DACO: A High-Performance Disk Architecture Designed Specially for Large-Scale Erasure-Coded Storage Systems

Mingqiang Li, *Student Member*, *IEEE*, and Jiwu Shu, *Member*, *IEEE*

Abstract—Large-scale erasure-coded storage systems have a serious performance problem due to I/O congestion and disk media access congestion caused by read-modify-write operations involved in small-write operations. All the existing technologies based on the conventional disk can provide very limited performance improvement. This paper presents a new Disk Architecture with Composite Operation (DACO), whose disk media access interface consists of three kinds of operations: READ, WRITE, and Composite Operation (CO). The CO adopts a sector-based pipeline technology to implement block-level data modify operations, and thus, can replace the read-modify-write operations involved in small-write operations. When the DACO is adopted in a large-scale erasure-coded storage system with t fault tolerance, t I/Os and t disk media access operations can be reduced in each small-write operation, respectively. This alleviates both I/O congestion and disk media access congestion in nature, and thus, can remarkably improve the performance of large-scale erasure-coded storage systems. A simulation study shows that the DACO can provide significant performance improvement: reducing the average I/O response time by up to 31.16 percent even in the worst case where $t = 1$. This paper also discusses the important implementation issues of the DACO and investigates the additional cost involved in the DACO.

Index Terms—Disk architecture, erasure code, small-write problem, storage system.

1 Introduction

As we enter the data-intensive computing era, large-scale storage systems that employ a large number of disks in a clustered or distributed manner have become very important and ubiquitous. Recently, with the sharp increase of capacity, data loss has become a serious problem in such large-scale disk-based storage systems, because of concurrent disk failures [1], [2] together with multiple unrecoverable sector errors [3], [4], [5]. There have been many technologies proposed to provide sufficient reliability against data loss. They can be divided into two categories: k-way mirroring technologies and erasure-coding technologies [6]. k-way mirroring technologies provide $k-1$ fault tolerance[1] by storing k duplicates of user data. They thus have very low storage efficiency (i.e., the ratio of user data to the total of user data plus redundancy data) and can consume a large amount of additional power. In contrast, erasure-coding technologies need only a small amount of additional capacity to store the parity that is used for fault tolerance and can provide optimal or approximately optimal storage efficiency. Correspondingly, they consume only a small amount of additional power. Because of their high storage efficiency and low power consumption, erasure-coding technologies have become an attractive trend for fault tolerance in large-scale disk-based storage systems and have recently been widely adopted in RAID subsystems [7], [8] and various fault-tolerant storage systems, such as OceanStore [9], Glacier [10], FAB [11], PASIS [12], [13], RobuSTore [14], Pergamum [15], Cleversafe [16], Allmydata [17], and Permabit [18]. In this paper, we call such storage systems *erasure-coded storage systems*.

However, erasure-coded storage systems have a serious small-write problem, which confuses system designers all along. In an erasure-coded storage system with t fault tolerance,[2] when a small-write operation updates a data strip, it should update all the t parity strips in the same stripe using read-modify-write operations. This process can involve $2(t+1)$ I/Os and $2(t+1)$ disk media access operations, respectively. It thus increases both I/O congestion and disk media access congestion by approximately a factor of $2(t+1)$. This can badly hurt storage performance, especially in large-scale high-performance computing systems that run data-intensive applications, such as Data-Intensive Super Computing (DISC) systems [19]. Moreover, with the growth of the capacity of erasure-coded storage systems and the increase of their fault tolerance, this small-write problem will become more serious.

In order to solve this small-write problem, several technologies [7], [20], [21], [22], [23], [24] based on the conventional disk have been proposed for small-scale erasure-coded storage systems (such as RAID 5[3] [7]). Since all these existing technologies do not change the

1. $k-1$ fault tolerance means that the maximum number of fault disks that can be reconstructed is $k-1$.

• *The authors are with the Institute of High-Performance Computing, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.*
E-mail: lmq06@mails.tsinghua.edu.cn, shujw@tsinghua.edu.cn.

Manuscript received 30 Oct. 2008; revised 3 Sept. 2009; accepted 13 Dec. 2009; published online 14 Jan. 2010.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2008-10-0534.
Digital Object Identifier no. 10.1109/TC.2010.22.

2. An erasure-coded storage system with t fault tolerance is a storage system that adopts an erasure code with t fault tolerance.

3. In RAID 5, a Single Parity Check (SPC) code, which can be regarded as an erasure code with one fault tolerance, is adopted.

read-modify-write mode in small-write operations, they can provide very limited performance improvement. Exactly speaking, they can improve some visible performance only in small-scale erasure-coded storage systems whose small-write problem is not very serious because of their low fault tolerance. However, in large-scale erasure-coded storage systems with high fault tolerance, since the small-write problem is much more serious, the performance improvement of these existing technologies will become negligible. To cope with this challenge, we will develop a new approach to solve the small-write problem in large-scale erasure-coded storage systems.

Our basic idea is to change the read-modify-write mode in small-write operations. As we know, in today's disk-based storage systems, the disk media access interface consists mainly of simple READ and WRITE operations. As mentioned in [25], more expressive interfaces, together with extended versions of today's Operating Systems (OSes) and firmware specializations, would allow the OSes and storage devices to cooperate to achieve performance and functionality that neither can be achieved alone. Thus, in this paper, we propose a new Disk Architecture with Composite Operation (DACO), whose disk media access interface is more expressive and consists of three kinds of operations: READ, WRITE, and Composite Operation (CO). The CO adopts a *sector-based pipeline technology* (whose fundamental unit of processed data is set to be a sector) to implement *block-level data modify operations*.[4] Unlike in the conventional disk, a block-level data modify operation then can be implemented by only a CO rather than a read-modify-write operation in the DACO. When we use the DACO in a large-scale erasure-coded storage system with t fault tolerance, only $t+2$ I/Os and $t+2$ disk media access operations will be involved in each small-write operation, respectively. This alleviates both I/O congestion and disk media access congestion caused by small-write operations in nature, and thus, can remarkably improve the performance of large-scale erasure-coded storage systems.

The main contributions of this paper are as follows:

- We provide an overview of the disk architecture of the DACO and also describe the implementation details of the CO. We deduce that the time required for the CO very approximates to that required for the general WRITE. We also reveal an interesting conclusion that the DACO can provide more performance improvement in the erasure-coded storage system whose I/O workload includes more random small writes.
- We demonstrate the performance potential of the DACO by carrying out a trace-driven simulation study [26]. We also investigate the effects of different factors on the performance improvement of the DACO. Numerical results show that the DACO can reduce the average I/O response time by up to 31.16 percent even in the worst case where $t=1$. We also claim that if applied to large-scale erasure-coded storage systems with higher fault tolerance, the DACO can provide more performance improvement.
- We discuss two important implementation issues of the DACO: head positioning servomechanism and failure handling during the CO. We also investigate the additional cost involved in the DACO and then claim that the DACO is the most cost-effective choice for large-scale erasure-coded storage systems.

This paper is organized as follows: In the next section, we first present the necessary background and related work. We provide an overview of the DACO in Section 3. Performance potential is then demonstrated in Section 4. In Section 5, we discuss the implementation issues. We also investigate the additional cost involved in the DACO in Section 6. Finally, we conclude this paper in Section 7.

2 Background and Related Work

2.1 Erasure-Coded Storage Systems and Small-Write Problem

There are many erasure codes [6] proposed for storage systems. According to whether they can provide optimal storage efficiency, these existing erasure codes can be divided into two categories: *Maximum Distance Separable (MDS) codes*[5] (such as Reed-Solomon codes [28], [29], [30], [31], EVENODD [32], X-code [33], STAR [34], and Liberation codes [35]) and non-MDS codes (such as WEAVER codes [36] and GRID codes [37]). In this paper, we will focus our discussion primarily on MDS codes because they are typical representatives among all the existing erasure codes. Moreover, among these MDS codes, since *horizontal codes* (such as Reed-Solomon codes, EVENODD, STAR, and Liberation codes) that store data and parity on separate strips can be implemented more flexibly than *vertical codes* (such as X-code) in which each strip within a stripe contains both data and parity, horizontal codes seem to be more popular in practice. Thus, we will focus our attention on horizontal MDS codes. Throughout this paper, when referring to erasure codes, we always mean horizontal MDS codes.

In an n-of-$(n+t)$ erasure-coded storage system with t fault tolerance, a stripe consists of $n+t$ strips, including n data strips (i.e., D_1, $D_2, \ldots,$ and D_n) and t parity strips (i.e., P_1, $P_2, \ldots,$ and P_t), which meet the following binary linear equations:

$$\begin{cases} (X_{1,1} \odot D_1) \oplus (X_{1,2} \odot D_2) \oplus \cdots \oplus (X_{1,n} \odot D_n) = P_1, \\ (X_{2,1} \odot D_1) \oplus (X_{2,2} \odot D_2) \oplus \cdots \oplus (X_{2,n} \odot D_n) = P_2, \\ \qquad\qquad \vdots \\ (X_{t,1} \odot D_1) \oplus (X_{t,2} \odot D_2) \oplus \cdots \oplus (X_{t,n} \odot D_n) = P_t, \end{cases} \tag{1}$$

where $X_{i,j}$ ($1 \le i \le t$ and $1 \le j \le n$) is a binary coefficient matrix, $\odot$ represents a binary matrix multiplication operator, and $\oplus$ represents a binary matrix addition operator.

