusion in Section 5.

2. BACKGROUND

2.1 Properties of Placement-ideal Layouts

Holland *et al.* [11] originally defined six desirable properties of declustered disk array layouts that can tolerate a single disk failure. Alvarez *et al.* [4] proved that six desirable properties can only be satisfied by a small portion of redundancy configurations. However, we noticed that the properties were introduced for parity-based redundancy architectures. For replication-based redundancy, these properties do not properly describe a placement-ideal requirement. We revise the desirable properties for a placement-ideal layout in multi-way replication based architecture. We define a redundancy group as the set of a data unit and all its replicas hereinafter.

Property 1 *Multiple Failures Correcting.*
A k-way ($k \geq 2$) replication architecture is able to provide $(k-1)$ failures correction. This requires that no two units within a redundancy group are mapped to the same disk.

Property 2 *Distributed Replica Information.*
If we distinguish data between primary copies and secondary copies (replicas), each disk should hold the same number of replicas to satisfy this property. However, if the copies are equally important, for example, they are accessed in a round robin manner, this property is satisfied by nature.

Property 3 *Distributed Reconstruction.*
If a unit is mapped to a disk, we say that the redundancy group containing this unit is mapped to this disk. If this property is satisfied, there is a constant number of redundancy groups mapped to any two disks, such that if a disk fails, the workload that is supposed to be handled by the failed disk will be dispatched evenly to all the surviving disks.

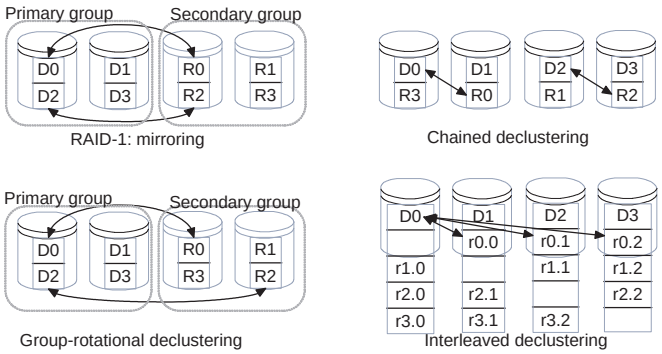

Figure 1: Two-way replication data layouts

Property 4 *Large Write Optimization.*
In parity-based architecture, this property means writing a large number of continuous units without pre-reading units for updating checksum information. However, for replication, any writing process incurs updating all replicas, but no pre-reading is needed. Therefore, this property is satisfied as long as the architecture is replication-based.

Property 5 *Maximal Parallelism*
A layout has maximal parallelism if the addresses of data in any interval $\{t, t+1, \ldots, t+n-1\}$ of n consecutive addresses are mapped to n different disks.

Property 6 *Efficient Mapping*
The functions that map client addresses to disk system locations are efficiently computable.

As far as a replication-based redundant architecture is concerned, property 4 is satisfied naturally. If primary and secondary copies are not distinguished, property 2 is also met. In the following sections, we focus on properties 1, 3 and 5.

2.2 Current Solutions for Replication-based Architecture

RAID-1 is a major building block in large-scale storage systems with two-way replication redundancy. It adopts the simplest data layout—mirroring, in which one disk has an identical twin with exactly the same data. In a multi-way replication domain, we refer a mirroring replication scheme as standard mirroring. To improve service performance and load balancing, several replication declustering schemes were developed. To the best of our knowledge, representative declustering strategies include chained declustering [12], group-rotational declustering [7], and interleaved declustering [8]. Figure 1 illustrates these layouts.

Chained declustering was developed as a high-availability data replication scheme for databases [12]. It distributes the primary copies of data over disks like a chain, and the secondary copy of one data unit is dispatched to the neighboring disk of the primary copy.

Group rotational declustering was first introduced to improve the performance of standard mirroring, and then extended to multi-way replication for high throughput media server systems [6]. It partitions all disks into several groups, and the number of groups equals to the number of data copies. Each group stores a complete copy of all data. Compared to standard mirroring, the data in the secondary group is distributed in a rotational way rather than standard mirroring.

Interleaved declustering first distributes all primary copies over all disks. After that, given a specific primary copy, it splits the secondary copy into several pieces and stores them evenly over disks that do not carry the primary copy.

In respect of data coding, replication-based redundancy can be viewed as a special case of parity-based redundancy, so that parity-based layouts could be adopted for replication based layouts. Some researchers have adopted solutions based on BIBD [15] theories. Alvarez *et al.* [4] proposed PRIME and RELPR as near-optimal layouts for general redundancy configurations with any parameters.

2.3 Comparison among Layout Schemes

In this section, we analyze the satisfiability of standard mirroring, chained declustering, interleaved declustering, group-rotational declustering, RELPR and PRIME in terms of meeting the properties defined in Section 2.1. Assume that the number of copies is k, and the number of disks is n, $k \leq n$. We also assume that the replicas are equivalent, so we just concentrate on properties 1, 3, and 5. A comparison summary is given in Table 1.

2.3.1 *Multiple Failures*

This property is satisfied as long as no two replicas of the same data are located in the same disk. This is the basic requirement of redundant storage architecture, and all layout schemes satisfy this property.

2.3.2 *Distributed Reconstruction*

If this property is satisfied, the requests that are supposed to be serviced by one failed disk are redirected to all other disks evenly, and this guarantees load balance of service under degraded mode. Ideally, each surviving disk should handle $1/(n-1)$ of the workload redirected from the failed disk. In non-ideal layouts, the workload of a failed disk is not distributed equally among all surviving disks. We evaluate this property using a *reconstruction workload bound* metric, which denotes the maximum workload fraction of a failed disk for which a surviving disk is responsible. If the reconstruction workload bound is larger than $1/(n-1)$, the layout does not satisfy this property.

In **standard mirroring**, each disk is mirrored to $k-1$ other disks, so if one disk fails, only $k-1$ disks can carry the load of the failed disk. In other words, each of these $k-1$ disks should carry $1/(k-1)$ workload of the failed disk, so the reconstruction workload is bounded by $1/(k-1)$. If $k < n$, $1/(k-1) > 1/(n-1)$. Therefore standard mirroring does not satisfy this property if $k < n$.

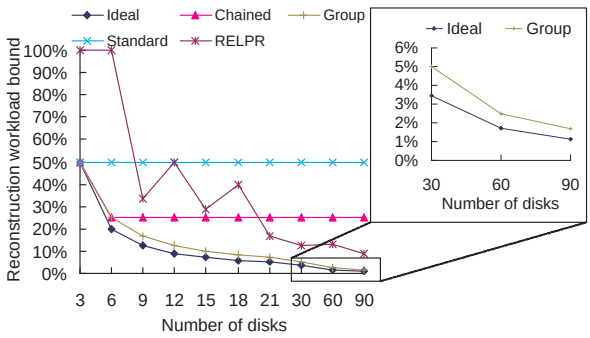

Figure 2: Rconstruction workload bound

In **chained declustering**, a disk shares more redundancy groups with neighboring disks, but shares less or no redundancy groups with distant disks. Therefore, if one disk fails, only $2(k-1)$ disks share the workload of the failed disk, so the reconstruction workload is bounded by $2/(k-1)$. If $k < 2n-1$, $2/(k-1) > 1/(n-1)$. Hence chained declustering does not satisfy this property if $k < 2n-1$.

