ROS: A Rack-based Optical Storage System with Inline Accessibility for Long-Term Data Preservation

Wenrui Yan Jie Yao* Qiang Cao†
Changsheng Xie

Wuhan National Laboratory for Optoelectronics,
Key Laboratory of Information Storage System,
Ministry of Education of China
School of Computer Science and Technology,
Huazhong University of Science and Technology
{wenrui_yan, jackyao, caoqiang,
cs_xie}@hust.edu.cn

Hong Jiang
Dept. of CSE University of Texas at Arlington
hong.jiang@uta.edu

Abstract

The combination of the explosive growth in digital data and the need to preserve much of this data in the long term has made it an imperative to find a more cost-effective way than HDD arrays and more easily accessible way than tape libraries to store massive amounts of data. While modern optical discs are capable of guaranteeing more than 50-year data preservation without migration, individual optical disks' lack of the performance and capacity relative to HDDs or tapes has significantly limited their use in datacenters. This paper presents a Rack-scale Optical disc library System, or ROS in short, that provides a PB-level total capacity and inline accessibility on thousands of optical discs built within a 42U Rack. A rotatable roller and robotic arm separating and fetching the discs are designed to improve disc placement density and simplify the mechanical structure. A hierarchical storage system based on SSD, hard disks and optical discs are presented to hide the delay of mechanical operation. On the other hand, an optical library file system is proposed to schedule mechanical operation and organize data on the tiered storage with a POSIX user interface to provide an illusion of inline data accessibility. We evaluate ROS on a few key performance metrics including operation delays of the mechanical structure and software overhead in a prototype PB-level ROS system. The results show that ROS stacked on Samba and FUSE can provide almost 323MB/s read and 236MB/s write throughput, about 53ms file write and 15ms read latency via 10GbE network for external users, exhibiting its inline accessibility. Besides, ROS is able to effectively hide and virtualize internal complex operational behaviors and be easily deployable in datacenters.

CCS Concepts • **Information systems** → *Cloud based storage*

* Corresponding author: jackyao@hust.edu.cn

† Corresponding author: caoqiang@hust.edu.cn

EuroSys '17, April 23-26, 2017, Belgrade, Serbia
© 2017 ACM. ISBN 978-1-4503-4938-3/17/04…$15.00
DOI: http://dx.doi.org/10.1145/3064176.3064207

1. Introduction

It has been considered highly desirable for much the digital data generated from various sources such as scientific research, social and commercial activities to be preserved for more than 50 years or longer, if not forever [9, 23, 26, 28]. In addition, since emerging big data analytics technologies enable to further unlock the value of data, an increasing amount of data that used to be discarded on a daily basis in the past are now considered target for long-term preservation once the storage systems become more cost-effective. More importantly, increasing analytics applications that depend on mining historical data require inline accessibility to these long-term preserved data in order to produce timely results. In other words, these data should be conveniently accessed by big data analytics applications in a manner similar to accessing databases or filesystems, without the cumbersome interventions of conventional archival or backup systems.

Unfortunately, reliably preserving data for the long term remains a significant challenge [10]. Mainstream online storage media such as solid state drive (SSD) and hard disks (HDD) have limited lifecycles of up to 5 years in which data reliably stored [3, 15]. Offline storage media like magnetic tapes only guarantee 10-year data preservation [19]. For longer-term data preservation, data migration from old media to new media is inevitable, increasing risks of data loss and complexity of maintenance[14]. Additionally, these three typical mainstream storage media demand strict environmental conditions such as constant temperature and humidity, thus consuming extra amount of energy.

Optical discs are shown to be able to reliably store data in excess of 50 years without specific environmental requirements[11], far outperforming hard disks and tapes in terms of reliable life expectancy and management cost. Since each optical disc is made of a piece of plastic and multiple thin layers of coating films at sereval *um*s [18], the material and manufacturing costs of optical discs are very low. Furthermore, considering the data migration and maintenance cost, the TCO (Total Cost of Ownership) of optical-discs based storage is significantly lower than that of hard-disks or tapes based storage for long-term data preservation. Nevertheless, there are three fundamental limitations of optical discs that greatly hamper their use in any storage systems, much less a stand-alone optical storage. First, the capacity of a single optical disc is still far less than that of a hard disk or tape. Second, the maximal access throughput for an optical disc is up to 40MB/s, far lower than that for a hard disk. Third, the writing pattern for optical discs is preferred to be write-all-once, where all prepared data are burned onto discs once.