When a small-write operation updates a data strip D_j ($1 \le j \le n$), it should update all the t parity strips (i.e., P_1, $P_2, \ldots,$ and P_t) in the same stripe using read-modify-write operations. During this process, the old data strip $\overset{old}{D}_j$ is

4. *Block-level data modify operations* are opposite to *file-level data modify operations*. Their primary difference is that block-level data modify operations are nonsemantic operations on disk data blocks, while file-level data modify operations are semantic operations on files.

5. *MDS codes* are the family of erasure codes that attain the Singleton bound [27], and thus, can provide optimal storage efficiency.

first read from the disk. After that, the new data strip $\overset{new}{D_j}$ is then written to the original location on the disk. This increases the response time of the write operation by approximately a factor of two. Meanwhile, all the t old parity strips (i.e., $\overset{old}{P_1}$, $\overset{old}{P_2}, \ldots$, and $\overset{old}{P_t}$) are read from other t disks separately. Then, t new parity strips (i.e., $\overset{new}{P_1}$, $\overset{new}{P_2}, \ldots$, and $\overset{new}{P_t}$) are calculated by the expression

$$\overset{new}{P_i} = Incr_i \oplus \overset{old}{P_i} \ (i = 1, 2, \ldots, t), \tag{2}$$

where $Incr_i$ represents the increment on P_i and is equal to $X_{i,j} \odot (\overset{old}{D_j} \oplus \overset{new}{D_j})$. Finally, t new parity strips are written to the original t locations on the t disks separately. This process involves $2(t+1)$ I/Os in total, and thus, increases I/O congestion by approximately a factor of $2(t+1)$. Correspondingly, it involves $2(t+1)$ disk media access operations in total, and thus, increases disk media access congestion by approximately a factor of $2(t+1)$. Therefore, small-write operations can badly hurt storage performance, especially in large-scale erasure-coded storage systems with high fault tolerance.

2.2 Technologies for Improving Small-Write Performance

In order to improve small-write performance, several technologies [7], [20], [21], [22], [23], [24] have been proposed for small-scale erasure-coded storage systems (such as RAID 5 [7]). They can be grouped into two categories:

1. **Group A:** This group of technologies improve the performance by optimizing the order of the I/O sequence [7], [20], [21], [22]. Among them, buffering and caching [7] are two general methods for I/O optimization. However, they are not proposed specially for small writes and can take effect only when the I/O sequence has very good temporal and spatial localities. Moreover, our DACO does not conflict with them and can be combined with them to further improve the performance. There are also some technologies [20], [21], [22] proposed specially for improving small-write performance. Some improve the performance by floating the parity to the nearest unallocated blocks on the disk [20]. However, to exploit unallocated blocks immediately following the parity being read, the stored data must be modified and remapped. This sometimes can badly destroy the logical continuity of the data. Others (like parity logging [21]) improve the performance by delaying the read of the old parity and the write of the new parity [21], [22]. Delaying these operations allows dispersive small parity updates to be grouped together into contiguous larger parity updates that can be performed more efficiently. However, they can take effect also only when the I/O sequence has very good temporal and spatial localities. Moreover, their approach is orthogonal to that of the DACO, which uses CO operations to replace read-modify-write operations involved in parity updates, and thus, they can be combined with the DACO to further improve the performance. Since this group does not change I/O interface (see Fig. 1a), and thus, cannot reduce the numbers of I/Os and disk media access operations involved in each small-write operation (see Fig. 2), their performance improvement is very limited, especially in busy I/O subsystems with a large number of random small I/O requests. This results in the emergence of the next group.
2. **Group B:** This group of technologies embed processors and a substantial amount of memory into the disk controller, but do not change the disk drive [23], [24]. In these technologies, parity modify operations are migrated from the disk driver into the disk controller (see Fig. 1b). These technologies introduce XOR commands [23], [24] into the I/O interface and then can reduce the number of I/Os involved in each small-write operation by t in an erasure-coded storage system with t fault tolerance (see Fig. 2a). However, since they do not change disk media access interface (see Fig. 1b), and thus, cannot reduce the number of disk media access operations involved in each small-write operation (see Fig. 2b), their performance improvement is also very limited, especially in busy disk media access subsystems.

All these existing technologies provide limited performance improvement. They can improve some visible performance only in small-scale erasure-coded storage systems whose small-write problem is not very serious

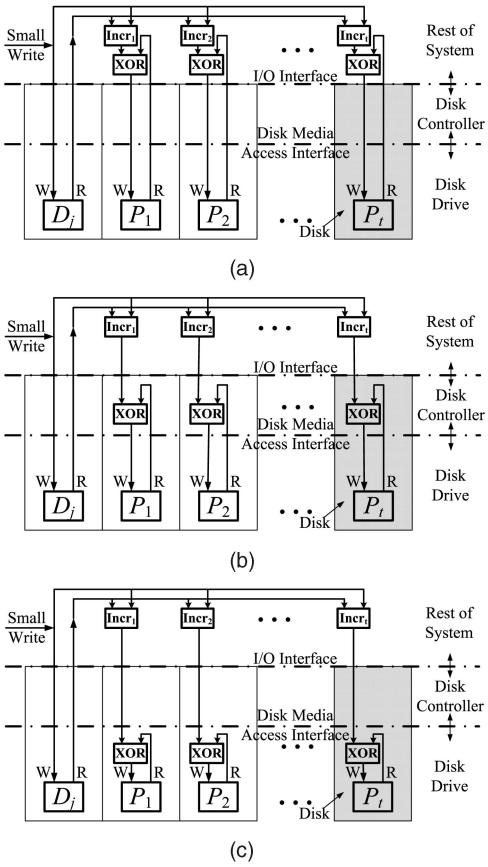

Fig. 1. Three implementation levels of a small-write operation. (a) Group A. (b) Group B. (c) DACO.

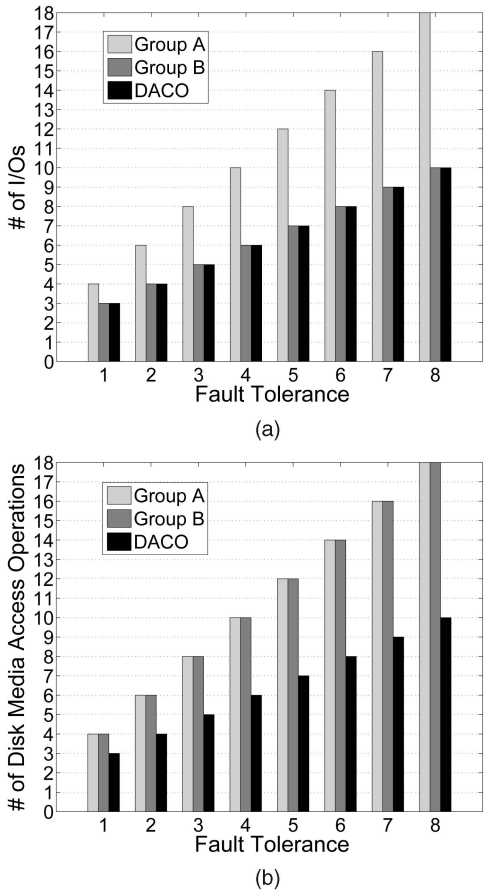

Fig. 2. Numbers of I/Os and disk media access operations involved in each small-write operation in an erasure-coded storage system with different fault tolerance when we use different categories of technologies to improve small-write performance. (a) Number of I/Os. (b) Number of disk media access operations.

because of their low fault tolerance. However, in large-scale erasure-coded storage systems with high fault tolerance, the small-write problem becomes much more serious, and the performance improvement of these existing technologies will become negligible. Moreover, with the increase of fault tolerance, the small-write problem will become more serious (see Fig. 2), and the performance improvement of these existing technologies will become more negligible.

To cope with this challenge, we propose a new disk architecture (namely DACO) designed specially for large-scale erasure-coded storage systems. In the DACO, parity modify operations are further migrated from the disk controller into the disk drive (see Fig. 1c). When the DACO is adopted in an erasure-coded storage system with t fault tolerance, t I/Os and t disk media access operations can be reduced in each small-write operation, respectively (see Fig. 2). This alleviates both I/O congestion and disk media access congestion in nature, and thus, can remarkably improve the performance of large-scale erasure-coded storage systems. Moreover, with the increase of fault tolerance, the performance benefit of the DACO will become more remarkable (see Fig. 2). This characteristic makes the DACO very attractive in large-scale erasure-coded storage systems with high fault tolerance.