In **group-rotational declustering**, the workload of a failed disk is shared by disks outside the group of the failed disk, and the number of disks that can absorb the workload is $n(k-1)/k$, so the reconstruction workload is bounded by $k/n(k-1)$. For $k < n$, $n(k-1)/k > 1/(n-1)$. Therefore group-rotational declustering does not satisfy this property if $k < n$.

In **RELPR**, the reconstruction workload is bounded by $(k-1)/\phi(n)$, where $\phi(n)$ counts the number of positive integers less than n that are relatively prime to n [4]. In **PRIME**, since the number of disks is prime, $\phi(n) = n-1$, and each surviving disk carries $(k-1)/(n-1)$ reconstruction workload [4]. For $k > 1 + (\phi(n)/(n-1))$, $(k-1)/\phi(n) > 1/(n-1)$, and $\phi(n) \leq (n-1)$, $1+(\phi(n)/(n-1)) \leq 2$. Hence, RELPR and PRIME do not satisfy this property if $k \geq 2$.

In summary, standard mirroring, chained declustering, group rotational declustering, RELPR and PRIME do not satisfy this property for arbitary values of k and n.

Interleaved declustering meets this property for a two-way replication architecture. If one disk fails, all the surviving disks share its workload evenly. Each surviving disk is responsible for $1/(n-1)$ of its workload. However, two major limitations exsit. First, because it dispatches data by dividing them into small pieces, there exists non-negligible overhead resulting from composing multiple data pieces into a complete data unit. Second, to scale up to a three-way replication, it is hard to place the third copy without violating a basic rule of replica distribution. For any given data unit, all disks have either a primary copy or part of a second copy. No matter where to place the third copy, it violates the basic rule in which no two copies are distributed to the same disk. Therefore, it is difficult to scale interleaved declustering up to three-way or higher replication architecture while still satisfy these properties.

Figure 2 demonstrates the *reconstruction workload bound* which is redirected to a surviving disk, in a **three-way** replication configuration. We analyze the reconstruction ability up to 90 disks, since there have already been central controlled disk arrays with 45 drives (e. g. Dell PowerVault MD 1000 disk expansion enclosure [2]) in use. The ideal reconstruction workload is shown in the bottom curve in Figure 2. The reconstruction workload bound is 50% and 25% in standard mirroring and chained declustering, respectively. In other words, two disks in standard mirroring and four disks in chained declustering would shoulder a heavy workload as a result of the redirected load from the failed disk, and result in a service bottleneck under degraded mode. The reconstruction workload bound in group-rotational declustering decreases with the number of disks, but there is still a good portion of disks which do not take on the failed disk's workload. For example, suppose there are 90 disks, the value of "ideal" is 1/89 (1.1%), and the value of "group-rotational" is 1/60 (1.7%). Although the curves look close in the graph, the reconstruction load based on group-rotational layout is 54% more than that in the ideal case. The fluctuation of reconstruction workload in RELPR is because the number

Table 1: Comparison among layout schemes

	Multiple Failures Correcting	Distributed Reconstruction	Maximal Parallelism	Comment
Standard	✓	×	✓	
Chained	✓	×	✓	
Group-rotational	✓	×	✓	
Interleaved	✓	✓	✓	Only for 2-way replication
RELPR	✓	×	×	
PRIME	✓	✓	×	Only for a prime number of disks

of relative primes to the number of disks. As seen in the figure, it is far from an ideal reconstruction workload.

2.3.3 Maximal Parallelism

This property requires that if the size of a request is n units, all n disks in the system should be accessed, and each disk provides exactly one unit. A metric named *parallel read count* was introduced to evaluate this property [4]. The function of parallel read count $\tau(o)$ is defined by the maximum number of data units that any disk must supply when reading any o consecutive units of client data. A layout is parallel optimal if $\tau(o) = \lceil o/n \rceil$ for all such o [4]. Chained declustering meets this property. In standard mirroring and group-rotational declustering, they could satisfy this property if request dispatch algorithms access all disks. PRIME and RELPR obain parallel read count of $\tau(o) = \lceil o/n \rceil + 1$ or $\tau(o) = \lceil o/n \rceil$ [4], which deos not always dissatisfy the property.

3. SHIFTED DECLUSTERING

We develop a new placement-ideal layout named shifted-declustering that satisfies all six properties and can be deployed in a wide variety of k-way replication configurations. In previous data layout schemes for replication-based architecture, chained-declustering satisfies all properties except *distributed reconstruction*. More specifically, a pair of neighboring disks share more redundant data than a pair of distant disks. Shifted-declustering is to leverage chained-declustering layout for two-way replication [12] to general cases by incrementing the distance between every two units belonging to the same *redundancy group*, one step per iteration, until all k units belonging to the same redundancy group are distributed over all the disks. The procedure of layout is like shifting each row of the units over all the disks in a circular fashion. In this way, it eliminates the limitation that consecutive disks carry more overlapped redundancy data than distant disks in chained declustering. This renders a satisfaction with *distributed reconstruction*.

Many large-scale storage systems such as super data clusters [9, 10] employ multi-way replication based storage architecture, which is often composed of a cluster of many building blocks that are connected by a high-speed network. Each building block is a central-controlled storage subsystem. Our shifted-declustering scheme works for such type of building-blocks. Some enterprise distributed storage systems, like FAB from HP lab [18] and IceCube from IBM lab [19] manage storage replicas by distributed protocols. The shifted-declustering layout can also be applied if integrated with distributed protocols.

3.1 Definitions and Notations

The notations are summarized in Table 2. There are two parameters for the shifted declustering layout: the number of disk drives n ($n \geq 2$), and the number of units per redundancy group k ($k \leq n$). Within a redundancy group, the units are named from 0 to $k-1$, and we view the unit 0 as the data unit, and other units as replica units. Distinguishing data units from replica units is only for the ease of representation, although they are identical. We mention our replication configuration as k-way replication hereafter. A k-way replication cluster is able to tolerate $k-1$ disk failures simultaneously.

Table 2: Notation summary

System configuration parameters	
n	Number of disks in the cluster
k	Number of units per redundancy group
Parameters used in computation	
a	The address to denote a redundancy group
(a, i)	The i-th unit in redundancy group a
q	Number of iterations of a complete round of layout
y, z	Intermediate auxiliary parameters
Computation output	
$disk(a, i)$	The disk where the unit (a, i) is distributed
$offset(a, i)$	The offset within $disk(a, i)$ where the unit (a, i) is distributed

We use an address a ($a \geq 0$) to denote a redundancy group. Also, we give a unit name (a, i), where $0 \leq i < k$, to the i-th unit in this redundancy group. Hence, a can also be considered as a redundancy group ID. Without the loss of generality, $(a, 0)$ represents the data unit, and (a, i) with $i > 0$ represents the i-th replica unit. The location of the unit (a, i) is represented by a tuple $(disk(a, i), offset(a, i))$. A complete round of layout is obtained by q iterations, and in each iteration, one row of data units and $k-1$ rows of replica units are placed, so that the total rows of units in a complete round is $r = kq$. Repeating complete rounds of layouts also yields placement-ideal layouts. In the following, we only consider the layouts within a complete round.

