MICRO: A Multilevel Caching-Based Reconstruction Optimization for Mobile Storage Systems

Tao Xie, *Member*, *IEEE*, and Hui Wang

Abstract—High performance, highly reliable, and energy-efficient storage systems are essential for mobile data-intensive applications such as remote surgery and mobile data center. Compared with conventional stationary storage systems, mobile disk-array-based storage systems are more prone to disk failures due to their severe application environments. Further, they have a very limited power supply. Therefore, data reconstruction algorithms which are executed in the presence of disk failure for mobile storage systems must be performance-driven, reliability-aware, and energy-efficient. Unfortunately, existing reconstruction schemes cannot fulfill these three goals simultaneously because they largely overlook the fact that mobile disks have much higher failure rates than stationary disks. Besides, they normally ignore energy saving. In this paper, we develop a novel reconstruction strategy, called multilevel caching-based reconstruction optimization (MICRO), which can be applied to RAID-structured mobile storage systems to noticeably shorten reconstruction times and user response times while saving energy. MICRO collaboratively utilizes storage cache and disk array controller cache to diminish the number of physical disk accesses caused by reconstruction. Experimental results demonstrate that, compared with two representative algorithms, DOR and PRO, MICRO reduces reconstruction times on average by 20.22 percent and 9.34 percent, while saving energy by no less than 30.4 percent and 13 percent, respectively.

Index Terms—Reconstruction algorithm, mobile storage system, multilevel caching, energy efficiency, RAID structure.

1 Introduction

Traditionally, disk-array-based storage systems are stationary in the sense that they are normally installed in data centers where air conditioning, uninterrupted power supplies, backup systems, and physical security are provided. Still, there are many situations where mobile disk-array-based storage systems are gradually becoming desirable and even indispensable. Here, we define a mobile storage system as an array of small form factor hard disks organized in RAID structures and connected to a host by a storage interface like serial-attached SCSI (SAS) in a mobile computing environment. Due to their severe application environments, mobile storage systems must be energy-efficient, extremely reliable, highly fault-tolerant, and physically robust.

A typical application scenario of mobile storage system is a mobile data center. Mobile data centers are an alternative to conventional stationary data centers that are enclosed in buildings. They could be built on self-contained trucks, airplanes, or ships that contain onboard generators, UPS, multiple high-capacity servers, and satellite Internet links [23]. Obviously, mobile storage systems located in mobile data centers are essential for applications such as disaster recovery, live video broadcast [20], and homeland security. For example, an NAAT Mobile Emergency Datacenter (MED) can accommodate up to 100 fully charged laptops, multiple high-performance servers, and a large capacity storage system with multiple terabytes of data in a 20-25 ft truck [23]. In this emergency-oriented application, mobile storage systems are indispensable because they can provide not only huge storage capacities but also quick response times. In fact, micro hard disk drives for mobile storage systems recently emerged in the HDD market. For instance, an Imation Micro Hard Drive can provide up to a 4 Gbyte capacity with a 0.85 in diameter and 0.84 oz weight [31] and a Seagate ST1 1-in Hard Disk Drive can offer up to a 12 Gbyte capacity [2], even though the applications of mobile disk arrays illustrated above are still in their infancy in terms of development and deployment. Nevertheless, we believe that they will become prevalent in the not-so-distant future due to the real needs of mobile data-intensive applications, high wireless network bandwidth, and advances of hard disk technology.

Mobile disk array, however, faces several new challenges that were not encountered by their stationary counterparts before, as explained below:

- **Stringent reliability requirements.** The reliability requirements for both mobile disk arrays and the data they store are much higher than stationary disk arrays. Data sampled from mobile and dynamic environments is most likely irreproducible and, thus, data loss is completely unacceptable. To improve data reliability, an efficient data recovery mechanism is a must. Besides, no or a very small number of spare disks can be carried in a mobile system due to the limited space. As a result, it is very

- *T. Xie is with the Department of Computer Science, San Diego State University, 5500 Campanile Drive, GMCS 544, San Diego, CA 92182. E-mail: xie@cs.sdsu.edu.*
- *H. Wang is with Ortiva Wireless Inc., 4225 Executive Sq., La Jolla, CA 92037. E-mail: jackawang@gmail.com.*

Manuscript received 22 July 2007; revised 5 Jan. 2008; accepted 14 Apr. 2008; published online 25 Apr. 2008.
Recommended for acceptance by B. Veeravalli.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2007-07-0360.
Digital Object Identifier no. 10.1109/TC.2008.76.

difficult, if not impossible, to obtain a backup disk once a mobile disk is corrupt.

- **High demands on fault tolerance.** Mobile disk arrays are more prone to failures than stationary ones because of their harsh application and operational environments. Consequently, mobile disk arrays should be able to gracefully degrade their performance in the event of the failure of some of their components.
- **Very limited power supply.** In contrast to traditional static storage systems located in data center buildings where electrical power is guaranteed, mobile storage systems only have a very limited power supply provided by either gasoline generators or batteries. Although disk arrays, mobile or not, demand energy conservation to save on the cost of electricity and cooling and to reduce the impact of high operating temperatures on the stability and reliability of the components [9], energy saving becomes more critical for mobile storage systems because their energy consumption can significantly affect the lifetime of the entire mobile systems of which they are a part.

Unfortunately, existing reconstruction algorithms reported in the literature [11], [18], [32] targeted only traditional stationary storage systems and thus largely overlooked the three new challenges imposed by mobile storage systems. Although they concentrate on minimizing the reconstruction time and alleviating performance degradation during the recovery process, they are inadequate for mobile storage systems. The reason is twofold. First, they are not aware of the fact that mobile disks have much higher failure rates than stationary disks. Consequently, the length of the reconstruction time (or "window of vulnerability") they achieved might not be sufficiently short for mobile storage systems, where the probability of a subsequent disk failure during a reconstruction process is not negligible. Second, they normally ignore energy saving. Although disk recovery is not a frequent event compared with normal operating mode, three facts, however, can dramatically increase the occurrence of the recovery mode. The first fact is that the more than one million hours datasheet Mean Time Between Failure (MTBF) specified by disk manufacturers is unrealistic. Schroeder and Gibson found that the annual disk replacement rates in the field are usually in the range of 2 percent to 4 percent and up to 13 percent, which are much higher than than manufacturers' datasheet annual failure rate (e.g., 0.88 percent for disks with 1,000,000 hours MTBF) [27]. The second fact is that, compared with their static counterparts, mobile disk arrays generally operate in a much worse environment, which could result in an even higher annual disk replacement rate. The third fact is that, nowadays, computer systems grow larger and larger and, thus, storage systems scale up significantly. A large-scale storage system with tens of thousands or more disks could expect a very high overall failure rate [32]. For example, in a petabyte-scale file system, disk failure will be a daily occurrence [35]. Thus, we argue that disk recovery is not a rare case in large-scale mobile storage systems. Moreover, the time to rebuild a single disk has lengthened and can be on the order of several hours or even longer [27] as increases in disk capacity far outpace increases in disk bandwidth [32]. Therefore, disk recovery, a nonrare and slow event, warrants an investigation in energy saving. In summary, a new high-performance, reliability-aware, and energy-efficient data construction strategy is needed for mobile storage systems.

In this paper, we propose a novel reconstruction strategy, called multilevel caching-based reconstruction optimization (MICRO), which can be applied to RAID-structured mobile storage systems to noticeably shorten reconstruction times and user response times during disk recovery while saving energy. The basic idea of MICRO is to collaboratively utilize storage cache and disk array controller cache to diminish the number of physical disk accesses caused by reconstruction. In addition, MICRO leverages on recent request access locality information, which is periodically collected before disk failure, to predict the workload access pattern during the reconstruction process. To the best of our knowledge, little research work has been directed toward integrating multilevel caching into the reconstruction algorithm for mobile storage systems. Most of the existing reconstruction algorithms perform either stripe-oriented [14] or disk-oriented [11], [32] parallel reconstruction processes. Exploiting the multilevel caches provided in modern storage systems, the MICRO strategy reconstructs popular data of the failed disk without accessing surviving disks and thus obtains a shorter reconstruction time, a more graceful performance degradation, and energy efficiency.

Extensive experimental results demonstrate that, compared with existing data construction algorithms, MICRO leads to a much quicker reconstruction time and a shorter user response time during disk recovery while noticeably saving energy. Specifically, compared with the well-known algorithm Disk-Oriented Reconstruction (DOR) [11] and a state-of-the-art scheme Popularity-based Multithreaded Reconstruction (PRO) [32], MICRO reduces reconstruction time by up to 20.46 percent and 9.61 percent, decreases mean response time during recovery on average by 46.59 percent and 26.91 percent, while saving energy by no less than 30.4 percent and 13 percent, respectively. The rest of this paper is organized as follows: In Section 2, we discuss the related work and motivation. In Section 3, we describe the design and implementation of the MICRO strategy. In Section 4, we evaluate the performance of MICRO based on synthetic benchmarks. Section 5 concludes this paper with a summary and future directions.

2 Related Work and Motivation

In this section, we first present the problem of data reconstruction and its current solutions. Next, we discuss multilevel cache architecture and its applications in high performance storage systems. Last, we clarify the motivation of our research.