2.3 A Brief Historical Retrospect of Multiple Arm/Head Disks

As will be shown in the next section, one of the interesting aspects of the DACO is the use of two heads mounted on the same arm on each platter surface. In this subsection, we first give a brief historical retrospect of multiple arm/head disks and then discuss the difference between the DACO and these multiple arm/head disks.

Multiple arm/head disks once existed in the market from early 1970s to early 1990s. The first disk architecture with two arms was the IBM 3340 disk drive [38], [39] developed in 1973, in which only one arm was capable of moving each time. Later, the possibility of multiple arms that are capable of moving independently was explored in [40]. Then, the IBM 3380 disk drive [41] with four arms that embodied this feature was released in 1981. There was also the Chinook disk drive [42] with two arms produced by Conner Peripherals, Inc., from late 1980s to early 1990s. All these multiple arm/head disks were off production later due to the cost of manufacturing and the availability of disk arrays. In recent years, with the price per megabyte in hard disk drive sharply declining [43], multiple arm/head disks have been suggested and researched again in patent literatures [44], [45] and academic research papers [46], [47], [48], [49], [50], [51] so as to meet the urgent demand of high-performance storage systems. The results in the 2008 ISCA paper by Sankar, Gurumurthi, and Stan [51] even show that an *intradisk parallelism*[6] design that uses multiple arm assemblies can facilitate breaking-even with, or even surpassing the performance of a disk array, while consuming significantly lower power than the disk array. This is a complete trend reversal from the multiple arm/head disks of decades past. This also reveals that the time of multiple arm/head disks has come again.

All these multiple arm/head disks proposed in related work are based on the fact that putting multiple heads in the disk drive would allow multiple areas of the disk to be accessed simultaneously, and thus, could greatly improve storage performance. In addition, all these disks, except the Dual-Actuator Logging Disk (DALD) proposed in [50], adopt multiple complex integrated read/write heads that are mounted on separate arms. We will see later in the next section that the DACO adopts only a simple read head and a simple write head that are mounted on the same arm. This simplifies the disk drive design, and thus, reduces the additional hardware cost, which will be discussed in detail in Section 6. Although the DALD also adopts only a simple read head and a simple write head, the two heads are mounted on separate arms. Moreover, since its objective is to minimize the access latency of synchronous writes, it is not suitable for the small-write problem. Finally, the DACO does not conflict with all these multiple arm/head disks and can be extended by adding some additional integrated read/write heads into the disk drive to further improve the performance.

3 Disk Architecture with Composite Operation

This section provides an overview of our DACO. It starts with the definition of Composite Operation (CO). The disk

6. Disk drives using *intradisk parallelism* can exploit parallelism in the I/O request stream.

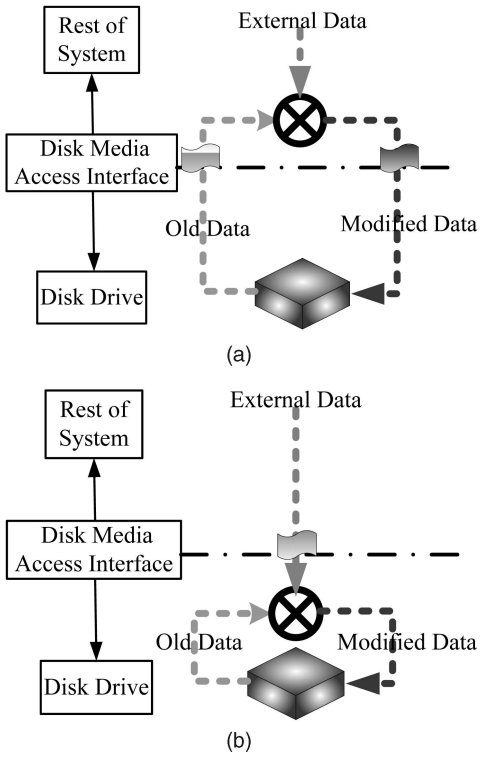

Fig. 3. Two implementation levels of a block-level data modify operation. (a) Conventional disk. (b) DACO.

architecture of the DACO is then described in the second subsection. The third subsection discusses the time required for the CO. Finally, it ends with a discussion on the time reduced by the DACO in a parity update scenario.

3.1 Composite Operation Definition

In the conventional disk, a block-level data modify operation is implemented by a read-modify-write process, as shown in Fig. 3a. From this figure, we can see that a block-level data modify operation can involve two disk media access operations, one READ and one WRITE, and will take, on average, a seek and $3/2$ rotations. Thus, in a large-scale erasure-coded storage system with t fault tolerance, a small-write operation can involve $2(t+1)$ disk media access operations, including $t+1$ READs to read old data and parity strips and $t+1$ WRITEs to write new data and parity strips.

We propose that a block-level data modify operation can be implemented at a lower level. As shown in Fig. 3b, the modify operation can be implemented in the disk drive, and then, only one disk media access operation is involved in each modify operation, only taking, on average, a seek and $1/2$ rotation. This disk media access operation can be regarded as the composite operation of READ and WRITE, and so is called *Composite Operation* (CO). With the introduction of the CO, a small-write operation will then involve only $t+2$ disk media access operations, including one READ to read old data strip, one WRITE to write new data strip, and t COs to modify t parity strips. This reduces $\frac{t}{2(t+1)} \times 100\%$ of disk media access operations in each small-write operation. When $t=1$, the reduced proportion occupies 25 percent. With the increase of the fault tolerance t, the reduced proportion will increase approximately to 50 percent.

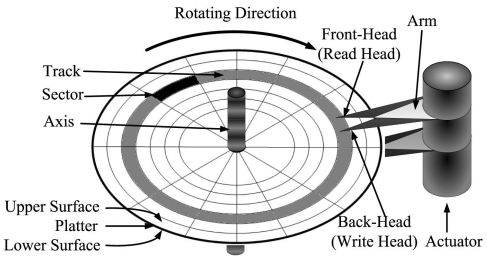

Fig. 4. Disk drive in the DACO.

3.2 Disk Architecture Description

The disk media access interface of the DACO consists of three kinds of operations: READ, WRITE, and CO. The READ and WRITE operations are the same as those in the conventional disk, while the CO operation is more complex.

In order to support the CO, each platter surface has two associated heads in the disk drive (see Fig. 4): a front-head and a back-head. The front-head is a simple read head, while the back-head is a simple write head. As mentioned in [40], the major difficulty with multiple arms is to simultaneously track multiple separate mechanical arms. This is because in multiple arm disks, track-seeking and track-following operations can occur at the same time, and the mechanical interaction caused by the reaction force of the track-seeking arm tends to degrade the positioning accuracy of the track-following arm [49]. To avoid this problem, these two heads are mounted on the same disk arm. This also reduces the additional hardware cost. Then, these two heads have the same seek time. In addition, in order to support sectors' error checking and correction (which has evolved from the early peak detection to the imminent iterative detection [52], [53]) during the CO and avoid crosstalk, these two heads should keep a several-sector distance from each other. Moreover, to achieve the accurate positioning of the two heads, a dual-stage servomechanism is introduced into the disk drive. This head positioning servomechanism will be discussed in detail in Section 5.1. In addition, an additional computing element is also employed to implement the block-level data modify operation in the disk drive. As will be shown later, since the modify operation is an XOR operation, its implementation is very simple and needs only a few simple gate circuits. Thus, the computing element can be implemented in one of the disk's built-in Digital Signal Processors (DSPs) or microcontrollers so as to minimize the additional hardware cost and maximize the flexibility.

We implement the CO by adopting a *sector-based pipeline technology*, whose fundamental unit of processed data is set to be a sector. The reason is that a sector is the fundamental unit of disk's accessed data, on which error checking and correction is performed in the disk drive's read/write channel. A CO pipeline is divided into three stages:

1. **Stage R:** Read one sector of the old data from the disk platter using the front-head (note that the sector's error checking and correction is also included in this stage);
2. **Stage M:** Modify the sector using the computing element; and

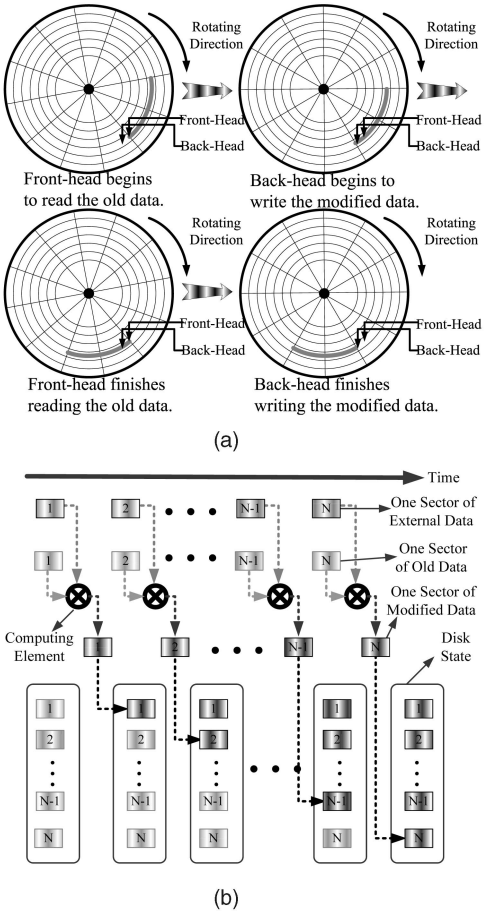

Fig. 5. Implementation details of the CO. (a) Behaviors of the two disk heads. (b) Behaviors of the computing element.