The most popular configurations of replication based architecture are two-way and three-way replication. A complete solution for 2-way replication is given in [4], and attached in Appendix A. In Section 3.2, we give placement-ideal layouts for 3-way replication.

3.2 Solution for $k = 3$

3.2.1 Basic solution for $n = 4$, or n is odd

In this case, the placement-ideal layout is summarized as equations 1 to 5, where $i = 0, 1, 2$. For the simplicity of representation, we use $x\%y$ to represent the modulo operator x mod y as in the C Programming Language.

$$q = \begin{cases} 1, \text{ if } n = 4 \\ (n-1)/2, \text{ if } n \text{ is odd} \end{cases} \quad (1)$$

$$z = \left\lfloor \frac{a}{n} \right\rfloor \quad (2)$$

$$y = (z\%q) + 1 \quad (3)$$

$$disk(a, i) = (a + iy)\%n \quad (4)$$

$$offset(a, i) = \left\lfloor \frac{a}{n} \right\rfloor + (k-1)z + i = kz + i \quad (5)$$

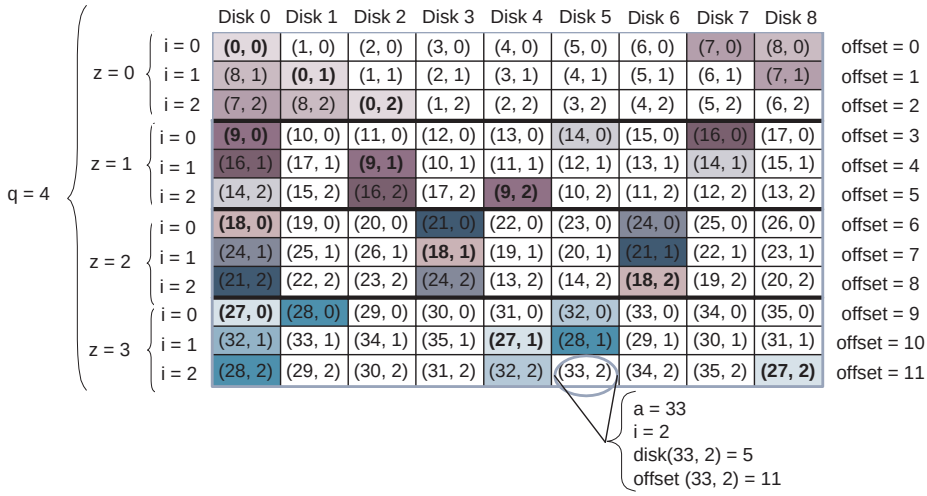

Figure 3: Example layout for $k = 3$, $n = 9$

The placement-idealness of the layout for $k = 3$, $n = 4$ or n is odd has been proved by verifying its satisfaction of the desirable properties, one by one, in Appendix B.1.

Figure 3 gives the layout for $k = 3$ and $n = 9$. In this case, $q = 4$, so there are 4 iterations within a complete round of layout. Each block in the figure indicates a unit, and the unit is represented by its name (a, i).The bold labels illustrate how the units are shifted. In the i-th row of the iteration l, the units are shifted by a distance of $i(l + 1)$ disks right-handedly. The colored blocks represent the redundancy groups mapped to each pairs of disk 0 and another disk. Particularly, the blocks with the same color (shade) belong to the same redundancy group. In this specific case, exactly three redundancy groups are mapped to any pair of disk 0 and another disk. Due to the equivalence between disks, any two disks share three redundancy groups.

3.2.2 *Solution for $n > 4$ and n is even*

For $k = 3$, shifted declustering provides placement-ideal layouts over odd number of disks. This motivates us to find an optimal solution for three-way replication over an even number (larger than 4) of disks. Notice that n is even, then $n - 1$ is odd. Intuitively, we consider to insert a "bubble" on a disk in each iteration, and distribute the units over the remaining $n - 1$ disks based on the layout scheme for odd number of disks. In the next iteration, the bubble is shifted to another disk, and then shifted declustering is applied on the remaining $n - 1$ disks. This procedure can be imagined as stretching the data of $n-1$ disks evenly over n disks. The layout is summarized in equations 6 to 13.

$$n' = n - 1 \quad (6)$$

$$q' = (n' - 1)/2 \quad (7)$$

$$z = \lfloor a/n' \rfloor \quad (8)$$

$$d_b = n' - (z\%n) \quad (9)$$

$$y = (z\%q') + 1 \quad (10)$$

$$disk(a, 0) = a\%n \quad (11)$$

$$disk(a, i) = \begin{cases} (a + iy + 1)\%n^* \\ (a + iy)\%n^{**} \end{cases}, (i \geq 1) \quad (12)$$

$$offset(a, i) = kz + i, (i \geq 0) \quad (13)$$

$*$: if one of the following applies:
i) $a\%n < d_b$ and $a + iy \geq d_b$,
ii) $a\%n > d_b$ and $a + iy \geq d_b + n$
$**$: otherwise

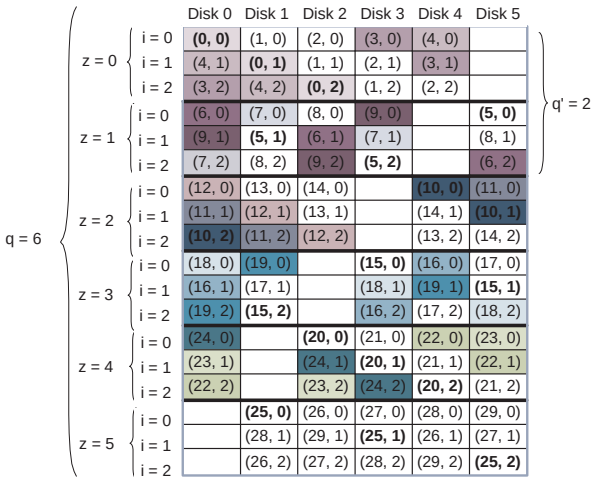

Figure 4: Example layout for $k = 3$, $n = 6$

Equation 6 defines the largest number of disks less than n that basic shifted declustering is applicable. Equation 7 is the same as equation 1 but replaces n with n'.Equation 8 defines the iteration number of a redundancy group a. Equation 9 defines the position of the bubble in the z-th iteration. Equation 10 defines the distance to shift between two units in a redundancy group. Equations 11 to 13 define the position of a unit (a, i). The number of iterations of a complete round $q = lcm(q', n)$, which is the least common multiple of the iteration number in the (k, n') configuration and the number of possible positions of the bubble. Notice that q is not necessary in computing $disk(a, i)$ and $offset(a, i)$ in this case.