In order to overcome these limitations of optical discs, we present an optical library called *ROS*, which stands for a Rack-based Optical Storage system for long-term data storage with inline accessibility. ROS contains thousands of optical discs to improve the overall storage capacity and tens of optical drives to enhance the sustained access throughput. More importantly, ROS is inline accessible and can be deployed easily. It can be conveniently integrated into rack-scale datacenters without the introduction of an extra backup or archival system. At the heart of ROS is a novel system-level co-design of mechanical-electrical and hardware-software that provides general and virtualized POSIX storage interfaces for transparent, inline access of the long-term preserved data from the user space. The main contributions of ROS are summarized as follows.

Mechanical subsystem: To maximize the overall capacity, ROS designs a high-precision electromechanical structure to systematically organize up to 12240 optical discs, 48 drives and a server-level controller in a 42U rack. This design offers a highly compact and relatively simple mechanical structure. Each 6120 discs are placed in one of the two rotatable rollers, each with 510 trays (of 12 discs each) that are organized in 85 layers with each layer containing 6 concentric slots. ROS further develops an efficient robotic arm to automatically switch discs into/out of drives, to correctly separate and place a disc from a disc array to a drive. All mechanical components can be precisely and correctly driven by ROS in a feedback control loop with a set of sensors.

Architecture: ROS combines long-term data preservation of optical discs with fast access performance of hard disks and SSDs to consolidate the best characteristics of the three technologies, constituting an externally transparent and internally tiered storage architecture based on SSDs and/or HDDs, referred indistinguishably as disks hereafter unless noted otherwise, optical drives, and discs in rollers. All the disks and drives are connected to a server-based controller accessible by external clients via high-speed network such as 10GbE and Infiniband, making ROS a network attached storage. The disks are used as write buffer and read cache. All optical drives can be accessed in parallel to ensure high sustained throughput. This design guarantees that all write requests can be completely buffered and most read requests can hit in disks to effectively hide the tens-of-seconds latency of the mechanical operations.

Software: ROS develops an optical library file system (OLFS) to orchestrate the interplay and interaction among software, hardware, and mechanical subsystems. It offers a general and programmable POSIX interface to realize *inline accessibility* for external applications, as well as effectively hide the long mechanical delays. OLFS can automatically access and manage PB-level data across tens of thousands of discs without manually partitioning datasets across physical discs and burning discs. OLFS maintains a global file view and strategically partitions all files into Universal Disc Format(UDF)[1] disc images on disks or discs. The global view is built on a fast and small mirroring volume while all file data and their abstract path are ultimately stored in disc images. ROS can further asynchronously generate parity disc images based on a set of prepared data disc images, enhancing data reliability at the system level.

Evaluation: We ran a series of experiments to evaluate ROS performance. The latency of critical operations, such as accessing disks and discs, and mechanical behaviors, can be further analyzed. The experimental results also illustrate that ROS can effectively and transparently respond to file requests through the POSIX interface at an acceptable performance for inline accessing long-term preserved data[2].

The rest of the paper is organized as follows. Section 2 provides the necessary background and analysis to motivate our research on optical storage for long-term preserved data. The ROS mechanical and hardware designs are detailed in Section 3. Section 4 presents the design of the optical library file system OLFS. In Section 5, we evaluate the performance and functionalities of ROS. Finally, Section 6 and Section 7 respectively discuss related work and conclude the paper.

2. Background and Motivations

Long-term preserved data have become a primary source of big data. Valuable data such as scientific experiments, commercial activities, social insurance and human health records are mandated to be regularly preserved without any modification for a very long time, if not forever. Besides, more daily accumulated data, from sources such as social networking and Internet of Things, are also desired to be long-term stored because these data could produce unexpected values when mined by emerging big data analytic technologies. Therefore, long-term and inexpensive data preserving techniques are attracting more and more attention from academia and industry alike.

Unfortunately, considering that current mainstream storage media deteriorate in relatively limited lifetimes, the long-term availability of their stored data is dependent upon storage media replacement and data migration. The lifetimes of SSD and HDD are generally less than 5 years[3, 15]. Tapes can in principle reliably store data for about 10 years under constant temperature, strict humidity, and rewinding operations every two years, which are inevitable to protect tapes from adhesion and mildew[19]. Additionally, these three media are also not resilient against electromagnetic pulse and nature disasters such as flood and earthquake. These essential media replacement and data migration significantly increase maintenance cost and risks of data loss.