2.1 Existing Approaches

When a disk failure occurs, a parity-encoding-based RAID-structured disk array can restore to the normal operating

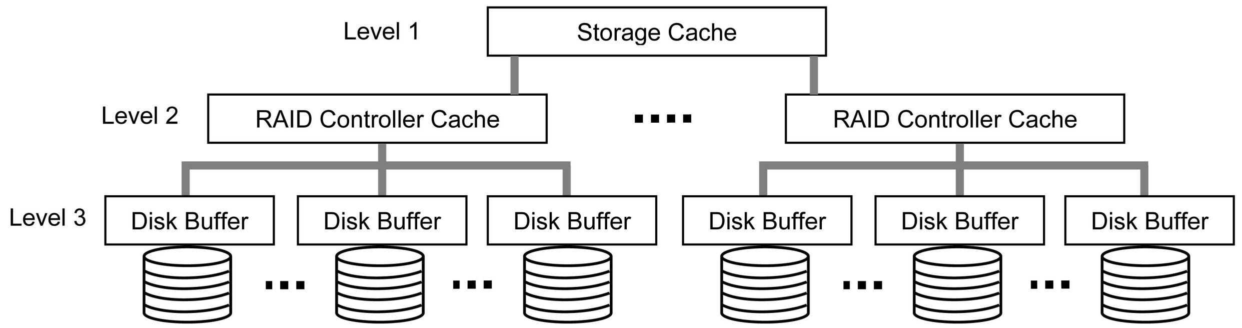


Fig. 1. Multilevel caches in a storage system.

state by successively rebuilding each block of the failed disk onto a replacement drive while it continues to serve I/O requests from users. This process is called reconstruction, which is normally performed by a background process activated in either the host or the array controller [13]. Since the efficiency of reconstruction algorithms not only affects the performance but also the reliability of storage systems [35], much effort has been devoted to the development of effective reconstruction schemes to maintain system reliability and to minimize the performance degradation [11], [13], [14], [18], [32]. Existing solutions to the reconstruction problem generally fall into two categories: data layout reorganization and reconstruction workflow optimization [32].

Traditional data layouts like RAID-5 significantly degrade performance in terms of response times during reconstruction process because the workload of each surviving disk increases by 100 percent during disk recovery [13]. To solve this problem, Muntz and Lui [24] suggested a data placement scheme called declustering, where each stripe is mapped to just k of the n disks in an array ($k \leq n$). The declustering scheme largely improves performance during both degraded operation and online disk reconstruction because a smaller number of stripe units need to be read during the reconstruction process. Further, Holland and Gibson [12] evaluated the declustering strategy by identifying six desirable properties for ideal layouts. Based on the six properties, Alvarez et al. presented a complete characterization of the collection of ideal declustered layouts possessing all six properties [1]. In addition, they developed two novel layout algorithms, PRIME and PELPR, which can tolerate multiple failures in a wide range of configurations. Considering large-scale distributed storage systems, Xin et al. presented FARM, a distributed recovery approach that exploits excess disk capacity and reduces data recovery time [34]. In addition, they examined essential factors that influence storage system reliability, performance, and cost.

The second category of existing reconstruction algorithms strives to improve reliability and alleviate performance degradation by optimizing reconstruction workflow. Compared with approaches in the first category, schemes in this camp possess an obvious advantage because there is no need for them to alter the data layout of widely used RAID installations. Stripe-Oriented Reconstruction (SOR) [13] and DOR [11] are two representative approaches in this category. Although both algorithms exploit parallelism to speed up the reconstruction process, the parallel processes they generate target different sources. Specifically, SOR creates a set of reconstruction processes associated with stripes, whereas DOR generates a group of processes with each corresponding to one disk. Holland et al. [13] demonstrate that DOR outperforms SOR in failure recovery time with only a small degradation in user response time during failure recovery. The improvement of DOR comes from a much more efficient utilization of the disk array's excess disk bandwidth. Sivathanu et al. designed D-GRAID, a gracefully degrading and quickly recovering RAID storage array which ensures that most files within the file system remain available even when an unexpectedly high number of faults occur [30]. Very recently, Tian et al. [32] proposed and evaluated a novel dynamic data construction optimization algorithm, PRO, which allows the reconstruction process to rebuild the frequently accessed areas prior to rebuilding infrequently accessed areas to exploit access locality. Essentially, the PRO scheme integrates workload characteristics into workflow-based reconstruction process to accomplish improvement of reliability and system performance simultaneously.

2.2 Multilevel Cache in Storage Systems

Hierarchical cache subsystems, illustrated in Fig. 1, are typical in modern storage systems for reliability and performance purposes [22]. Generally, in a large-scale storage system, there are three levels of caches: the storage cache attached to a storage server, the disk array controller cache associated with an array controller, and the disk drive onboard buffer [16]. The storage cache is a nonvolatile array of fast RAM that interfaces with other storage devices through multiple high-performance interconnects, such as Fiber Channel links, and its size varies from several gigabytes to 128 Gbytes [38]. While the size of disk array controller cache is normally in the range from 64 to 512 Mbytes [37], disk drives like SCSI disks only have a 1-4 Mbyte onboard buffer [38].

The majority of the existing work on multilevel storage cache architectures focuses on the collaboration between caches in client-side, like the IBM DB2 database server and storage cache (level 1 in Fig. 1) to improve performance [6], [33]. A common goal for these studies is to achieve *exclusive caching*, in which data is cached at either a client or the storage system but not both [33]. The first work integrating storage cache with disk energy saving [38] proposed two energy-aware cache replacement algorithms, PA-LRU and

PB-LRU, which use a single level storage cache to enlarge the idle time interval of a standby disk in a disk array. Only a little recent research targets energy saving by collaboratively using multilevel caches in storage systems. Yao and Wang presented a redundancy-based two-level I/O cache architecture called RIMAC to save energy without compromising performance [37]. In addition, they proposed two energy-aware read request transformation schemes (TRC and TRD), which can be applied to optimize the disk I/O access pattern such that the idle period of the standby disk can be elongated for significant energy savings.

2.3 Motivations

Although a number of studies reported in the literature have concentrated on how to improve storage system reliability, performance, and energy efficiency by utilizing caches in either single level or multilevel [6], [33], [37], [38], none of them attempted to address the data reconstruction issue by employing multilevel storage caches. On the other hand, existing reconstruction algorithms mainly focus on optimizing data layouts or reconstruction workflow, which are all essentially disk-oriented in the sense that the read/write access requests they generated during recovery are served by physical disks. Based on insightful observations made by the research work in multilevel storage caching subsystems discussed above and by our own research, we believe that collaborative utilization of multilevel storage caches is a new avenue to solving the data reconstruction problem because it can noticeably diminish the number of disk accesses caused by disk recovery, thus enhancing performance, improving reliability, and saving energy.

3 Design and Implementation of MICRO

In this section, we first present an overview of the architecture of a multilevel cache subsystem used by our MICRO approach, which is followed by a detailed algorithm description as well as a complexity analysis of MICRO.

3.1 Architecture Overview

Since the onboard buffer on current disk drives is very small (from 64 Kbytes to 1 Mbyte), MICRO only uses the level 1 (storage cache) and the level 2 (controller cache) caches illustrated in Fig. 1. In a modern storage system like EMC Symmetrix 5000 Enterprise Storage System, up to 128 Gbyte nonvolatile memory can be configured as the storage cache [8]. Meanwhile, the size of the disk array controller cache normally falls into the range from 64 Mbytes to several gigabytes. Considering that the storage cache will be shared by multiple disk arrays, MICRO evenly divides it into multiple partitions, with each being dedicated to one disk array. Based on real-world settings, for each disk array we assume that the total size of the two-level caches varies from 1/4 to 4 Gbytes with the fixed size of the level 2 cache being 128 Mbytes. In addition, we consider a RAID-5 structure with read-only requests, which is common in the Web search workload [29].