Figure 4 gives an example for $k = 3$, $n = 6$, where all the desired properties are satisfied. The proof of placement-idealness of this extended layout over an even number of disks is provided in Appendix B.2.

3.3 General solution for $k > 3$

Although more than three-way replication is expensive and only applied in some specific scenarios(e.g. saving critical data) of large-scale storage systems, we still give shifted

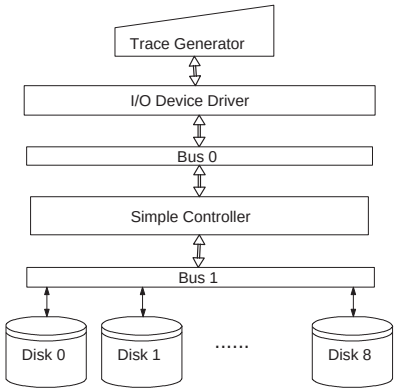

Figure 5: DiskSim simulation architecture

declustering solutions for the general cases. We first represent a basic solution over a prime number of disks, then extended it by adding imaginary bubbles over the non-prime number of disks.

1. *n is prime.*
   In this case, the distribution can be obtained from Equations 2 to 5, with the definition of q replaced by $q = n - 1$, and $i = 0 \dots k - 1$. According to this procedure, a complete round of layout has n iterations.

2. *n is non-prime.*
   In this case, the distribution can be obtained from Equations 8 to 13, with the definition of n' replaced by the largest prime number less than n, q' replaced by $q' = n' - 1$, and $i = 0 \dots k - 1$.

4. EXPERIMENTAL RESULTS

Simulations on DiskSim [1] are executed to demonstrate the performance of shifted declustering data layout for multiway replication architecture. We implement the address mapping algorithms for shifted declustering, chained declustering, group rotational declustering and RELPR[1]. After that, we compare their performance through trace driven simulations.

In DiskSim, the required architecture is demonstrated in Figure 5. A trace generator or an input trace file is at the top layer. The storage systems must be at the bottom layer. Since there must be exactly one or two controllers between the device driver and each disk with a bus connecting each pair of components [1], we incorporate a bus (Bus 0) and a simple controller between the device driver and disks, and their only job is passing requests.

We take IBM Ultrastar 36Z15 as the disk model, and the main parameters are shown in Table 3. We pick **three-way replication** since it is the most common case in multiway replication based architectures. The replica unit size is 64 KB.

In the main comparison experiments, we use 9 disks, because standard mirroring and group rotational declustering requires the number of disks to be a multiple of the number of copies. We generate synthetic traces to test the performance of the different configurations. The request size changes from 128 KB to 320 KB, and the average inter-arrival time varies from 40 ms to 500 ms. In each experiment, the request size is fixed, the inter-arrival time follows an exponential distribution of a given average value, and the requested addresses are random. In the scalability study, we scale the number of disks from 6 to 15. We use average response time and workload standard deviation to evaluate the service performance and load balancing, respectively. In Sections 4.1,4.2 and 4.4, the requests are read only. In Section 4.3, the requests are write only.

[1]RELPR is chosen instead of PRIME because PRIME requires a prime number of disks, but standard mirroring and group-rotational declustering needs the number of disks to be a multiple of the number of replications.

Table 3: Simulation Parameters

System Parameters	
Scheduling Policy: C-LOOK	Bus Bandwidth: 100 MB/s
Replica Unit: 64 KB	Number of Disks: 6, 9, 12, 15
Number of Copies: 3	
Disk Parameters:	
Capacity: 18.4 GB	Average Seek Time: 3.4 ms
Rotation Speed: 15000 RPM	Average Rotation Time: 2 ms
Disk Controller Cache: 4 MB	Transfer Rate: 55 MB/s
Workload Parameters	
Request Size : 128 KB, 256 KB, 320 KB	
Alignment :64 KB (stripe unit boundary)	
Inter Arrival Time:40 to 500 ms, exponential distribution	

4.1 Failure-free Performance

The response time statistics under a variety of workloads are summarized in Figure 6. The left diagrams demonstrate the overall average disk response time of all requests. Due to the load balancing capabilities of different data layouts, the average response time differs from disk to disk. We define the disk with the longest average response time as the *bottleneck disk*, and its average response time as the *bottleneck response time*. The bottleneck response time is illustrated in the right diagrams.

Shifted, chained and RELPR declustering obtain comparable performance in terms of both average and bottleneck response time under failure-free mode. In all diagrams of Figure 6, their curves almost overlap, and are lower than the curves of standdard mirroring and group-rotational declustering. These results agree with our theoretical analysis in Section 2.3. Under the failure-free mode, both shifted and chained declustering meet the property of *maximal parallelism*. RELPR has a near optimal parallelism, but is better than standard mirroring and group-rotational declustering.

4.2 Degraded Mode Performance

4.2.1 One-disk Failure

With one failed disk, the overall performances of all configurations are similar to the scenarios without disk failures. However, since shifted declustering layout satisfies the ideal reconstruction property, it redirects the workload of the failed disk evenly to other disks, it lowers the bottleneck response time (the longest average response time among disks) dramatically comparing to other layout schemes. The average and bottleneck response times are demonstrated in the left and right diagrams of Figure 7, respectively. The curve representing the response time of shifted declustering is always at the bottom in the diagrams, so shifted declustering provides the shortest response time in degraded mode with one failed disk.

To demonstrate the load balancing capability of shifted declustering, we assume disk 0 fails and implement load