3. **Stage W:** Write the modified sector onto the disk platter using the back-head.

Fig. 5 shows the implementation details of the CO. Since the data stored on the disk can be regarded as a sector sequence, we assume that the old data consists of N sectors. The external data then should have the same length. After seeking and rotating, the front-head arrives at the start position of the old data and then begins to read them. After the first sector of the old data is read from the disk platter, the disk drive checks the sector's correctness using one of error checking and correction technologies, such as the Iterative Detection Read Channel (IDRC) technology recently developed by Hitachi Global Storage Technologies (GST) [52]. If this sector is justified to be correct, the computing element then modifies it using the first sector of the external data. The first sector of the modified data is then passed to the back-head. When the back-head arrives at the start position of the old data, it writes the first sector of the modified data onto the disk platter. Since the modify operation is an XOR operation, its implementation is very simple, and thus, the rate of data modifying is much higher than that of data reading. After the computing element modifies the first sector of the old data, it waits for the second sector of the old data to be read from the disk platter. When the second sector of the old data arrives, this process is repeated as above until the last sector of the modified data is written onto the disk platter. Then, the CO operation is completed.

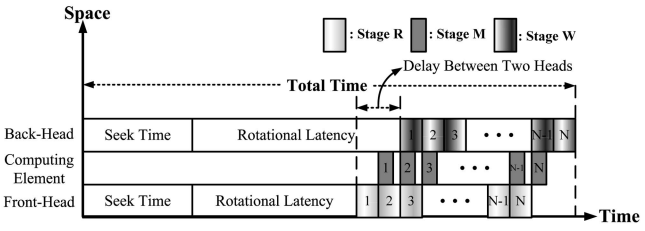

Fig. 6. Space-time diagram of the CO.

3.3 Time Required for the CO

We now discuss the time required for the CO (denoted by T_{CO}). Fig. 6 gives the space-time diagram of the CO. We use T_{WRITE} to denote the time required for the general WRITE operation. We also use T_{delay} to denote the delay between the two heads during the CO. From this figure, we can see that the time required for the CO is

$$T_{CO} = T_{WRITE} + T_{delay}. \tag{3}$$

Recently, the spindle speed of hard disks has reached 15,000 Revolutions Per Minute (RPM) in high-performance disk drives (such as the Seagate Cheetah hard drive family [54]) and will continually increase in the future. This can result in only 4 ms per revolution. In addition, as we know, today's hard disks with the use of *Zoned Bit Recording (ZBR)* [55] can have thousands of sectors in a single track. Since the front-head and the back-head in the DACO keep only a several-sector distance from each other, the delay between the two heads can then be in the time scale of 0.01s ms. However, in modern disks, the time required for the general WRITE operation is often in the time scale of 10s ms, which can be about 1,000 times as long as the delay between the two heads. We have $T_{delay} \ll T_{WRITE}$. Then, from Eq. (3), we can deduce that

$$T_{CO} \approx T_{WRITE}. \tag{4}$$

Therefore, the time required for the CO very approximates to that required for the general WRITE.

3.4 Time Reduced by the DACO in a Parity Update Scenario

Having known the time required for the CO, we then give some examples to demonstrate how much time can be reduced by the DACO in a parity update scenario. Suppose there are three parities to be updated. As shown in Fig. 7, if these three parities are distributed in the same track, one rotation can be reduced by the DACO; if these three parities are distributed in two separate tracks, two rotations can be reduced by the DACO; or if these three parities are distributed in three separate tracks, three rotations can be reduced by the DACO. From this figure, we can immediately deduce the following conclusion:

Conclusion 3.1. In a parity update scenario, if the parities to be updated are distributed in m tracks, m rotations can then be reduced by the DACO.

The above conclusion reveals an interesting phenomenon that the DACO can reduce more time in the parity update scenario where the parities to be updated are distributed more dispersedly (i.e., in more separate tracks). Thus, the

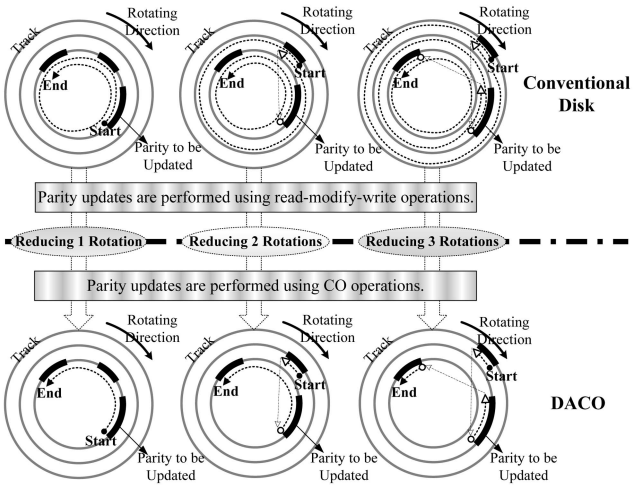

Fig. 7. Some examples to demonstrate how much time can be reduced by the DACO in a parity update scenario.

DACO not only can provide significant performance improvement in the erasure-coded storage system whose I/O workload exhibits some degrees of sequentiality and locality, but also can provide even more performance improvement in the erasure-coded storage system whose I/O workload includes more random small writes.

In this section, we have provided an overview of the DACO. Although the DACO is designed specially for reducing the performance loss caused by the introduction of erasure-coding technologies in which block-level modify operations are XOR operations, it can also be extended for reducing the performance loss caused by the introduction of other kinds of block-level technologies (such as block-level security technologies [56]) in which block-level modify operations are also logical operations. In fact, the modify operation in the DACO's computing element could be any other logical operation (such as AND, OR, NOT, NAND, NOR, or XNOR) that can be implemented in the same manner as an XOR operation. However, this topic is beyond the scope of this paper and will be studied in our future work.

4 Performance Potential

In this section, we demonstrate the performance potential of the DACO. We first introduce the evaluation methodology and then present and analyze the numerical results.

4.1 Evaluation Methodology

We carried out a trace-driven simulation study [26] to demonstrate the performance potential of the DACO. Since with the fault tolerance increasing, the performance benefit of the DACO will become more remarkable, we took the worst case where the fault tolerance is $t = 1$ as an example to demonstrate the performance potential of the DACO in our simulation study.

We built a simulator, based on the state-of-the-art disk simulator, DiskSim version 3.0 [57], to simulate erasure-coded storage systems constituted of DACOs. In the simulator, we assume that the time required for the CO is equal to that required for the general WRITE. This assumption is obviously reasonable from what has been discussed in Section 3.3.

The base disk model used in the simulation is the Quantum Atlas 10K (9.1 G, 10,025 RPM), for which publicly available configuration parameters have been calibrated against real-world drives. Note that although the Quantum Atlas 10K is somewhat old (released in 1999), it is the latest disk model available in the DiskSim version 3.0. A simulation study with the use of the Quantum Atlas 10K can still demonstrate the performance potential of the DACO.

In the simulation, we used $8 + 1$ disks to organize a striped erasure-coded disk array with $t = 1$ (note that although we used a disk array here that belongs to the category of small-scale erasure-coded storage systems, it can still help us to understand the performance potential of the DACO), in which each stripe consists of eight data strips and one parity strip, and all parity strips are rotated among all the $8 + 1$ disks. In addition, we set the size of the stripe unit to be 32 KB.

We used several real-world traces from different systems to drive our simulator. They were originally collected by the Storage Systems Department at HP Labs and are now made available as open source [58]. The TPC-C traces were collected in HP's client/server TPC-C application running at approximately 1,150 tpmC on a 100-warehouse database in 1994. The TPC-D1 and TPC-D2 traces belong to the 300 GB TPC-D Oracle traces from 1997. The Cello96 traces were taken from the Cello server over the period September 9 to November 29, 1996, when Cello was a K410 class machine (two CPUs) running HP-UX 10.10, with about 0.5 GB of main memory. The Cello99 traces were taken from the Cello server over the period January 14 to December 31, 1999, when Cello was a K570 class machine (four CPUs) running HP-UX 10.20, with about 2 GB of main memory. Note that although these traces are somewhat old, they are still widely used in today's academic research papers due to the lack of newer publicly accepted traces. Due to long simulation time and numerous combinations of simulation tests, we used only a fraction of these traces in the simulation. Specifically, we used the traces on the most active devices. Table 1 summarizes the key characteristics of the used portions of traces.