2.1 Optical Discs

In comparison, Blu-ray discs have been experimentally validated to preserve data for over 50 years and are potentially cost-effective data storage media. Optical discs have good compatibility. The first generation compact discs (CDs) manufactured 30 years ago can still be read using the current generation of optical drives. The Blu-ray disc technology inherits the 120mm physical feature from CDs with higher storage density. Additionally, optical discs are resilient against disasters[22], such as flood and electromagnetic pulse. They were the only digital media that survived the hurricane Katrina[27]. Considering that optical discs are each simply made up of a piece of plastics plus multiple thin layers of coating films at several *um*s, the material and manufacturing costs of optical discs are relatively low. Once the amount of produced discs exceeds a threshold where the investment of production lines for optical discs can be sufficiently amortized, the cost per GB for discs will surely be competitive with hard disks and tapes. Current media cost per GB of 25GB discs has become close to that of tapes. Hologram discs[6] with 2TB have been realized and demonstrated, although their drives are plans to be productized in two years. In the foreseeable future, 5D optical discs are poised to offer hundreds of TB capacity[29]. Assuming the use of tapes, hard disks, SSDs and blu-ray discs to build a datacenter with a capacity of 1PB for 100 years, Preeti Gupta and his group construct an analytical model to calculate the TCO of such a system [12]. Optical discs have an lifetime of more than 50 years while the lifetime of HDDs are only 5 years, so the HDD-based datacenter needs more data migration cost and media repurchase cost than the optical based datacenter. As mentioned above, the preservation of tapes requires strict environment and regular rewinding operations, thus the tape-based datacenter needs more operational cost than the optical disc based datacenter. The simulation result shows that the TCO of an optical disc based datacenter is 250K$/PB about 1/3 of an HDD-based datacenter, 1/2 of a tape-based datacenter. Consequently, industry and academia have also begun to consider store long-term data on optical discs as an alternative approach.

Despite of these potential advantages, the current capacity and performance of Blu-ray discs are individually far lower than those of hard disks. Even though 300GB Blu-ray discs are now becoming increasingly popular, they are still an order of magnitude smaller in capacity than normal hard disks with 4TB or 8TB per-disk capacity. Additionally, the basic reference speed of a Blu-ray disc is at most 4.49MB/s, which is defined as 1X. Current standard reference speeds for 25GB and 100GB discs are respectively 6X and 4X[17]. The maximal speed obtained in our experiments are 12X for 25GB discs and 6X for 100GB discs, which are far lower than the speed of hard disks of almost 150MB/s.

Most recordable optical discs are write-once-read-multiple (WORM) while re-writable (RW) discs can re-write with relatively low burning speed (2X), limited erase cycle (at most 1000) and high cost. Optical drives at high voltage burn a series of physical grooves on the unused smooth surface of discs. It is best to steadily and sequentially burn data once into discs to ensure high quality and low risk of errors. An entire prepared disc image is burned into a disc once, referred to as the write-all-once mode. Optical drives also support the Pseudo Over-Write mechanism where the drive can write multiple data tracks into a disc, with each track representing an independent disc image. An optical drive first takes tens of seconds to format a predefined metadata area and writes data. When over-write happens, the previously burned area cannot be used and the drive needs to format a new metadata area and then writes data. This mechanism causes capacity loss and performance degradation, and thus is not recommended.

2.2 Optical Library

In order to overcome the limitations of individual optical discs, optical libraries have been proposed to increase system-level capacity by leveraging the combined capacity of a large number of discs and improve performance by deploying multiple optical drives to access discs concurrently. The first generation optical jukebox only contained up to hundreds of discs in relevant physical slots and accessed by 1-6 drives. Their mechanical appliances can deliver discs between drives and slots. Current large-scale optical libraries, such as Panasonic LB-DH8[4] and Sony Everspan[24], can further deploy thousands of discs and tens of drives. These discs are grouped into a series of disc arrays as a logical block volume with an internal RAID redundant mechanism. The disc arrays can solely mount/unmount to a host via the SCSI interface. LB-DH8s are 6U racked-devices. Everspan has a controller rack unit consisting of drives and up to 13 additional expansion rack units with discs.

However, these optical libraries are more like tape devices rather than storage nodes deployed in current cloud datacenters, making them difficult to be integrated into datacenters. Current tape and optical libraries generally rely on a dedicated backup system running on a front host to manage all data on media in an off-line mode. The typical backup process can involve datasets collection, catalog creation or update, compression or deduplication, transforming datasets into a specific format suitable for tapes, then copying these formatted data into tapes. The restore process essentially reverses the above procedure. Besides, raw data on tapes are not readable for applications without interpretation of the backup system. In order to address this problem, IBM developed LTFS[20] that allows its files on tapes to be directly accessed by user applications through a standard POSIX. However, LTFS is only based on single tape like UDF for discs.