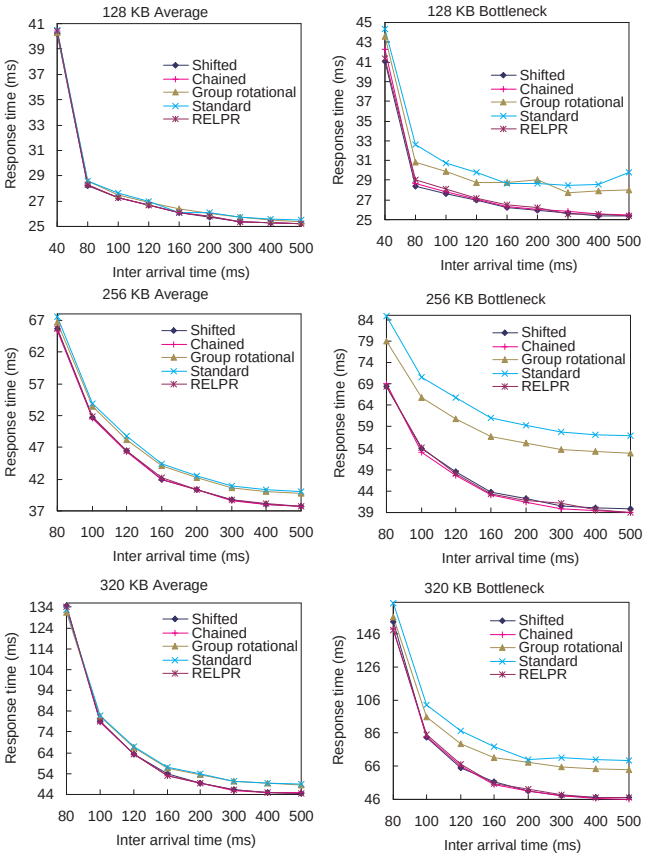

Figure 6: Overall and bottleneck average response time under failure-free mode

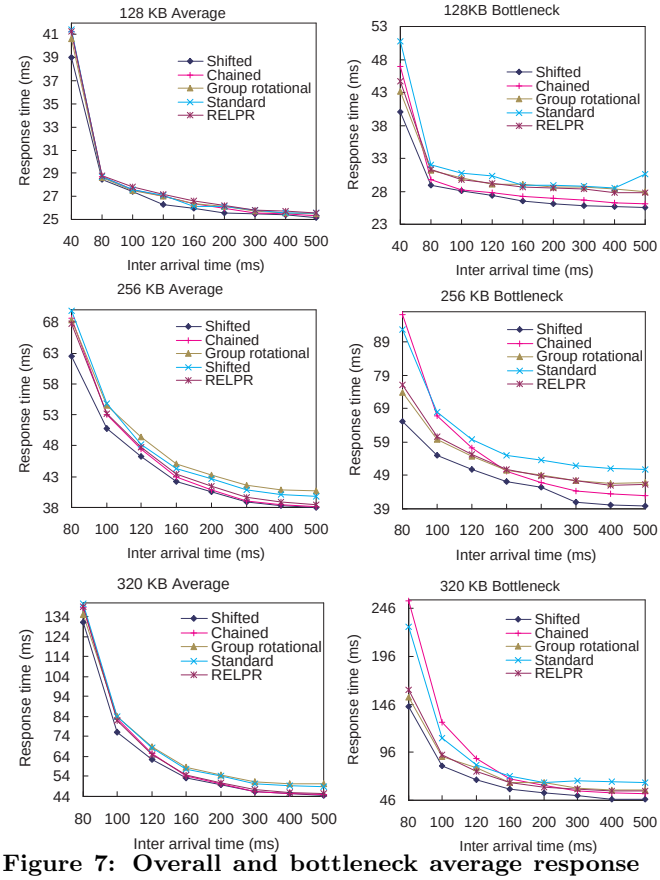

Figure 7: Overall and bottleneck average response time with one failed disk

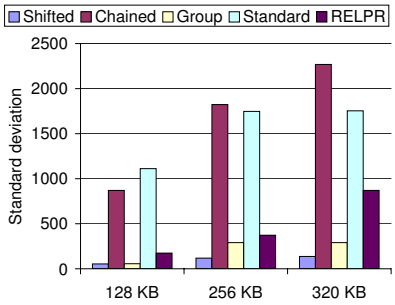

Figure 8: The standard deviation of workload with one failed disk

Table 4: Workload distribution with one failed disk

Layout	D1	D2	D3	D4	D5	D6	D7	D8
Shifted	13%	12%	12%	13%	12%	13%	12%	12%
Chained	22%	8%	14%	10%	11%	11%	15%	8%
Group	11%	15%	12%	13%	12%	12%	12%	13%
Standard	11%	15%	22%	4%	11%	11%	18%	8%
RELPR	13%	13%	11%	13%	13%	11%	13%	13%

redirecting codes in Disksim. We compare the workloads among all surviving disks. Figure 8 shows the standard deviation of number of requests which are dispatched to disks. The smaller the standard deviation is, the better load balancing capacity the layout scheme has. It is shown that shifted declustering always has the smallest standard deviation, which implies that it has the best load balancing ability under degraded mode with one failed disk.

Table 4 lists the percentage of requests on each surviving disk when the average request size is 256 KB. Since disk 0 is assumed failed, only disk 1 to 8 are counted (represented by D1 to D8 in the table). If the workload is perfectly balanced, each disk is distributed 12.5% (1/8) requests. The closer the percentage of workload is to 12.5%, the better load balancing ability the layout scheme provides. As shown in Table 4, the percentage of requests of each disk in shifted declustering is between 12% and 13%, and is the closest to 12.5% among all layouts in comparison.

4.2.2 *Two-disk Failure*

With two failed disks, shifted declustering still improves the bottleneck performance significantly, compared to the other schemes. The average and bottleneck response times are demonstrated in the left and right diagrams of Figure 9, respectively.

Similarly, the standard deviation of workload under two disk failure is shown in Figure 10. It is demonstrated that shifted declustering still has the least standard deviation.

The workload percentage of each surviving disk when the average request size is 256 KB is listed in Table 5. In this case, we assign disk 0 and disk 1 as failed, and there are seven surviving disks, so the ideal workload is 14.3%. For different request sizes, shifted declustering remains the load of each disk between 14% and 15%, and is the closest to 14.3% among all layouts in comparison.

4.3 Write Performance

In replication-based architecture, a writing request requires much more resources than a reading request does. We fix the average inter-arrival time to 300 ms, and compare the response time under different layout schemes. The results are shown in Figure 11.

Table 5: Workload distribution with two failed disks

Layout	D2	D3	D4	D5	D6	D7	D8
Shifted	14%	14%	15%	15%	14%	14%	14%
Chained	23%	8%	14%	10%	11%	15%	19%
Group	16%	5%	5%	5%	23%	23%	23%
Standard	16%	27%	13%	2%	7%	20%	16%
RELPR	16%	14%	13%	14%	14%	14%	16%

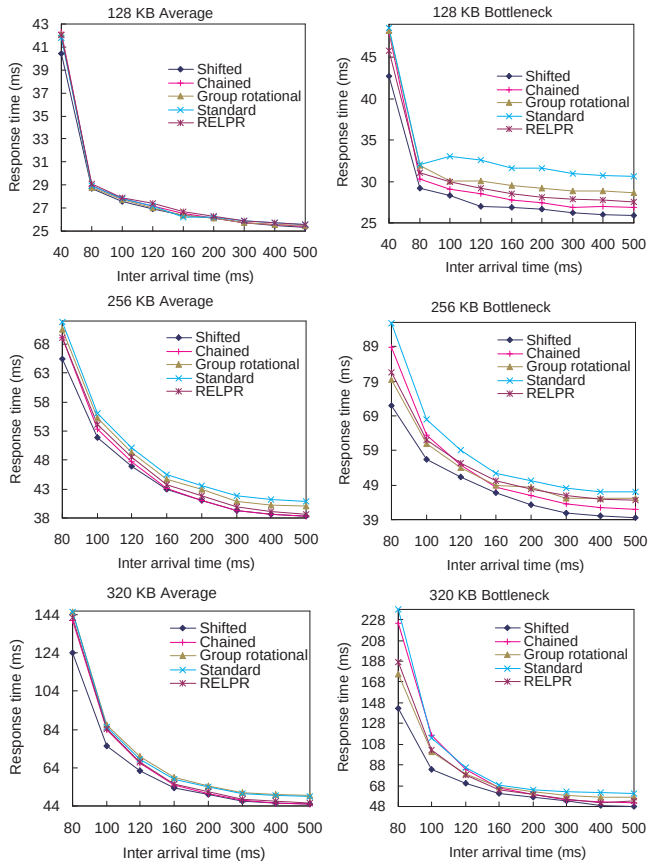

Figure 9: Comparison of average and bottleneck response time for degraded mode with two failed disks