4.2 Numerical Results and Discussions

In this subsection, we present and analyze the simulation results. Our purpose in introducing the DACO is to reduce the number of disk media access operations and ultimately to reduce the average I/O response time. Thus, we use the average I/O response time as the evaluation metric.

We first made a group of simulation experiments to give an overview of the performance improvement of the DACO. In the simulation, we replayed the five traces listed in Table 1 on two erasure-coded disk arrays constituted of conventional disks and DACOs, respectively. These two erasure-coded disk arrays were both configured as in the previous subsection.

Fig. 8 gives the simulation results for both the conventional disk and the DACO. From this figure, the following two observations are in order:

- When some writes are included in the I/O workload, the DACO always has better performance than

TABLE 1
The Key Characteristics of the Used Portions of Traces (IOPS: I/Os Per Second)

Trace Name	TPC-C	TPC-D1	TPC-D2	Cello96	Cello99
Number of I/Os	225200	25112	25022	411071	577694
Percentage of Writes	98.82%	0.00%	50.02%	85.56%	79.12%
I/O Request Rate	28.66 IOPS	3.60 IOPS	5.86 IOPS	57.10 IOPS	80.24 IOPS
I/O Request Size	2KB∼128KB	16KB∼64KB	32KB∼64KB	1KB∼52KB	1KB∼64KB

the conventional disk, due to the decrease of the number of disk media access operations. Compared with the conventional disk, the DACO can reduce the average I/O response time by up to 29.23 percent (in the case of the Cello99 traces).

- In the application with longer average I/O response time, the DACO can reduce more average I/O response time. This implies that the DACO can provide more performance improvement in the application that is more bottlenecked by the storage subsystem.

The above two observations reveal that the performance improvement of the DACO varies with the following two factors: the percentage of writes and the degrees of I/O congestion and disk media access congestion. In an erasure-coded disk array, the degrees of I/O congestion and disk media access congestion concretely depend on the following two factors: the I/O request rate (of the I/O workload driving the disk array) and the number of disks (employed in the disk array).

We then made three groups of simulation experiments to investigate the effects of the above three factors (i.e., percentage of writes, I/O request rate, and number of disks) on the performance improvement of the DACO, respectively. To accurately study each factor, we varied only the parameter corresponding to the factor each time and maintained all the other parameters as in the foregoing group of simulation experiments.

4.2.1 Effects of Percentage of Writes

We first made a group of simulation experiments to investigate the effects of percentage of writes on the performance improvement of the DACO. Due to long simulation time and numerous combinations of simulation tests, we only chose the three traces with very different percentages of writes: TPC-C, TPC-D2, and Cello99. In the simulation, we changed these three traces by varying the percentage of writes. When varying the percentage of writes, we randomly changed some read operations into write operations, or randomly changed some write operations into read operations.

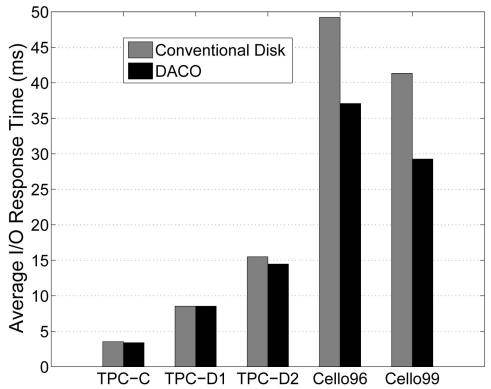

Fig. 8. Comparison between the conventional disk and the DACO.

Fig. 9a gives the simulation results for different percentages of writes. From this figure, we can see that for the I/O workload with higher percentage of writes, the DACO can reduce more average I/O response time in most cases, and this change tendency is often very remarkable. For example, for the Cello99 traces with the percentage of writes being 79.12 percent, the average I/O response time reduced by the DACO is 12.08 ms, which is about four times as long as that for the Cello99 traces with the percentage of writes being 27.68 percent. Note that for the TPC-C traces, there is a reverse change tendency on the average I/O response time. The reason is that when we varied the percentage of writes in the TPC-C traces, a large number of *logically sequential*[7] writes are separated into random small writes. This also proves the fact mentioned in Section 3.4 that the DACO can provide more performance improvement for the I/O workload with more random small writes.

4.2.2 Effects of I/O Request Rate

We then made a group of simulation experiments to investigate the effects of I/O request rate on the performance improvement of the DACO. Similarly, we also used the three traces: TPC-C, TPC-D2, and Cello99. In the simulation, we changed these three traces by varying the I/O request rate.

Fig. 9 gives the simulation results for different I/O request rates. From this figure, we can see that for the I/O workload with higher I/O request rate, the DACO can reduce more average I/O response time, and the improvement on the average I/O response time can increase up to 30.04 percent (in the case of the TPC-D2 traces with 23.42 IOPS). This change tendency is sometimes very remarkable. For example, for the TPC-D2 traces with 23.42 IOPS, the average I/O response time reduced by the DACO is 10.68 ms, which is more than 10 times as long as that for the TPC-D2 traces with 5.86 IOPS. This implies that the DACO can provide much more performance improvement in the application with busier I/O traffic.

4.2.3 Effects of Number of Disks

We finally made a group of simulation experiments to investigate the effects of number of disks on the performance improvement of the DACO. In the simulation, we changed

7. I/O requests are defined to be *logically sequential* if they are at adjacent disk addresses or disk addresses spaced by the file system interleave factor.

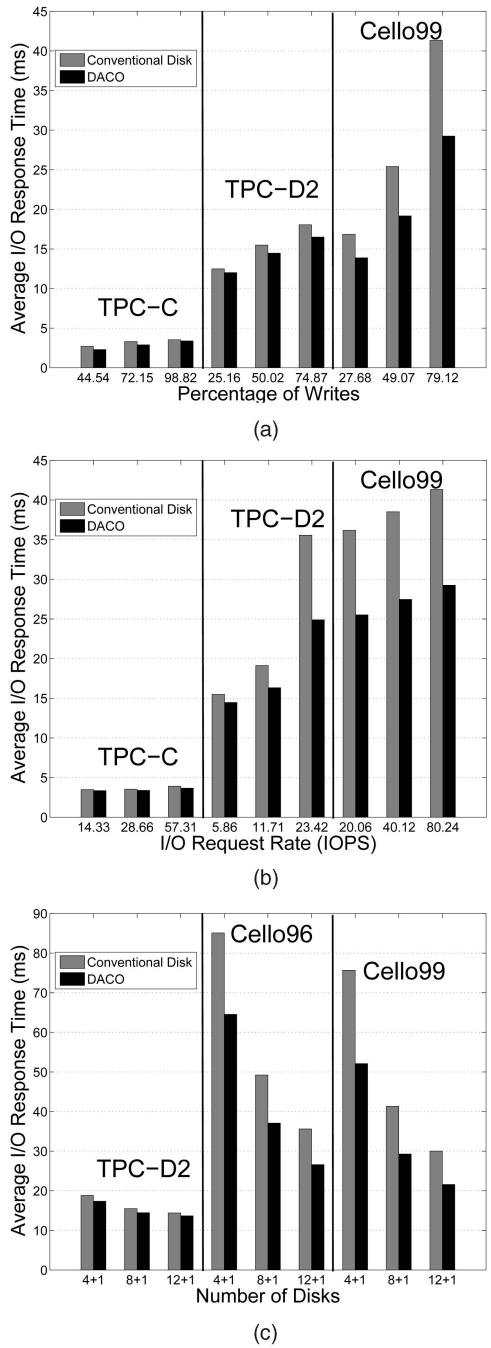

Fig. 9. The effects of different factors on the performance improvement of the DACO. (a) The effects of percentage of writes. (b) The effects of I/O request rate. (c) The effects of number of disks.

the organization of the erasure-coded disk array by varying the number of disks. Since the TPC-C traces cannot run on the disk array with $4+1$ Quantum Atlas 10K disks, we here chose the three traces: TPC-D2, Cello96, and Cello99.

Fig. 9c gives the simulation results for different numbers of disks. From this figure, we can see that with the number of disks decreasing, the DACO can reduce more average I/O response time. This change tendency is often very remarkable. For example, for the Cello99 traces, the average I/O response time reduced by the DACO in the disk array with $4+1$ disks is 23.58 ms, which is about three times as long as that reduced in the disk array with $12+1$ disks. Moreover, the improvement on the average I/O response time can increase up to 31.16 percent (in the case of the Cello99 traces running on the disk array with $4+1$ disks). This is because with the number of disks decreasing, the I/O throughput also decreases, and the application becomes more bottlenecked by I/O interface. Then, the performance benefit of the DACO will become more remarkable. This implies again that the DACO can provide more performance improvement in the application that are more bottlenecked by the storage subsystem.