2.3 Motivation

Optical discs have exhibited their advantages in long-term data preservation. However, their disadvantage in capacity and performance will persist for some time to come. An optical library is expected to maximize disc placement density within a limited physical space and provide a reliable and simple mechanical structure. Furthermore, an optical library is also designed to effectively manage discs and their storing of data, as well as to hide disc boundary, internal specific access behaviors and mechanical delays.

Additionally, for long-term data preservation, considering that hardware, software and mechanical components are not likely to have the same lifetime as discs and part of discs might be lost, data on survived discs should be by and large self-descriptive, independently accessible and understandable.

More importantly, in the era of big data, long-term preserved data are required to be conveniently and directly searched and retrieved through programmable interfaces by big data analytics applications without the aforementioned extra backup/archival system intervention. Therefore, optical libraries should provide a persistent online view of their data so that the data can be shared by external clients using standard storage interfaces that can be easily integrated and scaled in cloud datacenters. Furthermore, as storage nodes in current datacenters, optical libraries should have in-place computing capacity to process their data locally.

In summary, optical libraries as storage nodes used in cloud datacenters should be reliable, self-manageable, scalable, low cost, ease of use and capable of local processing of data. To this end, we design and implement a Rack-based Optical library System (ROS) with a sophisticated by effective mechanical-electrical integrated system with a full visualized file view and tiered storage consolidating SSDs, hard disks and discs. In next section, we detail our design and implementation considerations for ROS, including mechanical, hardware and software architectures and their co-design.

3. System and design

ROS is a complex mechanical-electrical integrated optical library system designed to achieve PB-level long-term data preservation at low cost. ROS is also designed to be a storage node that can be integrated into datacenters with inline data accessibility. In this section, we detail the system architecture, mechanical and hardware designs of ROS.

3.1 System Architecture

The system architecture of ROS is shown in Figure 1. The mechanical structure is comprised of one or two rollers each built with physical frame, robotic arms, a serial of motors and sensors. A roller with a height of 1.67m is a rotatable cylinder with the diameter of 433*mm* and contains up to 6120 discs. The hardware units include a server-level controller, Programmable Logic Controller(PLC), a set of hard disks and SSDs, a set of optical drives. The controller is a powerful server connecting all SSDs, hard disks and optical drives via a set of SATA interfaces extended by PCIe3.0. The SSD and hard disk are referred indistinguishably as disks hereafter unless noted otherwise. PLC controls the rollers, robotic arms, motors and sensors to move discs between optical drives and specific slots in the frame automatically and precisely. PLC can communicate with the controller via TCP/IP to perform mechanical operations. Besides, an optical library management system which will be detailed in the next section, is running on the controller to manage ROS and serve user requests effectively.

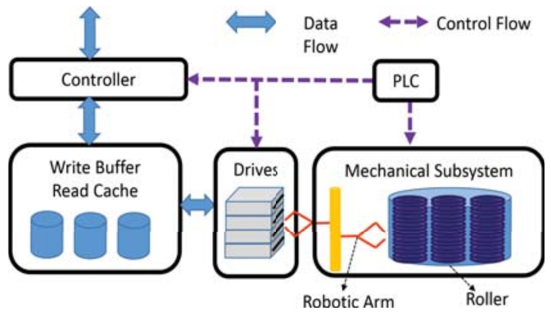


Figure 1. A conceptual view of the ROS architecture. The mechanical structure contains rollers and robot arms. Hardware consit of a powerful server with multiple hard disks, SSDs and optical drives.

3.2 Mechanical Subsystem

ROS is housed in a 42U standard rack containing up to 12240 discs. The rack can deploy 1 or 2 rollers each containing 6120 discs in 510 trays (12 discs/tray) organized in 85 layers. There are 6 trays in each layer arranged in a lotus like shape. 12 discs placed in each one tray are referred indistinguishably as a *disc array* hereafter unless noted otherwise. Figure 2 shows a schematic representation of the mechanical structure of a roller, with a cylinder containing 6 x 85 (depth) x 12 (height) discs. The roller can rotate back and forth. A robotic arm can move vertically, fetching and separating a disc array. Further, ROS is able to deploy 1-4 sets of optical drives with each set containing 12 optical drives. These drives can burn data into discs or read data from discs in parallel.

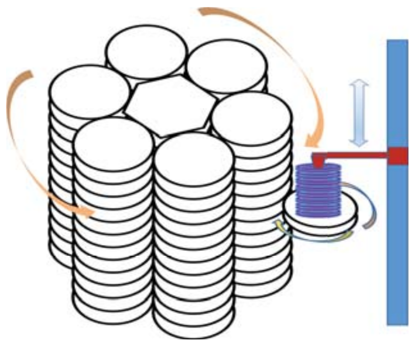


Figure 2. A schematic view of the ROS mechanical organization. The roller rotates and the robot arm moves vertically. These two simple movements combine to load/unload disc arrays.