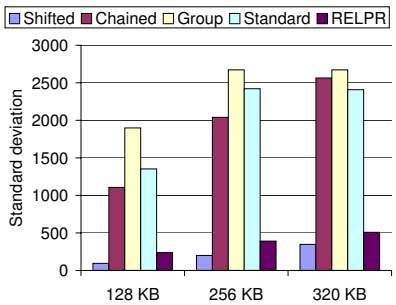

Figure 10: The standard deviation of workload with two failed disks

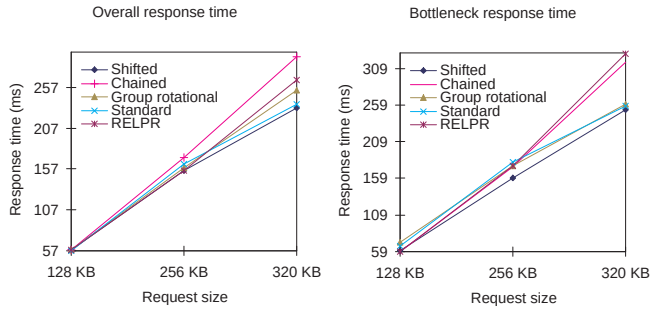

Figure 11: Writing performance

Table 6: Scalability study

System size	Mode	Workload per disk		
		Min	Max	Standard deviation
6 disks	Failure free	16%	17%	0.0052
	1-disk failure	19%	21%	0.0071
	2-disk failure	24%	26%	0.0082
9 disks	Failure free	11%	12%	0.0033
	1-disk failure	12%	14%	0.0076
	2-disk failure	13%	16%	0.010
12 disks	Failure free	8%	10%	0.0065
	1-disk failure	9%	10%	0.0030
	2-disk failure	10%	10%	0
15 disks	Failure free	6%	7%	0.0049
	1-disk failure	6%	8%	0.0047
	2-disk failure	6%	8%	0.0066

The overall average response times are close for smaller request sizes (128 KB). However, given an increasing request size, shifted declustering obtains a large performance gain over most of other schemes, due to its optimal parallelism. In addition, shifted declustering always provides the least bottleneck response time, which implies that shifted declustering also has the best load balancing capability for write.

4.4 Scalability Study

As mentioned before, shifted declustering is the placement-ideal layout for multi-way replication over any number of disks. We scale down the number of disks to 6, and scale it up to 12 and 15 in our simulation. We fix the request size to 320 KB, and fix the average inter-arrival time to 200 ms.

The load balancing statistics are shown in Table 6. Under any circumstances, the workload difference between disks is less than 2%. This result implies that the load balancing capability of shifted declustering is not sensitive to the number of disks.

5. CONCLUSIONS

In this paper, we have exploited the placement-ideal data layout in multi-way replication based storage architecture. We have summarized the differences in respect of data layout between replication and parity-based architectures. These differences were overlooked in previous researches. We also retrofitted the desirable properties defined for the declustering parity architecture, and proposed a novel placement-ideal data layout — shifted declustering. Shifted declustering layout achieves ideal parallelism in a wide variety of configurations, and thus obtains optimal service performance and load balancing in both fault-free and degraded modes, as proved by mathematical theories. Compared with traditional layout schemes such as standard mirroring, chained declustering, group rotational declustering, and RELPR declustering, comprehensive simulation results show that shifted declustering obtains much better service performance and load balancing.

6. ACKNOWLEDGMENTS

The authors would like to express their sincere thanks to the reviewers for their insightful comments, and to Mr. Christopher Mitchell, Mr. Pengju Shang and Mr. Ping Ge for proofreading. This work is supported in part by the US National Science Foundation under grants CNS-0646910, CNS-0646911, and the US Department of Energy Early Career Principal Investigator Award DE-FG02-07ER25747.

7. REFERENCES

[1] The disksim simulation environment (v3.0). http://www.pdl.cmu.edu/DiskSim/, 2007.

[2] Powervault md 1000 disk expansion enclosure details. http://www.dell.com/content/products/productdetails.aspx/pvaul_md1000, 2007.

[3] Seagate, samsung begin to ship 1tb desktop hard drives. http://www.dailytech.com/Article.aspx?newsid=7740, 2007.

[4] G. A. Alvarez, W. A. Burkhard, L. J. Stockmeyer, and F. Cristian. Declustered disk array architectures with optimal and near-optimal parallelism. In *ISCA*, pages 109–120, 1998.

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

[6] 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.

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

[8] 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.

[9] 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.

[10] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *SOSP*, pages 29–43, 2003.

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

[12] 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.

[13] J. L. Griffin, S. W. Schlosser, G. R. Ganger, and D. F. Nagle. Modeling and scheduling of mems-based storage devices. Technical Report CMU-CS-00-100, Carnegie Mellon University, 1999.

[14] F. J. MacWilliams and J. J. A. Sloane. *The Theory of Error-Correcting Codes*. Amsterdam, North-Holland, 1977.

[15] J. Marshall Hall. *Combinatorial theory (2nd ed.)*. John Wiley & Sons, Inc., New York, NY, USA, 1998.

[16] D. A. Patterson, G. Gibson, and R. H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *Proceedings of the 1988 ACM Conference on Management of Data (SIGMOD)*, pages 109–116, Chicago, IL, June 1988.

[17] W. W. Peterson and E. Weldon. *Error-Correcting Codes*. MIT Press, 1972.

[18] Y. Saito, S. Frolund, A. Veitch, A. Merchant, and S. Spence. Fab: building distributed enterprise disk arrays from commodity components. *SIGPLAN Not.*, 39(11):48–58, 2004.

[19] W. W. Wilcke, R. R. Garner, C. Fleiner, R. F. Freitas, and R. A. Golding. IBM intelligent bricks project: petabytes and beyond. IBM journal of research and development, 2006.

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

APPENDIX

A. SOLUTION FOR $K = 2$

For $k = 2$, [4] gives a layout scheme over any number of disks and proves its idealness. The layout is summarized as equations 14 to 18, where $i = 0, 1$:

$$q = \begin{cases} (n-1)/2, & \text{if } n \text{ is odd} \\ n-1, & \text{if } n \text{ is even} \end{cases} \quad (14)$$

$$z = \left\lfloor \frac{a}{n} \right\rfloor \quad (15)$$

$$y = ((z \bmod q) + 1) \bmod n \quad (16)$$

$$disk(a,i) = (a + iy) \bmod n \quad (17)$$

$$offset(a,i) = kz + i \quad (18)$$

B. PROOF OF PLACEMENT-IDEALNESS OF SHIFTED DECLUSTERING FOR $K = 3$

B.1 $n = 4$ or n is odd

1. *Failure Correcting.*
   A k-way replication redundancy configuration should be able to provide $k-1$ failure correction. This requires that no two units within a redundancy group are mapped to the same disk. In other words, for a redundancy group a, $disk(a,i) \neq disk(a,j)$ for $0 \leq i \neq j \leq k-1$. The shifted declustering scheme satisfies this property.