In this section, we have demonstrated the performance potential of the DACO in the case where the fault tolerance is $t=1$. Numerical results have shown that the DACO can reduce the average I/O response time by up to 31.16 percent. We believe that these results for $t=1$ are enough for demonstrating the performance potential of the DACO. We thus have omitted the numerical results for $t>1$. In fact, as mentioned in Section 3.1, with the increase of the fault tolerance t, the proportion of disk media access operations reduced by the DACO in each small-write operation will increase from 25 percent approximately to 50 percent, and the performance benefit of the DACO will become more remarkable. Thus, if applied to large-scale erasure-coded storage systems with higher fault tolerance, the DACO can provide more performance improvement.

5 Implementation Issues

Having demonstrated the performance potential of the DACO, we now discuss some important implementation issues in this section.

5.1 Head Positioning Servomechanism

In the conventional disk, servo fields that contain positioning information are embedded on all tracks and are interleaved with data fields. During head positioning, a Magneto-Resistive (MR) sensor is used to sense the servo information from the servo fields. The servo information is then sent to the Voice Coil Motor (VCM) that is the torque producing component of the head positioning servomechanism [59].

In the DACO, to achieve the accurate positioning of the two heads that are mounted on the same arm, we propose a dual-stage servomechanism (see Fig. 10) that evolves from the dual actuator servo system [60] adopted in recent disk drives. Similar to the dual actuator servo system proposed in [60], our dual-stage servomechanism consists of a VCM as a first-stage actuator and two second-stage actuators. The

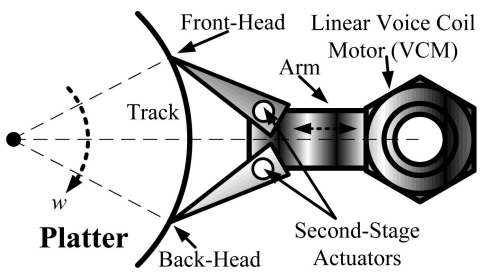

Fig. 10. Dual-stage servomechanism in the DACO.

VCM is a coarse actuator used for long-range seek. There are two types of VCM actuators proposed in the industry: linear VCM and rotary VCM [59]. We here choose the linear VCM for the DACO because its movement takes place along a radius of the disk, thus requiring a simpler calibration algorithm. The two second-stage actuators are finely positioning actuators designed for the two heads, respectively. As mentioned in [61], they could be piezoelectric actuators or micromachined actuators.

When a CO request arrives, the VCM uses the servo information gained from the front-head to move the arm along a radius of the disk. Because the front-head is a read head that can be used as a positioning sensor, no additional MR sensor is needed. After coarse positioning, the two heads are located around their expected positions. Then, the two second-stage actuators begin to accurately locate the two heads on their expected positions. During this process, the back-head that is a write head should also be capable of sensing the servo information. However, a write head is a Thin-Film Inductive (TFI) recording head that does not have this capability [59]. Thus, an additional MR sensor is assembled in the back-head. It should be noted that this additional MR sensor also enables the back-head (i.e., the write head) to seek and position without involving the front-head (i.e., the read head) during the WRITE operation.

5.2 Failure Handling during the CO

During the CO, various failures, which are caused by *latent sector errors*[8] [3], power-off problems, transient faults, and so on, may occur. Although their occurrence frequency is very low, they could still result in data corruption. In this subsection, we will develop some mechanisms to carefully handle these failures for different cases.

In the case where a failure occurs before the back-head begins to write the modified data, the CO will be canceled in the disk drive. If the failure is caused by a latent sector error, a data loss event will be reported to the host; else, the CO will be retried later for failure recovery.

However, in the case where a failure occurs after the back-head begins to write the modified data, since the CO is a *nonidempotent operation*, it cannot be retried, otherwise partially stale data will be produced (see Fig. 11). Then, the failure handling is more complex:

1. If the failure is caused by a power-off failure, the CO can be completed by supplying additional power. This can be easily implemented by embedding an additional small battery into the disk.
2. If the failure is caused by other problems (such as latent sector errors or transient faults), a data corruption event will be reported to the host, and the host will then handle it using high-level data redundancy mechanisms.

6 Cost Considerations

Since the significant performance benefit of the DACO is obtained by extending the conventional disk drive with additional hardware, we immediately raise a question: *How much additional hardware cost can be involved in the DACO?* Studies from the disk drive industry have shown that although building a disk drive involves material cost, also labor cost, and other overheads, the main portion of the disk manufacturing cost comes from the materials [51], [67], [68]. Thus, we focus on the hardware material cost. Note that since integrating the disk controller into the disk drive has become a popular trend in the disk drive industry, we propose that the DACO integrates its disk controller into its disk drive, and the following discussion on the disk drive thus will also involve the built-in disk controller. In the DACO, in order to support the CO, each platter surface has two associated heads: a front-head and a back-head. The front-head adopts only an MR sensor, while the back-head integrates both a TFI transducer and an MR sensor together. Compared with the conventional disk in which the one and only head is an integrated read/write head (that also includes a TFI transducer and an MR sensor), the DACO employs only an additional MR head, which is much simpler than an integrated read/write head, and thus, can involve much less additional cost than an additional integrated read/write head. Besides, to accurately position the additional head, an additional second-stage actuator is also involved in the dual-stage servomechanism (see Fig. 10). We believe that this dual-stage servomechanism can minimize the additional hardware cost caused by the additional head. Meanwhile, the disk controller should also be extended to manage and control the additional head and its corresponding second-stage actuator. This can be easily implemented by adding some additional control circuits, including some additional Digital Signal Processors (DSPs) to perform the additional computation, into the disk controller. In addition, in order to support the CO, a very simple computing element is also employed to implement the block-level data modify operation in the disk drive. In order to minimize the additional hardware cost and maximize the flexibility, this computing element can be implemented in one of the disk's built-in DSPs or microcontrollers. At the same time, the disk controller should also be extended to meet the real-time requirement in performing the CO. This can also be easily implemented by adding some additional control circuits (including some additional DSPs) into the disk controller. Finally, besides READ and WRITE commands, the interface of the disk controller should also be extended to include CO commands. Within the last 10-15 years, since the areal density of the disk increased by average 60-100 percent per year, the price per megabyte declined by average 37-50 percent per year,

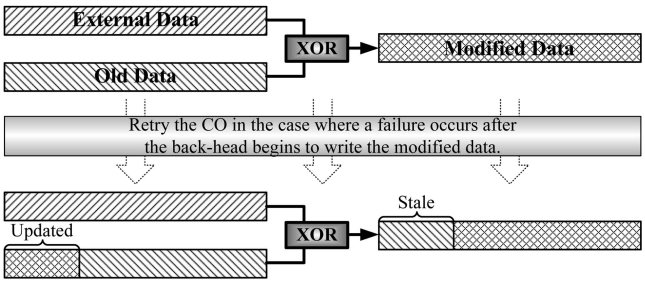

Fig. 11. The nonidempotent nature of the CO.

8. A *latent sector error* occurs when a particular disk sector cannot be read or written, or when there is an uncorrectable error detected by an Error-Correcting Code (ECC) or an iterative ECC (such as a Low-Density Parity Check (LDPC) code [62], [63], [64], [65], [66]). Any data previously stored in the sector is lost.

respectively [43]. Following this trend, the price per megabyte will decline to very low. Compared with the significant performance benefit of the DACO, the additional hardware cost involved in the DACO will become very acceptable.

In addition, unlike that of the conventional disk, the disk media access interface of the DACO consists of READ, WRITE, and CO. In order to adapt to this change, the software interface that communicates with the DACO also needs to be modified. However, since the CO is a block-level data operation, the introduction of the CO will not affect the implementations of the file system and the upper levels of the operating system. We need to only slightly modify the disk driver, by replacing each read-modify-write operation involved in small-write operations by a CO operation. This can be done simultaneously when we develop the disk driver for large-scale erasure-coded storage systems. Compared with the significant performance benefit of the DACO, the additional software cost caused by the DACO is also very acceptable.

Someone may also raise another question: *Is it worthy to spend the additional cost on the DACO, or is there any other more cost-effective choice available for large-scale erasure-coded storage systems?* As mentioned in Section 2.1, large-scale erasure-coded storage systems have a very serious performance problem due to I/O congestion and disk media access congestion caused by small-write operations, and with the increase of their fault tolerance, this small-write problem will become more serious and can hardly be solved using the existing technologies based on the conventional disk. Under this background, we propose the DACO that extends the conventional disk with additional hardware to solve this small-write problem. A possible alternative is to use Solid-State Disks (SSDs), such as flash [69] or MicroElectroMechanical System (MEMS) [70]. Since the parity update is the major cause of the small-write problem, we could use special SSDs to store the parity. However, as mentioned in [51], the cost per megabyte of flash and MEMS are orders of magnitude higher than hard disks. Thus, it is much more cost-effective to extend the conventional disk to solve the small-write problem. Moreover, since flash easily wears out as we increase the number of write operations [71], it is not suitable for storing the parity that is frequently updated. Therefore, we believe that the DACO is the best choice for large-scale erasure-coded storage systems.