Fig. 2 displays the data organization in RAID-5 and the cache placement scheme employed by MICRO. The size of a basic unit to store data in the two levels of caches is configured as the same size as a data unit in a RAID-5 disk

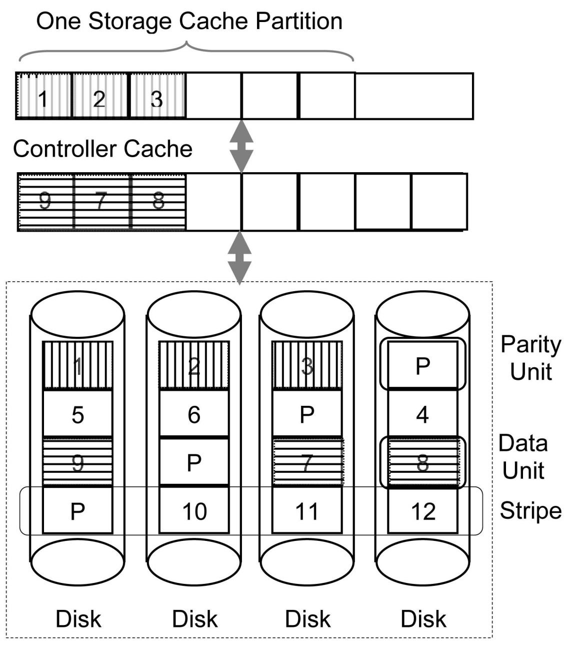


Fig. 2. MICRO cache usage example.

array. MICRO utilizes a request-driven storage cache placement policy, which was also used in [37]. Only the data units requested by users are placed into the storage cache (level one). Meanwhile, a controller cache is used as a second-level cache of its corresponding "storage cache partition" in the storage cache (see Fig. 2). Since we only consider read-only applications, there is no need to read the parity data. As a result, only data units are stored in the two-level caches. In this paper, each access to a file is a sequential read of the entire file, which is a typical scenario in most file systems or WWW servers [17]. In addition, a user access is mapped to multiple stripes. In the normal operation mode, if all data units necessary to satisfy a host I/O request are in the storage cache, a *Level One Read Hit* occurs and the data units are returned to the host without searching the controller cache. Otherwise, the controller cache will be searched to find the missing data units. In the case where all missing data units are found in the second-level cache, a *Level Two Read Hit* happens and the data units are escalated to the storage cache. If the data units needed are still missing from the controller cache, a *Read Miss* occurs and the corresponding data is fetched by accessing physical disks. In this situation, the data units from disks are stored in the storage cache for future use. Both the storage cache and the controller cache employ the simple least recent used (LRU) replacement policy. When there is no free room to accommodate a new data unit in the storage cache, the LRU data unit in the storage cache will be moved to the controller cache. Similarly, when there is no free space to hold newly incoming data units from the storage cache, the controller cache will evict its LRU data units. While data units in the storage cache are the most popular data requested by clients, data stored in the controller cache is second in terms of popularity. In Fig. 2, we can see that the data units of the first stripe (units 1, 2, 3) are in the storage cache, whereas data units 9, 7, 8 of the third stripe reside in the controller cache. The capacity of a storage

TABLE 1
A Sample File Popularity Table

File ID	Stripe_set	Number of Accesses	Dump
1	{sp_1, sp_2, sp_3}	105	1
2	{sp_7}	18	0
3	{sp_4, sp_5}	33	1
…	{……}	…	0
m	{sp_i, …, sp_w}	*k*	0

cache portion associated with one disk array is much larger than the array controller's local cache.

3.2 Implementations

From the beginning of serving I/O requests, MICRO launches a Popularity Evaluator (PE) process to record each file's popularity in terms of number of accesses within one epoch in a table called File Popularity Table (FPT). The FPT, which maintains the latest popularity information for each stripe set corresponding to each file, will be used later by Reconstruction Data Fetcher (RDF) processes to guide their reading sequence of reconstruction data from surviving disks. When one of the disks in a disk array fails, the MICRO strategy is activated in the array controller automatically. There are two phases in a MICRO-based reconstruction process: the dumping phase and the rebuilding phase. When a disk failure occurs, MICRO activates a Data Dumper (DD) process, which first dumps all of the data units from the failed disk in the local controller cache to a replacement disk. Next, all data units from the failed disk in the storage cache partition will be transferred into the local controller cache, from where they are eventually written onto the replacement disk. While transferring cached data to the replacement disk, the DD process records each dumped stripe set by changing its "Dump" field in the FPT to 1. A sample FPT is given in Table 1.

Note that a storage system only takes 20 μs to determine if a particular record is in cache [8]. As a result, the search time caused by finding data from the failed disk in cache is ignored in this study. In addition, since any transfers between the storage cache and the controller cache are achieved at electronic speeds that are a quantum leap faster than transfers involving disks [8], the data transmission time between the two levels of caches for reconstruction is omitted as well. Besides, the data processing bandwidth between cache and disk can be optimized up to 720 Mbytes/s in a modern storage system [8], which implies a very short data dumping (from the controller cache to the replacement disk) time. By transferring part of the failed disk data units directly from caches to the replacement disk, MICRO saves disk accesses needed to rebuild these data units. On the other hand, both DOR [11] and PRO [32] have to reconstruct every single data unit of the failed disk. Thus, compared with DOR and PRO, MICRO diminishes the number of disk accesses caused by disk recovery.

After all cached data units from the failed disk are restored in the replacement disk, MICRO enters into its rebuilding phase. Fig. 3 shows the workflows for core

```
PE (Popularity Evaluator)
1. Initialize a File Popularity Table (FPT)
2. Generate a temporary version of FPT (tFPT)
3. for each request arrived in current epoch do
4.     Extract File ID from the request
5.     Map the File ID to address Stripe_set
6.     Increase the popularity of Strip_set by 1
7.     Record the increase in tFPT
8. end for
9. Copy tFPT to FPT
10. Clear tFPT
11. Go to Step 3 for next epoch

DD (Data Dumper)
Copy FPT from storage cache to controller cache
Repeat
Find a file stored in controller cache.
Dump corresponding units of the file to the replacement disk.
Update the file's "Dump" field from 0 to 1 in the FPT table in controller cache.
Until (all cached units in CC have been dumped)
Repeat
Find a file stored in storage cache.
Move it to controller cache.
Dump corresponding units of the file to the replacement disk.
Update the file's "Dump" field from 0 to 1 in the FPT table in controller cache.
Until (all cached units in SC have been dumped)

RDF (Reconstruction Data Fetcher)
Repeat
Consult FPT to find the currently most popular unit on this disk that is needed for reconstruction.
Issue a low-priority request to read the indicated unit into a buffer.
Wait for the read request to complete.
Submit the unit data to a centralized buffer manager for XOR, or block the process if the buffer is full.
Until (all necessary units have been read)

RDD (Reconstructed Data Deliverer)
Repeat
Request the next full buffer from the buffer manager, blocking itself if none is available
Issue a low-priority write of the buffer to the replacement disk.
Wait for the write to complete.
Until (the failed disk has been reconstructed)
```

Fig. 3. The MICRO algorithm.

routines of MICRO. MICRO optimizes the reconstruction workflow by fetching reconstruction data of popular stripe sets from the failed disk prior to fetching reconstruction data of unpopular stripe sets. In fact, two types of processes run concurrently in a disk array with n disks when the disk array is in its rebuilding phase. The disk array controller creates $n-1$ processes, called RDF. Each RDF process associates with one surviving disk. In addition, a process named Reconstructed Data Deliverer (RDD) is launched in the disk array controller to write the reconstructed data onto the replacement disk. The functions of RDF and RDD are similar to those of the DOR algorithm [11] except for the following difference: An RDF process always selects the next most popular "under construction" unit rather than choosing the next sequential unit as the DOR algorithm. The workflow of PE, DD, RDF, and RDD is depicted as follows: CC represents the controller cache and SC denotes the

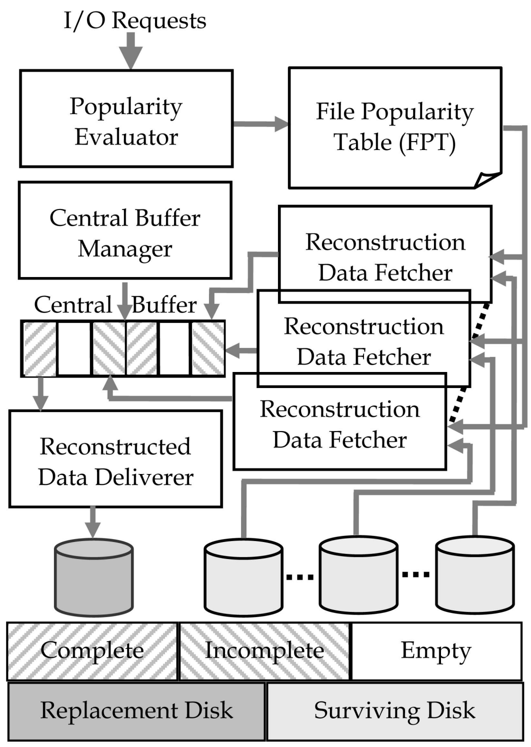

I/O Requests
Popularity Evaluator
File Popularity Table (FPT)
Central Buffer Manager
Reconstruction Data Fetcher
Central Buffer
Reconstruction Data Fetcher
Reconstruction Data Fetcher
Reconstructed Data Deliverer
Complete
Incomplete
Empty
Replacement Disk
Surviving Disk

Fig. 4. The architecture of MICRO.

storage cache. The interactions between the main components of MICRO are shown in Fig. 4.

3.3 Overhead Analysis

In this section, we analyze the overhead of MICRO in terms of space, time, and energy. While PE is always working in the storage cache to periodically update the FPT, DD, RDF, and RDD are running in the controller cache during tje rebuilding phase after one disk fails. First of all, the sizes of the execution codes of all components (PE, DD, RDF, and RDD) in our MICRO strategy are small due to their simple logic. In other words, they only consume a small portion of the cache compared with the total cache size in gigabyte scale. Second, in terms of memory usage, MICRO does not need the extra memory required by the PRO algorithm for the storage of the reconstruction thread descriptor [32]. More importantly, MICRO, unlike PRO, does not generate reconstruction jobs. Therefore, the space overhead in terms of $n-1$ job queues associated with the $n-1$ surviving disks can be saved. Next, since MICRO adopts a similar implementation methodology as DOR, its memory requirements are approximately those of DOR as well. The only extra memory required by MICRO is two tables: FPT and a temporary version of FPT (tFPT). However, as each record in the FPT only possesses 10 bytes in our implementation, for a file set with 5,000 files, the two tables use around 100 Kbyte cache memory in total, which is acceptable in modern storage systems.