   PROOF. Without the loss of generality, in the following part of the proof, we assume that $i > j$. According to equation 4, $disk(a,i) = disk(a,j)$ if and only if $(a+iy) \bmod n = (a+jy) \bmod n$. This happens only if $(i-j)y$ is a multiple of n. Since $0 \leq j < i \leq k-1$, $1 \leq i-j \leq k-1$. In addition, according to the definition of q and y in equations 1 and 3, $1 \leq y \leq q$, so $1 \leq (i-j)y \leq (k-1)q$. Now we will prove that under the two conditions: *1)* $n = k+1$, *2)* n is odd, $(i-j)y$ is never a multiple of n.

   Case 1: $n = k+1$

   In this case, $q = 1$, thus $y = 1$ and $1 \leq (i-j)y \leq k-1$. Since $n = k+1$, we can conclude that $(i-j)y < n$, so $(i-j)y$ cannot be a multiple of n.

   Case 2: n is odd

   In this case, $q = (n-1)/2$, thus $1 \leq y \leq (n-1)/2$ and $1 \leq (i-j)y \leq n-1$. Similar to case 1, $(i-j)y$ cannot be a multiple of n. □

2. *Distributed Redundancy Information*[2].
   Although the replicas are equivalent of each other,

[2]The concept of redundancy information is adapted from checksum information of data-checksum redundancy configurations. This property is adapted from the property of "distributed checksum information" of [4].

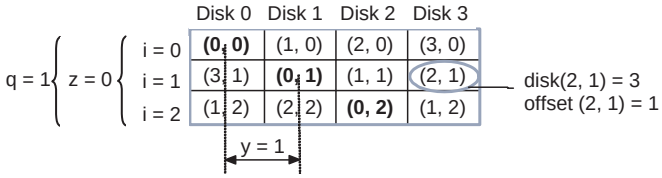

Figure 12: Example layout for $k = 3$,$n = 4$

we can view the 0-th unit in a redundancy group as the data unit, and the remaining $k-1$ replicas as redundancy information. Since the i-th replica is always placed in the i-th row in an iteration, the redundancy overhead (the fraction of the locations on a disk that contain redundancy units) of each disk is $(k-1)/k$, so the checksum units are evenly distributed.

Actually, without consideration of data layout, in k-way replication system, we can always view $(k-1)/k$ of data on a disk as redundancy, so this property holds as long as the system is configured for multi-way replication redundancy.

3. *Distributed Reconstruction.*

This property requires that there is a constant λ_{opt}, that any two disks share λ_{opt} redundancy groups. Here disks d and d' sharing a redundancy group x means that there are two units of x, say units (x, i) and (x, i'), distributed to d and d', respectively.

In our placement, within a complete round of layout, any two disks share 2 redundancy groups if $n = 4$, and share 3 redundancy groups if n is odd and greater than 4, so it satisfies distributed reconstruction.

PROOF. According to the definition of q, there are two cases:

Case 1: $n = 4$, $q = 1$.

In this case, $y = 1$, and n redundancy groups are needed to complete a round of layout. For any redundancy group $0 \le x \le n-1$, its units are distributed to disks $(x \bmod n), ((x+1) \bmod n)$, and $((x+2) \bmod n)$. Since $n = 4$, there is exactly one disk d that does not contain any unit of x, namely, $d = (x+3) \bmod n$, say that disk d is the "hole" of redundancy group x.

On the other hand, if disk d is the hole of x, since a disk has 3 units and $n = 4$, d has units from all redundancy groups other than x. Hence, if d is the hole of x, it is not the hole of other redundancy groups. In other words, x has only one hole disk which is d, and d is only the hole of x.

Similarly, for another disk d', we can find it is the hole of another redundancy group x'. Therefore, for any pair of disks d and d', they share all redundancy groups other than x and x'.

As a result, the number of shared redundancy groups is $n - 2 = 2$.

Figure 12 is an example layout for $k = 3$ and $n = 4$, and it is easily enumerable that any two disks shares 2 redundancy groups.

Case 2: n is odd.

Given any two disks d and d', without the loss of generality, $d > d'$. Based on Equations 1 to 4, they share a redundancy group x, if and only if there are i and i' with $0 \le i, i' \le 2$ and $i \ne i'$ such that

$$d = (x + iy) \bmod n \quad (19)$$

$$d' = (x + i'y) \bmod n \quad (20)$$

In equations 19 and 20, y is defined in equation 3. Subtracting the two equations of d and d' gives $d - d' = (i - i')y \bmod n$. Defining $\Delta = d - d'$, and $\delta = i - i'$, we have

$$\Delta = \delta y \bmod n \quad (21)$$

Since $d > d'$, $1 \le \Delta \le n-1$. Since $i - i' \ne 0$, there are 6 combinations of i and i', which are $(0,1), (0,2), (1,0), (1,2), (2,0)$ and $(2,1)$. In addition, the possible values for δ are -2, -1, 1 and 2. Notice that based on the definition of y, $1 \le y \le (n-1)/2$. Therefore, $-(n-1) \le \delta y \le n-1$. Hence equation 21 can be written as:

$$\Delta = \begin{cases} \delta y, & \delta > 0 \\ n + \delta y, & \delta < 0 \end{cases} \quad (22)$$

We will prove that for given d and d', only 3 combinations of i and i' yield valid $y (1 \le y \le (n-1)/2)$ that is the solution of equation 22. As a result, d and d' share exactly 3 common redundancy groups.

Given d, d', i and i', the values of Δ and δ are fixed. Solve y from equation 22 with given Δ and δ as follows:

If $1 \le \Delta \le (n-1)/2$, then

$\delta = 1 \Rightarrow y = \Delta$;
$\delta = 2 \Rightarrow y = \Delta/2$ if Δ is even;
$\delta = -2 \Rightarrow y = (n - \Delta)/2$ if Δ is odd;
$\delta = -1 \Rightarrow$ no solution for y.

Through these solutions, we can conclude that if $1 \le \Delta \le (n-1)/2$, $\delta = 1$ always yields a valid y, either $\delta = 2$ or $\delta = -2$ yields another valid y. Notice that $\delta = i - i'$, $\delta = 1$ is corresponding to two combinations of i and i': $(1,0)$ and $(2,1)$, $delta = 2$ is corresponding to the combination $(2,0)$, and $delta = -2$ is corresponding to the combination $(0,2)$. As a result, only three combinations of i and i' yield valid y for a given Δ with $1 \le \Delta \le (n-1)/2$.

Similarly, if $(n-1)/2 + 1 \le \Delta \le n-1$, then

$\delta = 1 \Rightarrow$ no solution for y;
$\delta = 2 \Rightarrow y = \Delta/2$ if Δ is even;
$\delta = -2 \Rightarrow y = (n - \Delta)/2$ if Δ is odd;
$\delta = -1 \Rightarrow y = n - \Delta$.