7 Conclusions

Erasure-coding technologies have recently been widely adopted for fault tolerance in large-scale disk-based storage systems. However, large-scale erasure-coded storage systems have a serious performance problem due to I/O congestion and disk media access congestion caused by small-write operations, and with the increase of their fault tolerance, this small-write problem will become more serious and can hardly be solved using the existing technologies based on the conventional disk. In this paper, in order to solve this small-write problem, we present a Disk Architecture with Composite Operation (DACO) designed specially for large-scale erasure-coded storage systems. When we use the DACO in a large-scale erasure-coded storage system with t fault tolerance, t I/Os and t disk media access operations can be reduced in each small-write operation, respectively. This alleviates both I/O congestion and disk media access congestion caused by small-write operations in nature, and thus, can remarkably improve the performance of large-scale erasure-coded storage systems. A simulation study shows that the DACO can provide significant performance improvement: reducing the average I/O response time by up to 31.16 percent even in the worst case where $t = 1$. We believe that if applied to large-scale erasure-coded storage systems with higher fault tolerance, the DACO can provide more performance improvement. In this paper, we also discuss two important implementation issues of the DACO: head positioning servomechanism and failure handling during the CO. Finally, we investigate the additional cost involved in the DACO and then claim that the DACO is the most cost-effective choice for large-scale erasure-coded storage systems.

It is to be noted that the DACO proposed in this paper can also be extended for reducing the performance loss caused by the introduction of other kinds of block-level technologies, such as block-level security technologies [56]. This will be studied in our future work.

Acknowledgments

The authors would like to thank the anonymous reviewers and the Associate Editor Dr. Mazin Yousif for their constructive and valuable comments that helped in improving this paper. This work was supported by the National Natural Science Foundation of China (Grant No. 60873066), the National High Technology Research and Development Program of China (Grant No. 2009AA01Z139), and the Specialized Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20070003092).

References

[1] E. Pinheiro, W.D. Weber, and L.A. Barroso, "Failure Trends in a Large Disk Drive Population," *Proc. USENIX Conf. File and Storage Technologies (FAST '07)*, Feb. 2007.

[2] B. Schroeder and G.A. Gibson, "Disk Failures in the Real World: What Does an MTTF of 1,000,000 Hours Mean to You?," *Proc. USENIX Conf. File and Storage Technologies (FAST '07)*, Feb. 2007.

[3] L.N. Bairavasundaram, G.R. Goodson, S. Pasupathy, and J. Schindler, "An Analysis of Latent Sector Errors in Disk Drives," *Proc. SIGMETRICS '07*, June 2007.

[4] A. Dholakia, E. Eleftheriou, X.-Y. Hu, I. Iliadis, J. Menon, and K.K. Rao, "A New Intra-Disk Redundancy Scheme for High-Reliability RAID Storage Systems in the Presence of Unrecoverable Errors," *ACM Trans. Storage*, vol. 4, no. 1, pp. 1-42, May 2008.

[5] I. Iliadis, R. Haas, X.-Y. Hu, and E. Eleftheriou, "Disk Scrubbing Versus Intra-Disk Redundancy for High-Reliability RAID Storage Systems," *Proc. SIGMETRICS '08*, June 2008.

[6] J.S. Plank, "Erasure Codes for Storage Applications," Tutorial Slides, FAST 2005, Dec. 2005.

[7] P.M. Chen, E.K. Lee, G.A. Gibson, R.H. Katz, and D.A. Patterson, "RAID: High-Performance, Reliable Secondary Storage," *ACM Computing Surveys*, vol. 26, no. 2, pp. 145-185, June 1994.

[8] C. Carlane and A. Osuna, "IBM System Storage *N* Series Implementation of RAID Double Parity for Data Protection," IBM Redpaper REDP-4169-00, http://www.redbooks.ibm.com/redpapers/pdfs/redp4169.pdf, Apr. 2006.

[9] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "OceanStore: An Architecture for Global-Scale Persistent Storage," *Proc. Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS '00)*, Nov. 2000.

[10] A. Haeberlen, A. Mislove, and P. Druschel, "Glacier: Highly Durable, Decentralized Storage Despite Massive Correlated Failures," *Proc. Second Conf. Symp. Networked Systems Design and Implementation (NSDI '05),* May 2005.

[11] S. Frolund, A. Merchant, Y. Saito, S. Spence, and A. Veitch, "A Decentralized Algorithm for Erasure-Coded Virtual Disks," *Proc. Int'l Conf. Dependable Systems and Networks (DSN '04),* June 2004.

[12] G.R. Goodson, J.J. Wylie, G.R. Ganger, and M.K. Reiter, "Efficient Byzantine-Tolerant Erasure-Coded Storage," *Proc. Int'l Conf. Dependable Systems and Networks (DSN '04),* June 2004.

[13] J. Hendricks, G.R. Ganger, and M.K. Reiter, "Low-Overhead Byzantine Fault-Tolerant Storage," *Proc. 21st ACM Symp. Operating Systems Principles (SOSP '07),* Oct. 2007.

[14] H. Xia and A.A. Chien, "RobuSTore: A Distributed Storage Architecture with Robust and High Performance," *Proc. Conf. Supercomputing (SC '07),* Nov. 2007.

[15] M.W. Storer, K.M. Greenan, E.L. Miller, and K. Voruganti, "Pergamum: Replacing Tape with Energy Efficient, Reliable, Disk-Based Archival Storage," *Proc. USENIX Conf. File and Storage Technologies (FAST '08),* Feb. 2008.

[16] Cleversafe, Inc., "Cleversafe Dispersed Storage," Open Source Code Distribution, http://www.cleversafe.org/downloads, 2009.

[17] Allmydata, Inc., "Unlimited Online Backup, Storage, and Sharing," http://www.allmydata.com/, 2009.

[18] Permabit Technology Corporation, "Disk Based Enterprise Archive, Data Archiving Solutions," http://www.permabit.com/, 2009.

[19] R.E. Bryant, "Data Intensive Supercomputing: The Case for DISC," Technical Report CMU-CS-07-128, School of Computer Science, Carnegie Mellon Univ., May 2007.

[20] J. Menon, J. Roche, and J. Kasson, "Floating Parity and Data Disk Arrays," *J. Parallel and Distributed Computing,* vol. 17, nos. 1/2, pp. 129-139, Jan./Feb. 1993.

[21] D. Stodolsky, M. Holland, W.V. Courtright, and G.A. Gibson, "Parity-Logging Disk Arrays," *ACM Trans. Computer Systems,* vol. 12, no. 3, pp. 206-235, Aug. 1994.

[22] R.A. Demoss and K.B. Dulac, "Delayed Initiation of Read-Modify-Write Parity Operations in a Raid Level 5 Disk Array," US Patent No. 5388108, Feb. 1995.

[23] G. Houlder, J. Elrod, and M. Miller, "XOR Commands on SCSI Disk Drives," ANSI Specification X3T10/94-111r9, http://www.t10.org/ftp/t10/document.94/94-111r9.pdf, 1994.

[24] R.A. DeKoning, "Method for Performing a RAID Stripe Write Operation Using a Drive XOR Command Set," U.S. Patent No. 5742752, Apr. 1998.

[25] G.R. Ganger, "Blurring the Line Between OSes and Storage Devices," Technical Report CMU-CS-01-166, School of Computer Science, Carnegie Mellon Univ., Dec. 2001.

[26] R.A. Uhlig and T.N. Mudge, "Trace-Driven Memory Simulation: A Survey," *ACM Computing Surveys,* vol. 29, no. 2, pp. 128-170, June 1997.

[27] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes.* Elsevier, 1977.

[28] I.S. Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," *J. Soc. for Industrial and Applied Math.,* vol. 8, no. 2, pp. 300-304, June 1960.

[29] R.R. Roth and A. Lempel, "On MDS Codes via Cauchy Matrices," *IEEE Trans. Information Theory,* vol. 35, no. 6, pp. 1314-1319, Nov. 1989.

[30] J. Blomer, M. Kalfane, M. Karpinski, R. Karp, M. Luby, and D. Zuckerman, "An XOR-Based Erasure-Resilient Coding Scheme," Technical Report TR-95-048, Int'l Computer Science Inst., Aug. 1995.

[31] J.S. Plank and L. Xu, "Optimizing Cauchy Reed-Solomon codes for Fault-tolerant Network Storage Applications," *Proc. IEEE Int'l Symp. Network Computing and Applications (NCA '06),* July 2006.