Although the PE module is running all the time in the storage cache, its runtime overhead is trivial. Note that Steps 4-7 (see Fig. 3) each only take $O(1)$ time to accomplish and the FPT (see Table 1) is only around 50 Kbytes. Executing the loop (Steps 3-8 in PE) in one epoch (1,000 s) when the aggregate access rate is 100/s using a modern 3 GHz processor only requires 0.132 ms. In addition, copying 50 Kbyte data from tFPT to FPT (Step 9 in Fig. 3) only takes around 50 μs when the data transfer rate of the fast RAM in the storage cache is 1 Gbyte/s (e.g., the peak data transfer bandwidth for memory chip PC-266 DDR is 2.1 Gbytes/s [21]). Considering that, nowadays, the disk recovery process could last several hours or even longer due to the large amount of data [27], the risk of data loss because of a second disk failure during the recovery becomes high. Since data loss generally is prohibitively expensive and sometimes even unacceptable, we believe that paying PE's runtime overhead on the order of milliseconds to facilitate shrinking the reconstruction time, which in turn shortens the "window of vulnerability," is definitely worthwhile. The time for transferring cached data from the controller cache to the replacement disk and the time for updating each cached file's "Dump" field in the FPT are dominant in total time consumed by the dumping phase. Since there is only 64 Mbyte to 1 Gbyte data that could be cached for a failed disk in a four-disk array with a total of 1/4 to 4 Gbyte cache and data processing bandwidth between cache and disk can be optimized up to 720 Mbytes/s, the time for dumping cached data from controller cache to the replacement disk is around 1.39 seconds. Assume that, among the total m files in the storage system, k files were cached before the disk failure occurs. The time spent on updating each cached file's "Dump" status in FPT is $O(k)$, where $k << m$ because of the very limited size of the cache. Thus, the dumping time, which is not necessary for DOR and PRO, is trivial in the whole recovery time span. In the rebuilding phase, comparing with PRO, MICRO obviously saves time because it does not need to spend time in thread scheduling and job queue operations (*enqueue* and *dequeue*). Compared with DOR, MICRO requires extra time overhead $O(m-k)$ because it consults the FPT for every noncached file's reconstruction.

MICRO pays extra energy consumption caused by dumping cached data from the controller cache to the replacement disk and maintaining the FPT, which are not necessary for both DOR and PRO. However, in what follows, we demonstrate that the additional energy consumption of MICRO is minimal. First of all, considering that the typical energy consumption rate of the Seagate Cheetah ST3146854LW disk drive is 17.5 W [28] and the time for dumping cached data from the controller cache to the replacement disk is around 1.39 seconds, the energy consumption of the dumping process is around 24.3 J. Next, the storage cache normally consists of static RAM chips due to their high speed. Typical access time for modern static RAM is on the order of 20-50 ns [10]. Besides, these devices usually consume several hundred milliwatts when they are in active mode [10]. In what follows, we will use an illustrative example to show that the energy used for maintaining the FPT is also negligible. For example, the read cycle time and the write cycle time of the static RAM IDT71V30 are only 25 ns, respectively [10]. In addition, when it is in active mode, the energy consumption rate is only 375 mW. Let us assume a heavy workload condition, where 200 requests arrive every second (i.e., the aggregate access rate is 200) and each request reads a file from the beginning to the end. Therefore, every second, the PE module needs to access the tFPT (see Fig. 3) 400 times (200 times for reading the current number of accesses for each visited file and 200 times for updating their access numbers). Thus, the energy consumed by memory accesses for updating the tFPT in 1 second is $400 * 25 * 10^{-9} * 375 * 10^{-3} = 375 * 10^{-8}$ J. Consequently, the energy consumed by maintaining the tFPT in one month is around 10.04 $(375 * 10^{-8} * 3{,}600 * 24 * 31 = 10.044)$ J. In addition, copying 50 Kbyte (i.e., the size of the tFPT) data from tFPT to FPT (Step 9 of the PE module in Fig. 3) once in every epoch only takes around 50 μs. As a result, with an epoch that is equal to 1,000 seconds, energy consumption caused by copying 50 Kbyte data from tFPT to FPT is around 0.05 $(50 * 10^{-6} * 375 * 10^{-3} * (3{,}600/1{,}000) * 24 * 31 = 0.05022)$ J in one month. Thus, the total monthly energy consumption caused by maintaining the FPT is around 10.09 J. On the other hand, as we explained in Section 1, the annual disk replacement rates in the field are usually in the range of 2 percent to 4 percent [27]. Let us assume a moderate annual disk replacement rate of 3 percent for a mobile disk array with a total of 384 disks (e.g., EMC Symmetrix 8000 can accommodate up to 384 disks in RAID-5 organization [8]). Hence, around 12 $(384 * 3 \text{ percent} = 11.52)$ disks among the 384 disks need to be replaced within one year, which implies one occurrence of disk recovery every month. Considering the energy saved by the MICRO algorithm when compared with DOR and PRO during one disk recovery process is on the order of 10^6 J (see Figs. 6, 7, 8, 9, and 10), 34.39 $(10.09 + 24.3 = 34.39)$ J extra energy overhead of MICRO in one month is trivial and can be safely ignored.

4 Performance Evaluation

This section presents the results of a comprehensive experimental study comparing the proposed MICRO strategy with two existing algorithms, PRO and DOR, using extensive simulations. Section 4.1 introduces the methodology utilized in this study, including storage system configurations. A detailed analysis of the characteristics of both synthetic workloads and a real-world trace that have been employed in our simulations is provided in Section 4.2. Experimental results from synthetic simulations are discussed in Sections 4.3 and 4.4, which are then validated by a trace-driven experiment presented in Section 4.5.

4.1 Methodology

While the DOR algorithm is well recognized as the most effective existing reconstruction algorithm due to the fact that it has been implemented in many real applications [11], the PRO strategy represents the latest advances in storage system reconstruction optimization [32]. In order to fully examine our MICRO algorithm, we compare it with the representative algorithm DOR as well as the state-of-the-art strategy PRO in this simulation study. A brief introduction of the two algorithms is presented below.

1. DOR (Disk-Oriented Reconstruction. In a disk array with n disks, DOR activates $n-1$ processes associated with $n-1$ surviving disks to sequentially fetch reconstruction data and then put it in a centralized buffer. In addition, one process dedicated to the replacement disk repeatedly transfers reconstructed data from the centralized buffer to the replacement disk. The goal of DOR is to absorb all of the array's bandwidth that is not absorbed by users. DOR is a classic heuristic that is widely used in real applications.
2. PRO (Popularity-based Multithreaded Reconstruction). PRO reconstructs high-popularity data units of a failed disk, which are the most frequently accessed units in terms of the workload characteristics, prior to reconstructing other units. Therefore, the PRO algorithm has the potential to recover many units ahead of users' accesses with high probability to avoid performance degradation caused by recovery. The major difference between PRO and DOR is that PRO optimizes the workflow of the reconstruction procedure by utilizing workload localities, which were not considered in DOR.

Note that neither DOR nor PRO employs multilevel cache to facilitate their reconstruction processes. Moreover, although both MICRO and PRO leverage access localities, MICRO uses PE, which maintains a recent history of file popularities from the beginning of the execution of an application, to predict access pattern during the reconstruction process. On the other hand, PRO launches its Access Monitor (AM) only after a disk failure occurs to establish multiple hot zones based on the number of accesses in the replacement disk, which burdens system load and delays reconstruction progress.

We analyze reconstruction performance in terms of mean response time during reconstruction, reconstruction time, and energy consumption during disk recovery. The three performance metrics are defined as follows:

- *Mean Response Time*: Average user response time in milliseconds during the recovery process.
- *Reconstruction Time*: The time in seconds needed for a disk array to recover from failure mode to normal mode.
- *Energy Consumption*: The amount of energy in Joules consumed by the reconstruction process.

We have developed an execution-driven simulator, called the Cache-Aware Reconstruction SIMulator (CARSIM) 1.0, which models a storage system with n disk arrays (see Fig. 1). Each disk array is made up of m small form factor mobile hard disks organized in a RAID-5 structure. Notice that existing small form factor hard disk drives like Seagate ST1.3 [2] and Imation Micro [31] are intended only for consumer electronics. Generally, they have very low capacities and performance. For example, the largest micro hard disk, Seagate ST1.3, is only 12 Gbytes, with an effective transfer rate of 10.2 Mbytes/s. Therefore, they are not suitable for server-class mobile applications like a mobile data center. However, based on hard disk technology trends, one can reasonably conjecture that server-class small form factor (less than 2 in) mobile hard disk drives which can provide similar capacity and performance as

TABLE 2
System Characteristics

Description	Value	Description	Value
CPU	Intel Pentium4 2.8 GHz	Memory	2 GB DDR SDRAM
OS	Windows XP SP2	Simulator	CARSIM 1.0
Disk model	Seagate Cheetah ST3146854LW	Standard interface	SCSI Ultra320
Capacity	147 GBytes	Rotational speed	15,000 rpm
Average seek time	3.5 ms	Transfer rate	64 Mbytes/second
Active Energy	61 mJoule/8KB	Idle power	6.26 Watts

current enterprise-level stationary disks like Seagate Cheetah ST3146854LW (3.5 in) will be available on the market in the near future. Thus, our simulator CARSIM 1.0 employs the parameters of the Seagate Cheetah ST3146854LW to emulate a future mobile storage system. The main characteristics of the simulation platform and the disk are shown in Table 2. Energy-related parameters in Table 2 are derived from [25]. In addition, two levels (Level 1 and Level 2) of caches were simulated within CARSIM 1.0 as well (see Fig. 1).