Still, only three combinations of i and i' yield valid y for a given Δ with $(n-1)/2 + 1 \le \Delta \le n-1$.

Moreover, fixing i, y and d fixes x, namely $x = (d - iy) \bmod n$, so for a disk pair d and d', if a pair of i and i' pair yields one common redundancy group, it yields only one common redundancy group. According to above cases, any two disks share 2 redundancy groups if $n = 4$, and share 3 redundancy groups if n is odd. □

4. *Large Write Optimization.*

Large write optimization requires that each stripe interval containing $m = k - f$ consecutive addresses where f is the maximum tolerable simultaneous failures, such that each stripe interval is mapped to the same stripe [4]. A k-way replication redundancy can tolerate $k-1$ failures, so that $f = k - 1$, and the stripe interval is

$m = k - f = 1$, it is mapped to the same stripe. Therefore, large write optimization is satisfied by replication based redundancy.

5. *Maximal Parallelism*
A layout has *maximal parallelism* if the addresses of data in any interval $\{t, t+1, \ldots, t+n-1\}$ of n consecutive addresses are mapped to n different disks. That is, for every starting point t with $0 \leq t \leq bm-n$ and every i and i' with $0 \leq i < i' \leq n-1$, $disk(t+i) \neq disk(t+i')$ [4]. In shifted declustering layout, $disk(t+i, 0)$ is defined as $(t+i)$ mod n, so $disk(t+i) \neq disk(t+i')$ always holds. Defining a metric *parallel read count* $\tau_r(c)$ to represent the number of units each disk is accessed if c continuous data units are requested, then if maximal parallelism is satisfied, $\tau_r(c) = \lceil c/n \rceil$ [4].

6. *Efficient Mapping*
Since the computation of the layout is given through equations 1 to 5, the complexity of obtaining the proper position of a given unit is constant.

B.2 $n > 4$, and n is even

1. *Failure Correcting.*
Similar as in Section B.1, this property requires that for a redundancy group a, $disk(a,i) \neq disk(a,j)$ for $0 \leq i \neq j \leq k-1$. Extended shifted declustering scheme over even number of disks satisfies this property.

PROOF. Without the loss of generality, in the following part of the proof, we assume that $i > j$. According to equations 11 and 12, $disk(a,i) = disk(a,j)$ is equivalent to $((a+\alpha) \bmod n) = ((a+\beta) \bmod n)$, where $\alpha = iy$ or $\alpha = iy+1$, and $\beta = jy$ or $\beta = jy+1$. This equation holds if and only if $\alpha - \beta$ is a multiple of n.
The following steps show that it is impossible for $\alpha - \beta$ to be a multiple of n:
Since $i > j$, $1 \leq i - j \leq 2$. According to the definition of y in equation 10, $1 \leq y \leq (n-2)/2$. The value of $\alpha - \beta$ can be one of the following cases:

(a) If $\alpha = iy$, $\beta = jy$, or $\alpha = iy+1$, $\beta = jy+1$, then $\alpha - \beta = (i-j)y$. $(i-j)y \leq (n-2)$, so $\alpha - \beta$ is not a multiple of n.

(b) If $\alpha = iy$, $\beta = jy+1$, then $\alpha - \beta = (i-j)y - 1$.$(i-j)y - 1 \leq (n-2)-1$, so $\alpha - \beta$ is not a multiple of n.

(c) If $\alpha = iy+1$, $\beta = jy$, then $\alpha - \beta = (i-j)y + 1$.$(i-j)y + 1 \leq (n-1)$, so $\alpha - \beta$ is not a multiple of n.

As a result, $disk(a,i) = disk(a,j)$ does not hold if $i > j$. □

2. *Distributed Redundancy Information*
As proved in Section B.1, this property holds as long as the system is configured as multi-way replication redundancy.

3. *Distributed Reconstruction.*
This property requires that there is a constant λ_{opt}, that any two disks share λ_{opt} redundancy groups. Here disks d and d' sharing a redundancy group x means that there are two units of x, say units (x,i) and (x,i'), distributed to d and d', respectively.
In extended shifted declustering layout over even number of disks, any two disks share $qk(k-1)/n$ redundancy groups within a complete layout, so it satisfies distributed reconstruction.

4. *Large Write Optimization.*
As proved in Section 3.2.1, large write optimization is satisfied by replication based redundancy.

5. *Maximal Parallelism*
A layout has *maximal parallelism* if the addresses of data in any interval $\{t, t+1, \ldots, t+n-1\}$ of n consecutive addresses are mapped to n different disks. That is, for every starting point t with $0 \leq t \leq bm-n$ and every i and i' with $0 \leq i < i' \leq n-1$, $disk(t+i) \neq disk(t+i')$ [4]. In exteded shifted declustering layout, $disk(t+i, 0)$ is defined as $(t+i)$ mod n, so $disk(t+i) \neq disk(t+i')$ always holds.

6. *Efficient Mapping*
Since the computation of the layout is given through equations 6 to 13, the complexity of obtaining the proper position of a given unit is constant.

C. PROOF OF PLACEMENT-IDEALNESS OF SHIFTED DECLUSTERING FOR GENERAL K AND N

In Section B.1, proofs for properties 1, 2, 4, 5 and 6 are not specific to $k = 3$. In addition, prime numbers (except two) are a subset of odd numbers, so the proofs are valid for general values of k if n is prime. Hence, we only need to prove property 3 for the general k and prime n.

Referring to Equations 1 to 5, and replacing the definition of q as $q = n-1$, we get $1 \leq y \leq n-1$. If disks d and d' shares a redundancy group x, there are i and i' with $0 \leq i, i' \leq k$ and $i \neq i'$ such that

$$y = ((z \bmod q) + 1) \bmod n \quad (23)$$

$$d = (x + iy) \bmod n \quad (24)$$

$$d' = (x + i'y) \bmod n \quad (25)$$

Subtracting the two equations of d and d' gives $d - d' = (i-i')y \bmod n$. Defining $\Delta = d - d'$, and $\delta = i - i'$, we have

$$\Delta = \delta y \bmod n \quad (26)$$

Given Δ and δ, there is exactly one y between 1 and $n-1$ satisfies equation 26. This is to say, for any two disks d and d', each pair of i and i' yields one common redundancy group. In addition, there are $k(k-1)$ combinations of i and i', so the total number of common redundancy groups yielded is $k(k-1)$, which is also $qk(k-1)/(n-1)$ because $q = n-1$.

Moreover, fixing i, y and d fixes x, namely $x = (d - iy) \bmod n$, so for a disk pair d and d', if a pair of i and i' pair yields one common redundancy group, it yields only one common redundancy group.

For n is non-prime, we can apply the similar process as in Section B.2 to get the proof. For property 6, the complexity is not constant anymore, because the complexity for finding out the largest prime which is less than n is $O(\log n)$. However, it is better than linear complexity and is affordable.