[32] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures," *IEEE Trans. Computers,* vol. 44, no. 2, pp. 192-202, Feb. 1995.

[33] L. Xu and J. Bruck, "X-Code: MDS Array Codes with Optimal Encoding," *IEEE Trans. Information Theory,* vol. 45, no. 1, pp. 272-276, Jan. 1999.

[34] C. Huang and L. Xu, "STAR: An Efficient Coding Scheme for Correcting Triple Storage Node Failures," *Proc. USENIX Conf. File and Storage Technologies (FAST '05),* Dec. 2005.

[35] J.S. Plank, "The RAID-6 Liberation Codes," *Proc. USENIX Conf. File and Storage Technologies (FAST '08),* Feb. 2008.

[36] J.L. Hafner, "WEAVER Codes: High Fault Tolerant Erasure Codes for Storage Systems," *Proc. USENIX Conf. File and Storage Technologies (FAST '05),* Dec. 2005.

[37] M. Li, J. Shu, and W. Zheng, "GRID Codes: Strip-Based Erasure Codes with High Fault Tolerance for Storage Systems," *ACM Trans. Storage,* vol. 4, no. 4, pp. 1-22, Jan. 2009.

[38] K. Haughton, "Design Considerations in the IBM 3340 Disk File," *Proc. IEEE Computer Soc. Conf.,* Feb. 1974.

[39] R.B. Mulvany, "Engineering Design of a Disk Storage Facility with Data Modules," *IBM J. Research and Development,* vol. 18, no. 6, pp. 489-505, 1974.

[40] A.J. Smith, "On the Effectiveness of Buffered and Multiple Arm Disks," *Proc. Int'l Symp. Computer Architecture (ISCA '78),* Apr. 1978.

[41] J.M. Harker, D.W. Brede, R.E. Pattison, G.R. Santana, and L.G. Taft, "A Quarter Century of Disk File Innovation," *IBM J. Research and Development,* vol. 25, no. 5, pp. 677-689, Sept. 1981.

[42] J.P. Squires, G.N. Bagnell, C.M. Sander, and K.M. Anderson, "Multiple Actuator Disk Drive," U.S. Patent No. 5293282, Mar. 1994.

[43] E. Grochowski and R.D. Halem, "Technological Impact of Magnetic Hard Disk Drives on Storage Systems," *IBM Systems J.,* vol. 42, no. 2, pp. 338-346, Apr. 2003.

[44] P. Gilovich, "Multiple Actuator Assemblies for Data Storage Devices," U.S. Patent No. 6057990, May 2000.

[45] A.R. Howard, "Disk Data Storage Apparatus and Method Using Multiple Head Actuators," U.S. Patent No. 7102842, Sept. 2006.

[46] N.-K. Lee, T.-D. Han, S.-D. Kim, and S.-B. Yang, "High Performance RAID System by Using Dual Head Disk Structure," *Proc. Eighth Int'l Conf. High-Performance Computing in Asia-Pacific Region (HPC-Asia '97),* Apr. 1997.

[47] N.-K. Lee, S.-B. Yang, T.-D. Han, and S.-D. Kim, "Modeling and Performance Analysis of Dual Head Disk Structure," *J. Systems Architecture,* vol. 44, nos. 9/10, pp. 787-802, June 1998.

[48] R. Wood, J. Miles, and T. Olson, "Recording Technologies for Terabit Per Square Inch Systems," *IEEE Trans. Magnetics,* vol. 38, no. 4, pp. 1711-1718, July 2002.

[49] J. Zheng, G. Guo, and Y. Wang, "Feedforward Decoupling Control Design for Dual-Actuator System in Hard Disk Drives," *IEEE Trans. Magnetics,* vol. 40, no. 4, pp. 2080-2082, July 2004.

[50] J.A. Chandy, "Dual Actuator Logging Disk Architecture and Modeling," *J. Systems Architecture,* vol. 53, no. 12, pp. 913-926, Dec. 2007.

[51] S. Sankar, S. Gurumurthi, and M.R. Stan, "Intra-Disk Parallelism: An Idea Whose Time Has Come," *Proc. Int'l Symp. Computer Architecture (ISCA '08),* June 2008.

[52] R. Galbraith and T. Oenning, "Iterative Detection Read Channel Technology in Hard Disk Drives," Hitachi White Paper WPIDRC08EN-01, http://www.hitachigst.com/tech/techlib.nsf/techdocs/FB376A33027F5A5F86257509001463AE/$file/IDRC_WP_final.pdf, Oct. 2008.

[53] R. Wood, "Future Hard Disk Drive Systems," *J. Magnetism and Magnetic Materials,* vol. 321, no. 6, pp. 555-561, Mar. 2009.

[54] Seagate Technology LLC, "The Seagate Cheetah Hard Drive Family," http://www.seagate.com/www/en-us/products/servers/cheetah/, 2009.

[55] S.W. Ng, "Advances in Disk Technology: Performance Issues," *Computer,* vol. 31, no. 5, pp. 75-81, May 1998.

[56] M.K. Aguilera, M. Ji, M. Lillibridge, J. MacCormick, E. Oertli, D. Andersen, M. Burrows, T. Mann, and C.A. Thekkath, "Block-Level Security for Network-Attached Disks," *Proc. USENIX Conf. File and Storage Technologies (FAST '03),* Apr. 2003.

[57] J.S. Bucy, G.R. Ganger, and Contributors, "The Disksim Simulation Environment Version 3.0 Reference Manual," Technical Report CMU-CS-03-102, School of Computer Science, Carnegie Mellon Univ., Jan. 2003.

[58] Storage Systems Department at HP Labs, "Trace Data," Open Source Software, http://tesla.hpl.hp.com/opensource/, 2009.

[59] A.A. Mamun, G. Guo, and C. Bi, *Hard Disk Drive: Mechatronics and Control.* CRC Press, 2007.

[60] J. Ding, S.-C. Wu, and M. Tomizuka, "Settling Control with Reference Redesign for Dual Actuator Hard Disk Drive Systems," *Ann. Rev. in Control,* vol. 28, no. 2, pp. 219-227, Sept. 2004.

[61] D. Abramovitch and G. Franklin, "A Brief History of Disk Drive Control," *IEEE Trans. Automatic Control,* vol. 22, no. 3, pp. 28-42, June 2002.
[62] R.G. Gallager, *Low-Density Parity-Check Codes,* Monograph. MIT Press, 1963.
[63] M.G. Luby, M. Mitzenmacher, A. Shokrollahi, and D.A. Spielman, "Efficient Erasure Correcting Codes," *IEEE Trans. Information Theory,* vol. 47, no. 2, pp. 569-584, Feb. 2001.
[64] R.M. Tanner, "A Recursive Approach to Low-Complexity Codes," *IEEE Trans. Information Theory,* vol. 27, no. 5, pp. 533-547, Sept. 1981.
[65] J.S. Plank and M.G. Thomason, "A Practical Analysis of Low-Density Parity-Check Erasure Codes for Wide Area Storage Applications," *Proc. Int'l Conf. Dependable Systems and Networks (DSN '04),* June 2004.
[66] J.S. Plank, R.L. Collins, A.L. Buchsbaum, and M.G. Thomason, "Small Parity-Check Erasure Codes-Exploration and Observations," *Proc. Int'l Conf. Dependable Systems and Networks (DSN '05),* June 2005.
[67] S. Hampton, "Process Cost Analysis for Hard Disk Manufacturing," Technical Report 96-02, Information Storage Industry Center, Univ. of California, Sept. 1996.
[68] R. Bohn and C. Terwiesch, "The Economics of Yield-Driven Processes," *J. Operations Management,* vol. 18, no. 1, pp. 41-59, Dec. 1999.
[69] M. Wu and W. Zwaenepoel, "ENVy: A Non-Volatile, Main Memory Storage System," *Proc. Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS VI),* Oct. 1994.
[70] S.W. Schlosser, J.L. Griffin, D. Nagle, and G.R. Ganger, "Designing Computer Systems with MEMS-Based Storage," *Proc. Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS IX),* Nov. 2000.
[71] T. Kgil, D. Roberts, and T. Mudge, "Improving NAND Flash Based Disk Caches," *Proc. Int'l Symp. Computer Architecture (ISCA '08),* June 2008.

Mingqiang Li received the BSc degree in mathematics from the University of Electronic Science and Technology of China in 2006 and is currently working toward the PhD degree in the Department of Computer Science and Technology at Tsinghua University. His area of interests include storage systems, coding theory, and distributed computing. He is a student member of the IEEE and the IEEE Computer Society.

Jiwu Shu received the PhD degree in computer science from Nanjing University in 1998. He finished his postdoctoral research at Tsinghua University in 2000. Since then, he has been working as a teacher at Tsinghua University. He is now a professor in the Department of Computer Science and Technology at Tsinghua University. His area of interests include storage systems and parallel computing. He is a member of the IEEE and the IEEE Computer Society.