While the storage cache is controlled by a storage server processor, a controller cache is managed by a RAID controller. In our performance study, cache size refers to the combinational size of the storage cache and the controller cache associated with a disk array. In other words, the cache size of a disk array is the sum of the size of the disk array's RAID controller cache and the size of one storage cache partition (see Fig. 2). For simplicity, we set n as 1, which implies that only one disk array exists in the simulated storage system. The number of disks in the disk array varies from 3 to 7 ($3 \leq m \leq 7$) as modern storage systems such as EMC Symmetrix prefer to utilize small-scale (≤ 8 disks) disk arrays as building blocks [37]. We choose cache size from 1/4 to 4 Gbytes with 128 Mbyte fixed size for the controller cache, which approximates real-world cache configurations in current storage systems [8].

Both synthetic workloads and a real-world trace were used to drive the CARSIM simulator. The advantage of using synthetic simulations is that the impacts of major workload features on the reconstruction algorithms' performance can be examined. Trace-driven simulations, however, can validate the experimental results obtained from synthetic workloads.

4.2 Synthetic Workload Analysis

Since we are considering a Web server application running in a mobile data center, we believe that Web I/O traces like ClarkNet-HTTP [3] are appropriate to represent a typical workload of a mobile disk array storage system. Therefore, the main characteristics of the synthetic workload used in Sections 4.3 and 4.4 are based on normal Web workloads. Because workload characteristics directly influence data reconstruction, we identified four key characteristics: skew degree, aggregate access rate, base file size, and file size distribution.

1. *Skew degree.* File popularity weight relates to the frequency with which file requests arrive at the disk array. Since the frequency of file access usually exhibits a Zipf-like distribution, we assume that the distribution of file access requests is a Zipf-like distribution with a skew parameter $\theta = \log \frac{X}{100} / \log \frac{Y}{100}$, where X percent of all accesses were directed to Y percent of files. The value of $X:Y$ is called skew degree in this paper. In our simulations, we tested four values of θ with skew degree $(X:Y)$ changing from 50:50 to 90:10. Since 70:30 is a realistic assumption of the level of data skew [26] and it has been widely used in the literature [4], [15], [19], [26], we select it as the default value of skew degree in our synthetic simulations.
2. *Aggregate access rate.* This is defined as the average number of I/O requests arriving in a disk array per second. Each file access represents a sequential read of the entire file. Hence, the service time of a file access request is proportional to the file's size. We assume that each file has a fixed request arrival rate, λ_i, and the arrival interval times are exponentially distributed. The aggregate access rate of the entire system is defined as $\sum_{i=1}^{5,000} \lambda_i$. The value of the aggregate access rate represents the intensity of the total access requests submitted to the disk array where 5,000 files have been assigned across. Note that our PE has no knowledge of each file's λ_i as a prior. It dynamically records each file's popularity in terms of number of accesses within one epoch in the FPT. We set an epoch to 1,000 seconds in our simulations. Note that, when the aggregate access rate is 88, the average request interarrival time is 11.36 ms, which is close to the interarrival pattern of ClarkNet-HTTP [3].
3. *Base file size.* The base file size is the size of the smallest file among all of the 5,000 files saved across a disk array. The sizes of all of the other files can be computed based on the file size distribution curve shown in Fig. 5 since the file size in a Web application usually varies from several kilobytes to several megabytes. Besides, considering that the stripe unit was set to 64 Kbytes, our selection of base file size from 250 Kbytes to 4 Mbytes is reasonable.
4. *File size distribution.* The distribution of access rates across the files and the distribution of file sizes were inversely correlated with the same skew parameter θ, as shown in Fig. 5. The file size distribution is reasonable because the phenomena that popular files are generally small ones can be frequently observed.

Note that the transfer rate (Table 2) of the disk combined with the file size decides a file access request's service time, which in turn affects the mean response time. Table 3 summarizes the configuration parameters of a simulated disk array used in our experiments and the characteristics of the synthetic workload. All synthetic workloads used

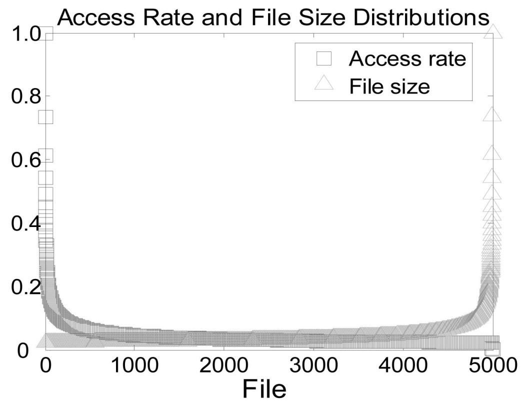


Fig. 5. Access rate and file size distributions.

TABLE 3
Experimental Parameters

Parameter	Value (Fixed) – (Varied)
File number	(5000)
Disk number	(4) – (3, 4, 5, 6, 7)
Cache Size (GB)	(1) – (1/4,1/2,1,2,3,4)
Skew Degree	(70:30)–(50:50, 60:40, 70:30, 80:20, 90:10)
Aggregate Access Rate (1/second)	(18) – (18, 22, 30, 44, 88)
Base File Size (MB)	(1) – (1/4,1/2,1,3/2,4)
Stripe unit (KB)	64

from Section 4.2 to Section 4.3 were created by our trance generator. Although cache size, skew degree, number of disks, aggregate access rate, and size of files are synthetically generated, we examined their impacts on system performance by controlling the parameters. All experiments were conducted on a simulated RAID-5 disk array.

4.3 The Impact of Storage System Configurations

The goal of this experiment is to examine the impact of storage system configurations on storage system performance and energy consumption. More specifically, we investigate the impact of the size of caches associated with a disk array, as well as the impact of the number of disks in a disk array.

Fig. 6 shows that MICRO outperforms the two existing algorithms in all four performance metrics in all cases when the cache size varies from 1/4 to 4 Gbytes. It is obvious that MICRO gains more improvements with more caches being used. On the other hand, all metrics of DOR and PRO are kept constant because they do not exploit the two-level caches. Compared with DOR and PRO, MICRO on average decreases reconstruction time by 21.22 percent and 10.46 percent, respectively. Since, in a mobile disk-array-based storage system, the possibility that a second disk drive fails shortly after the first disk failure is higher than the same size stationary storage system, it is very important to shorten the reconstruction time to avoid a second disk failure during the recovery of the first disk failure for the sake of reliability and availability of the mobile storage system. In terms of mean response time during reconstruction, MICRO on average improves by 55.23 percent and 36.61 percent. More importantly, MICRO saves energy by at least 32.69 percent and 15.86 percent compared with the two current approaches. We attribute the performance improvements and energy savings to the shortened reconstruction time and the reduced number of disk accesses during the recovery process. The larger the caches are, the more popular data can be cached and, thus, their corresponding requests can be served in the caches.

We then evaluated the impact of the number of disks in an array. Fig. 7 demonstrates that the reconstruction time and mean response time decrease when the number of disks increases. This is because more surviving disks can share the workload caused by normal I/O requests and reconstruction I/O requests. Energy consumption, however, goes down slightly because of more working disks in the array. The performance improvements gained by MICRO become less because, with a greater number of disks, fewer files can be cached in the two-level caches.

4.4 The Impact of Workload Characteristics

To verify the performance impact of workload characteristics, including aggregate access rate, skew degree, and base file size, we evaluated the performance as a function of these parameters in this section.

When the request load increases, results from Fig. 8 show that MICRO obtains more improvement in terms of rebuild mean response time. The reason is that the major part of the increased number of requests during recovery targets popular files, which most likely have been restored to the replacement during the data dumping phase before they are queried. On the contrary, DOR and PRO do not have the

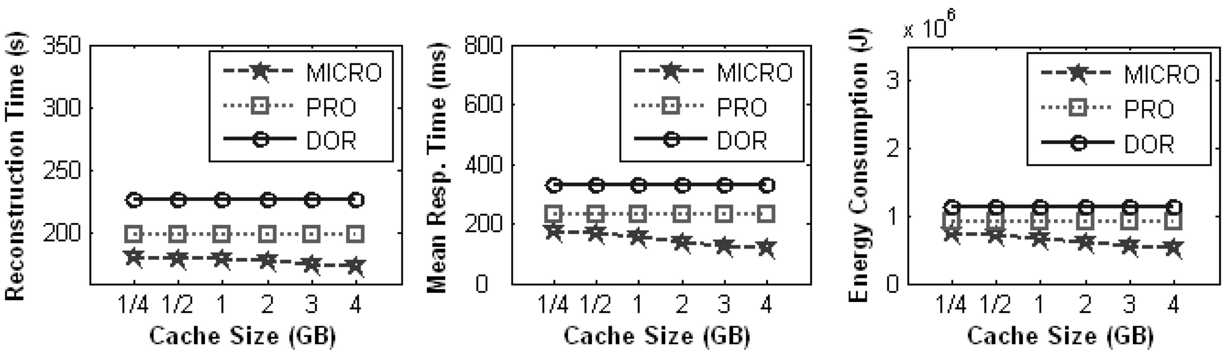


Fig. 6. Impact of cache size.

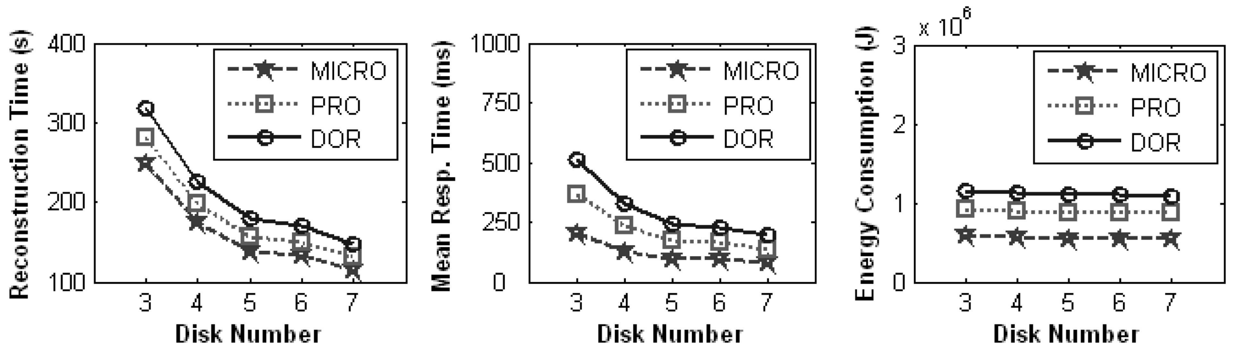


Fig. 7. Impact of number of disks.

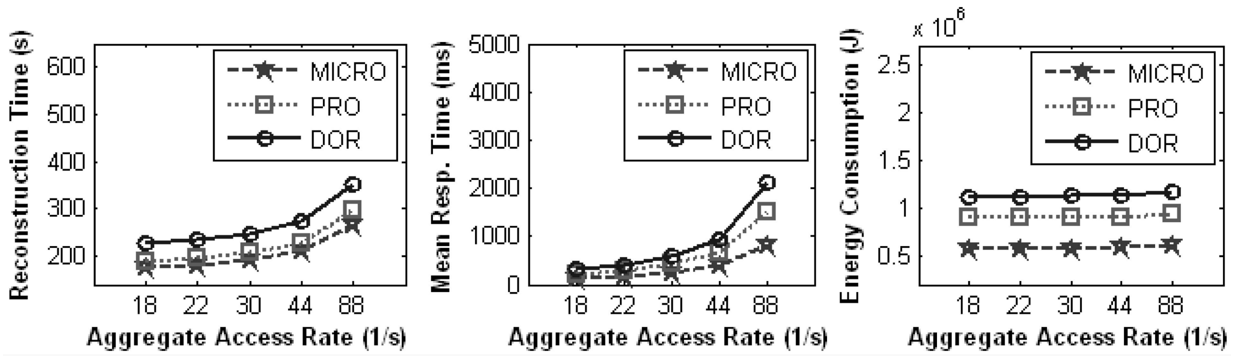


Fig. 8. Impact of aggregate access rate.

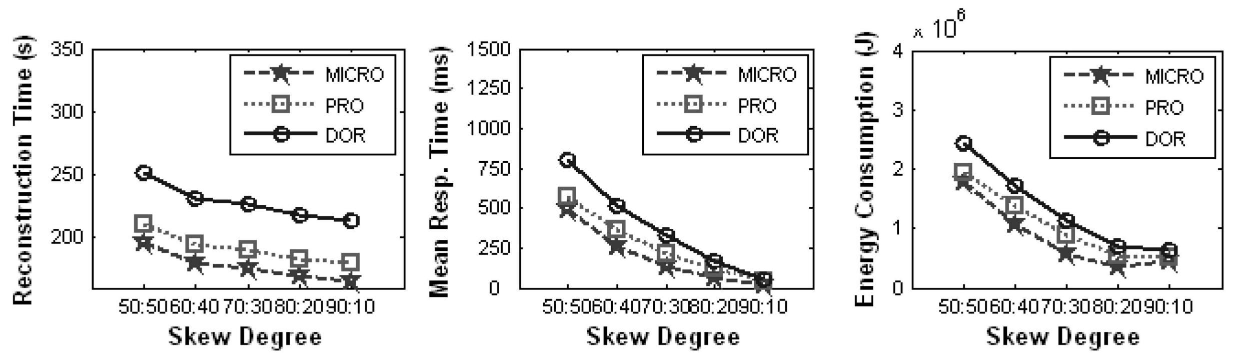


Fig. 9. Impact of skew degree.

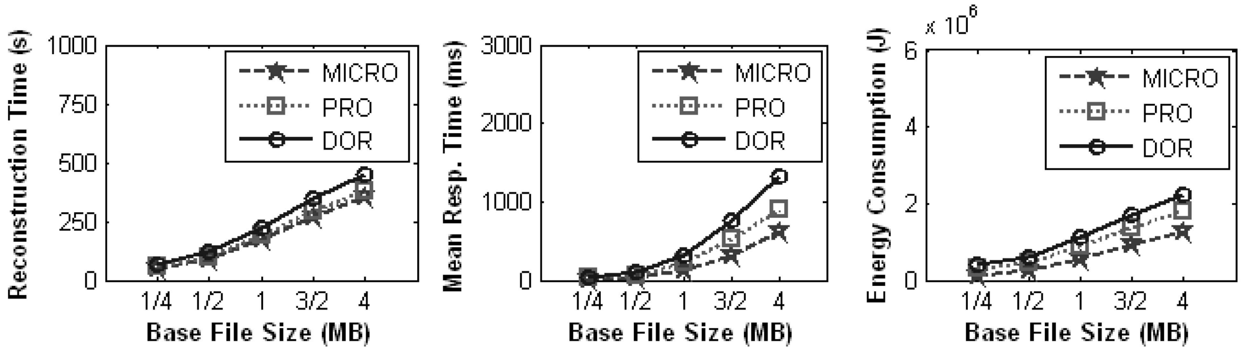


Fig. 10. Impact of base file size.

data dumping phase and thus need to rebuild every single file from the surviving disks.

When the skew degree is set to 50:50 (Fig. 9), MICRO only marginally outperforms PRO in terms of reconstruction time and energy consumption. This is because the skew parameter is equal to 1, which means the access requests were evenly distributed across all files without any request skew. As a result, there is no popular file in the entire file set. In this case, the data restored from caches to the replacement disk is not popular. As a result, data dumping cannot noticeably help reduce reconstruction time. On the other hand, however, when the skew degree was enlarged

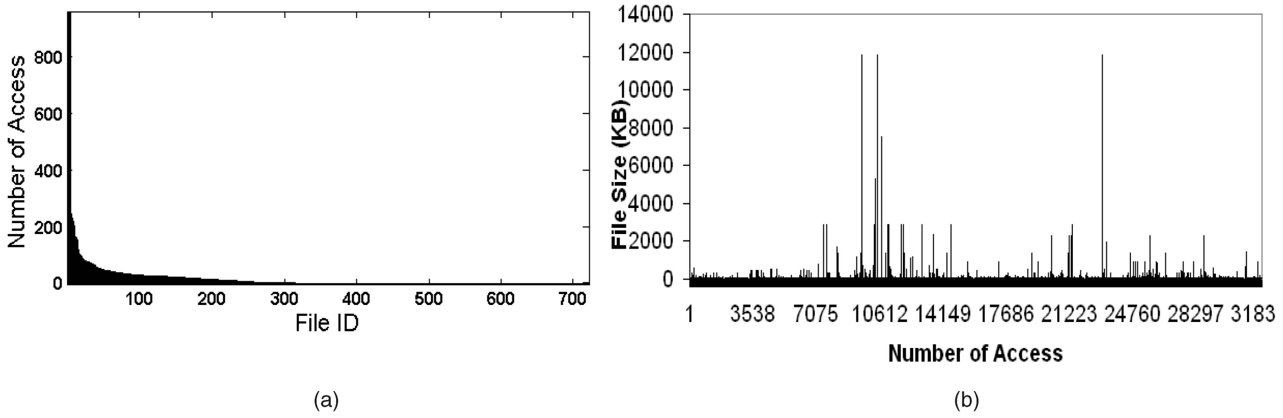


Fig. 11. (a) Access rate distribution. (b) File size distribution of the ClarkNet-HTTP trace.

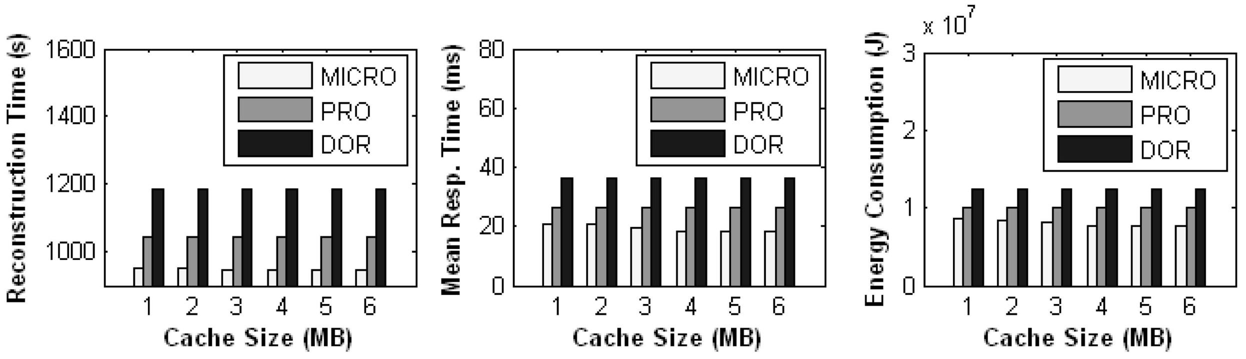


Fig. 12. Impact of cache size.

to 70:30, MICRO obviously improved the mean response time and energy consumption. We observed from Fig. 8 that MICRO achieves the best mean response time improvement when the skew degree is 70:30.

For completeness, we also conducted an experiment on the reconstruction metrics by increasing the base file size. Intuitively, reconstruction time and response time increase while the file size becomes larger. This is natural because serving I/O and recovering disk takes a longer time, regardless of what algorithm has been used. Fig. 10 illustrates that MICRO is still superior to the two existing approaches when file size increases. In particular, MICRO results in a slower degradation in both performance and energy than the two baseline algorithms.

In summary, by exploiting multilevel caching and access locality, the MICRO algorithm consistently outperforms the DOR and PRO algorithms in both performance and energy consumption during recovery. The experimental results above strongly imply that MICRO has the potential to be applied in mobile storage systems where high reliability, high performance, and energy efficiency are musts.

4.5 Trace-Driven Simulations

To validate the experimental results from synthetic workloads, we evaluate the three reconstruction algorithms by trace-driven simulations in this section.

The real-world trace that we used is ClarkNet-HTTP log [3], which was collected by ClarkNet, an Internet service provider, for a week from 09/04/95 to 01/10/95 with a total of 3,328,587 requests. The frequency of file access in the trace follows a Zipf-like distribution, which has been widely used in storage system research [5], [7], [36]. Since the reconstruction times in our experiments are much shorter compared with the time span of the ClarkNet-HTTP trace, we only used the first 50,000 requests in the trace to conduct the simulations. In addition, we filtered out all write accesses because the trace is read-dominant. The number of different files accessed by the 50,000 requests is 2,745, with an average file size of 49.8 Kbytes. In addition, the most popular 10 percent of files (275 files) with a total size of 4.65 Mbytes attracted 88.5 percent requests and 90 percent requests targeted on 15 percent files (412 files) with a total size of 17.47 Mbytes. Fig. 11 shows the characteristic of the trace. Although the distribution of user requests generally follows a Zipf-like distribution (Fig. 11a), there is only a weak correlation between file access frequency and file size (Fig. 11b). In other words, some popular files are not small in size. In addition, the size of a stripe unit is set to 8 Kbytes.

The results from Figs. 12 and 13 are consistent with those of Figs. 6 and 7. We noticed that the performance improvement in terms of reconstruction time and mean response time became less significant in the trace-driven simulations. The reason is that there is only a weak correlation between file access frequency and file size shown in the trace. Some popular files are also large in size, which degraded the efficiency of caching. Detailed evaluation between MICRO and PRO is presented in Table 4. In summary, the trace-driven simulation results validated the synthetic experimental results. More importantly, they indicate that our MICRO algorithm has the potential to be applied in real-world applications.

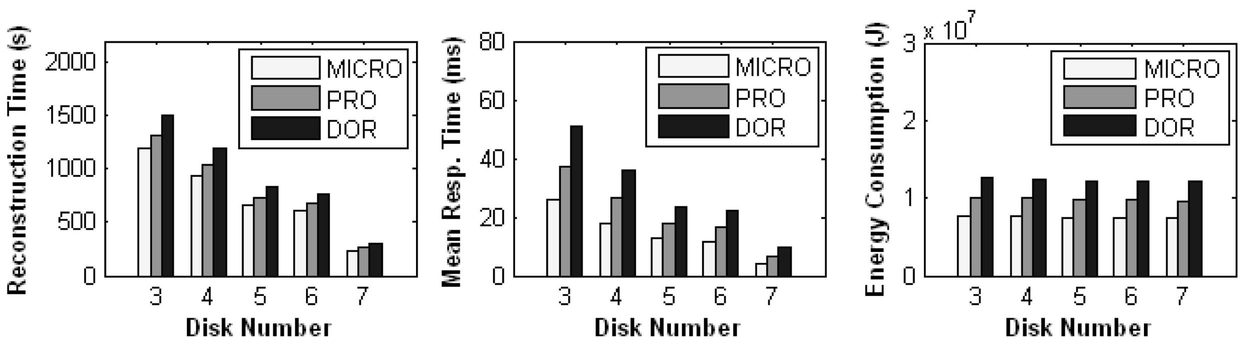


Fig. 13. Impact of number of disks.

TABLE 4
A Comparison between PRO and MICRO

Cache (MB)	Reconstruction Time (second)			Mean Response Time (ms)			Energy Consumption (Joule)		
	PRO	MICRO	Improved	PRO	MICRO	Improved	PRO	MICRO	Improved
1	1044.75	950.22	9.05%	26.31	20.81	20.90%	9929972.1	8639338.6	13.00%
2	1044.75	948.98	9.17%	26.31	20.47	22.97%	9929972.1	8487951.9	14.52%
3	1044.75	948.46	9.22%	26.31	19.62	25.43%	9929972.1	8105804.8	18.37%
4	1044.75	946.75	9.38%	26.31	18.45	29.87%	9929972.1	7770746.1	21.74%
5	1044.75	944.32	9.61%	26.31	18.12	31.13%	9929972.1	7651167.0	22.95%
6	1044.75	944.32	9.61%	26.31	18.12	31.13%	9929972.1	7651167.0	22.95%

Disk Number	Reconstruction Time (second)			Mean Response Time (ms)			Energy Consumption (Joule)		
	PRO	MICRO	Improved	PRO	MICRO	Improved	PRO	MICRO	Improved
3	1318.09	1195.18	9.32%	37.16	25.66	30.95%	10026983.0	7748394.3	22.72%
4	1044.75	944.32	9.61%	26.31	18.12	31.13%	9929972.1	7651167.0	22.95%
5	739.48	670.12	9.26%	17.91	12.80	28.53%	9821273.1	7542854.5	23.20%
6	674.88	603.60	10.56%	16.44	11.31	31.20%	9798702.6	7519488.8	23.26%
7	264.06	239.30	9.38%	6.57	4.26	35.16%	9652897.5	7374607.8	23.60%

5 Summary and Future Work

In this paper, we have addressed the issue of data reconstruction in a RAID-structured mobile disk storage system where high reliability, high performance, and energy saving are demanded. To achieve these goals simultaneously, a new reconstruction strategy, called MICRO, is developed for RAID-structured mobile storage systems to noticeably save energy while providing shorter reconstruction times and user response times. MICRO collaboratively utilizes the storage cache and the disk array controller cache to diminish the number of physical disk accesses caused by reconstruction. Compared with two representative reconstruction algorithms, DOR and PRO, MICRO achieves noticeable improvement in reliability, performance, and energy consumption with reasonable spatial and time complexity. Comprehensive experimental results show that MICRO consistently improves the performance of mobile disk storage systems in terms of response times and saves energy. Compared with DOR and PRO, MICRO on average decreases reconstruction time by 20.22 percent and 9.34 percent, respectively. In terms of mean response time during reconstruction, MICRO on average improves by 46.59 percent and 26.91 percent. More importantly, MICRO saves energy by at least 30.4 percent and 13 percent compared with the two current approaches.

In summary, the MICRO strategy realizes energy saving not at the cost of performance degradation and system reliability, but, rather, it delivers much shorter mean response times during recovery compared with existing algorithms. The MICRO strategy in its current form only works for read-dominated workloads such as Web, proxy, ftp, and e-mail server applications. Future studies in this research can be performed in the following directions: First, we will extend our scheme to accommodate write-dominated workloads, where each file access demands parity information updating. The major new challenge for MICRO to incorporate write operations is how to maintain data consistency between multilevel caches and disks. Next, we intend to complete our caching-based data reconstruction algorithm by taking more RAID architectures into account. We only consider the RAID-5 structure in this work.

Acknowledgments

This work was supported by US National Science Foundation Computing Processes and Artifacts (CISE-CCF) under Grant 0742187. The authors would like to thank the anonymous reviewers whose comments noticeably improved the quality of this paper.

References

[1] G. Alvarez, W. Burkhard, L. Stockmeyer, and F. Cristian, "Declustered Disk Array Architectures with Optimal and Near-Optimal Parallelism," *Proc. 25th Int'l Symp. Computer Architecture,* pp. 109-120, 1998.

[2] D. Anderson and W. Whittington, "Hard Drives: Today and Tomorrow," *Proc. Fifth USENIX Conf. File and Storage Technologies,* Feb. 2007.

[3] M. Arlitt and C. Williamson, "Web Server Workload Characterization: The Search for Invariants," *Proc. ACM SIGMETRICS Int'l Conf. Measurement and Modeling of Computer Systems,* pp. 126-137, 1996.

[4] A. Bhalekar and J. Baras, "Cumulative Caching for Reduced User-Perceived Latency for WWW Transfers on Networks with Satellite Links," *Lecture Notes in Computer Science,* vol. 3126/2004, pp. 179-186, 2004.

[5] E. Carrera, E. Pinheiro, and R. Bianchini, "Conserving Disk Energy in Network Servers," *Proc. 17th Ann. Int'l Conf. Supercomputing,* pp. 86-97, 2003.

[6] Z. Chen, Y. Zhang, Y. Zhou, H. Scott, and B. Schiefer, "Empirical Evaluation of Multi-Level Buffer Cache Collaboration for Storage Systems," *Proc. ACM SIGMETRICS Int'l Conf. Measurement and Modeling of Computer Systems,* pp. 145-156, 2005.

[7] B. Diniz, D. Guedes, W. Meira, and R. Bianchini, "Limiting the Power Consumption of Main Memory," *Proc. 34th Ann. Int'l Symp. Computer Architecture,* pp. 290-301, 2007.

[8] EMC Corp., Symmetrix Enterprise Storage Systems Product Description Guide, 1999.

[9] S. Gurumurthi, J. Zhang, A. Sivasubramaniam, M. Kandemir, H. Fanke, N. Vijaykrishnan, and M. Irwin, "Interplay of Energy and Performance for Disk Arrays Running Transaction Processing Workloads," *Proc. IEEE Int'l Symp. Performance Analysis of Systems and Software,* pp. 123-132, Mar. 2003.

[10] High-Speed 1K $\times$ 8 Dual-Port Static RAM, http://www.semiconductorstore.com/pdf/newsite/idt/71V30S55TFG_DS.pdf, 2008.

[11] M. Holland, "On-Line Data Reconstruction in Redundant Disk Arrays," PhD dissertation CMU-CS-94-164, Carnegie Mellon Univ., Apr. 1994.

[12] M. Holland and G. Gibson, "Parity Declustering for Continuous Operation in Redundant Disk Arrays," *Proc. Fifth Int'l Conf. Architectural Support for Programming Languages and Operating Systems,* pp. 23-35, 1992.

[13] M. Holland, G. Gibson, and D. Siewiorek, "Fast, On-Line Failure Recovery in Redundant Disk Arrays," *Proc. 23rd Ann. Int'l Symp. Fault-Tolerant Computing,* pp. 422-443, 1993.

[14] M. Holland, G.A. Gibson, and D. Siewiorek, "Architectures and Algorithms for On-Line Failure Recovery in Redundant Disk Arrays," *J. Distributed and Parallel Databases,* vol. 2, no. 3, pp. 295-335, July 1994.

[15] B.S. Jeong and E. Omiecinski, "Inverted File Partitioning Schemes in Multiple Disk Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 6, no. 2, pp. 142-153, Feb. 1995.

[16] M. Kaaniche, L. Romano, Z. Kalbarczyk, R. Iyer, and R. Karcich, "A Hierarchical Approach for Dependability Analysis of a Commercial Cache-Based RAID Storage Architecture," *Proc. 28th Ann. Int'l Symp. Fault-Tolerant Computing,* pp. 6-15, 1998.

[17] T. Kwan, R. Mcgrath, and D. Reed, "Ncsas World Wide Web Server Design and Performance," *Computer,* vol. 28, no. 11, pp. 67-74, Nov. 1995.

[18] J.Y.B. Lee and J.C.S. Lui, "Automatic Recovery from Disk Failure in Continuous-Media Servers," *IEEE Trans. Parallel and Distributed Systems,* vol. 13, no. 5, pp. 499-515, May 2002.

[19] L.W. Lee, P. Scheuermann, and R. Vingralek, "File Assignment in Parallel I/O Systems with Minimal Variance of Service Time," *IEEE Trans. Computers,* vol. 49, no. 2, Feb. 2000.

[20] B.A. Mah, S. Seshan, K. Keeton, R.H. Katz, and D. Ferrari, "Providing Network Video Service to Mobile Clients," *Proc. Fourth Workshop Workstation Operating Systems,* pp. 48-54, 1993.

[21] "Memory Speeds: Have You Ever Wondered How They Were Determined?" The DEW Assoc., http://www.dewassoc.com/performance/memory/memory_speeds.htm, 2008.

[22] J. Menon and J. Cortney, "The Architecture of a Fault-Tolerant Cached RAID Controller," *Proc. 20th Ann. Int'l Symp. Computer Architecture,* pp. 76-86, 1993.

[23] Mobile Emergency Datacenter, North Am. Access Technologies, http://www.naat.com/Disaster%20Recovery/mobile_datacenter.htm, 2006.

[24] R. Muntz and J. Lui, "Performance Analysis of Disk Arrays under Failure," *Proc. 16th Int'l Conf. Very Large Data Bases,* pp. 162-173, 1990.

[25] E. Pinheiro and R. Bianchini, "Energy Conservation Techniques for Disk Array-Based Servers," *Proc. 18th Ann. Int'l Conf. Supercomputing,* pp. 68-78, June 2004.

[26] D. Rinfret, P. O'Neil, and E. O'Neil, "Bit-Sliced Index Arithmetic," *Proc. ACM SIGMOD '01,* pp. 47-57, 2001.

[27] 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. Fifth USENIX Conf. File and Storage Technologies,* Feb. 2007.

[28] Seagate Cheetah 15k.4 Mainstream Disc Drive Storage, http://www.seagate.com/docs/pdf/marketing/ds_1566_004_cheetah_15k_4.pdf, 2008.

[29] P.G. Sikalinda, L. Walters, and P.S. Kritzinger, "A Storage System Workload Analyzer," Technical Report CS06-02-00, Univ. of Cape Town, 2006.

[30] M. Sivathanu, V. Prabhakaran, F. Popovici, T.E. Denehy, A.C. Arpaci-Dusseau, and R.H. Arpaci-Dusseau, "Improving Storage System Availability with D-GRAID," *Proc. Third USENIX Conf. File and Storage Technologies,* Mar. 2003.

[31] The Smallest Hard Disk Drive, http://news.softpedia.com/news/The-Smallest-Hard-Disk-Drive-4533.shtml, 2008.

[32] L. Tian, D. Feng, H. Jiang, K. Zhou, L. Zeng, J. Chen, Z. Wang, and Z. Song, "PRO: A Popularity-Based Multi-Threaded Reconstruction Optimization for RAID-Structured Storage Systems," *Proc. Fifth USENIX Conf. File and Storage Technologies,* Feb. 2007.

[33] T. Wong and J. Wilkes, "My Cache or Yours? Making Storage More Exclusive," *Proc. USENIX Ann. Technical Conf.,* pp. 161-175, June 2002.

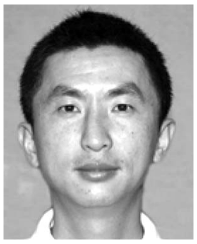
[34] Q. Xin, E. Miller, and T.J. Schwarz, "Evaluation of Distributed Recovery in Large-Scale Storage Systems," *Proc. 13th Int'l Symp. High Performance Distributed Computing,* pp. 172-181, 2004.

[35] Q. Xin, E. Miller, T.J. Schwarz, and D. Long, "Reliability Mechanism for Very Large Storage Systems," *Proc. 20th IEEE/11th NASA Goddard Conf. Mass Storage Systems and Technologies,* 2000.

[36] J. Yang, W. Wang, and R. Muntz, "Collaborative Web Caching Based on Proxy Affinities," *Proc. ACM SIGMETRICS Int'l Conf. Measurement and Modeling of Computer Systems,* pp. 78-89, 2000.

[37] X. Yao and J. Wang, "RIMAC: A Novel Redundancy-Based Hierarchical Cache Architecture for Energy Efficient, High Performance Storage Systems," *Proc. EuroSys Conf.,* pp. 249-262, 2006.

[38] Q. Zhu and Y. Zhou, "Power-Aware Storage Cache Management," *IEEE Trans. Computers,* vol. 54, no. 5, pp. 587-602, May 2005.



Tao Xie received the BSc and MSc degrees from Hefei University of Technology, Hefei, China, in 1991 and 2000, respectively, and the PhD degree in computer science from the New Mexico Institute of Mining and Technology in 2006. He is currently an assistant professor in the Department of Computer Science at San Diego State University, California. His research interests include storage systems, high-performance computing, cluster and grid computing, parallel and distributed systems, and real-time/embedded systems. He is a member of the IEEE and the IEEE Computer Society.

Hui Wang received the BS degree in information and image sciences from Chiba University, Chiba, Japan, in 2002 and the MS degree in computer science from San Diego State University in 2007. He is currently a network engineering manager at Ortiva Wireless, which offers the industry's only commercial solution for rich media content delivery optimization over wireless networks.