All discs are placed in trays in the roller and can be automatically loaded into drives on demand. Each tray can fan out/in independently. During the process of tray fanning out, an outer side hook of this tray is locked by the robotic arm and an inner side connector of this tray rotates with the roller to a specific angle. Once the tray is fanned out, the robotic arm can fetch all 12 discs in this tray and quickly lift them up to a position atop all optical drives. Then, the robotic arm releases the lock of the tray, the roller reversely rotates the same angle and the opened tray can fan back into the roller. Afterwards, these 12 optical drives are simultaneously opened, and the robotic arm begins to separate the bottom disc of the disc array, places this bottom disc on the uppermost drive. Subsequently, this drive withdraws its tray and starts to access data on the disc. This disc-separating operation is performed one by one from top to bottom until these 12 discs are all placed into the 12 drives. When a disc array needs to be unloaded from drives to tray in the roller, the robotic arm fetches the discs on the ejected tray of the uppermost drive one by one from top to bottom. Afterwards, this disc array is placed into the specific tray in the roller. Finally, the opened tray is closed with the roller rotating reversely.

The ROS design attempts to simplify both the mechanical and tray structure. Traditional optical libraries and tape libraries, such as Panasonic Data Archiver LB-DH8 series, have a delicate magazine to contain storage media and place all magazines in fixed slots. To carry a magazine between drives and specific slots, the magazine must be rejected from the slot as whole, followed the robotic arm needs to moving the magazines in three dimensions, before a set of discs on the magazine are precisely separated and placed into a set of drives. The magazine based structure not only increases the mechanical complexity and but also reduces the disc placement density. The three-dimension movement of the robotic arm also needs more precise control from motors and has longer executing time. In contrast, ROS spins the roller and moves the robotic arm only in the vertical direction, thus reducing mechanical movement. Additionally, the robotic arm only carries a single disc array rather than a magazine, eliminating a complex magazine structure. Therefore, these design choices of ROS reduce the mechanical complexity of both the robotic arms and trays, improving mechanical reliability and disc physical placement density. One concern is that this integral-rotation of the roller might draw more power than the traditional systems. In fact, rotating the entire roller consumes less than 50 watts and is not frequently performed.

Our mechanical design also attempts to reduce the mechanical delay. The mechanical operations with the longest delays are to separate or collect a disc array one by one into drives, both of which take almost 70 seconds. Precisely scheduling movements of the roller and robotic arm in parallel can further reduce the delay of conveying discs, which can save up to almost 10 seconds. However, more complex design and scheduling policies seem to only slightly reduce this delay. In order to hide the delay of these mechanical operations, ROS introduces a tiered storage architecture by utilizing disks as a fast access tier to absorb as many I/O requests as possible.

3.3 Electronic Subsystem

The key components of ROS electronic subsystem are two controllers, the PLC (Programmable logic Controller) controller and the system controller (SC). The former defines an instruction set to execute basic mechanical operations, while the latter orchestrates all operations of PLC via an internal TCP/IP network, both hardware and software, to complete data access operations, as well as to manage the overall library and data. All the software modules run on the SC.

More specifically, PLC manages all motors and monitors all sensors in real time. All mechanical operations can be executed correctly by precise feedback control. ROS has three kinds of motors: motors that to rotate the roller, motors that to move robotic arms up and down, tiny motors on the robotic arm to separate discs. ROS monitors all the sensors to continuously track the current mechanical states and to calibrate the current operations. For instance, ROS partitions discs into drives at the 0.05*mm* precision using a set of range sensors.

SC is a server-level controller with two Xeon processors, 64GB DDR4 memory, two 10Gbps NICs, 4 PCIe3.0*4 slots and 4 PCIe3.0 HBA cards connecting hard disks and optical drives (24 per rack by default). ROS also supports infiniband and Fibre channel (FC) networks that are commonly used in storage area network (SAN) scenarios. ROS can utilizes 10Gbps networks to connect clients in a shared network attached server (NAS) mode, providing more than 1GB/s external throughput, which is suitable for datacenter environments. All optical drives with SATA interfaces can be ultimately aggregated into the internal PCIe 3.0 HBAs. Actually, ROS can deploy more optical drives and disks. All disks can be further configured as multiple RAID volumes to improve overall throughput and reliability. All optical drives are grouped into sets of 12 drives each. All optical drives are off-the-shelf half-height drives with far lower cost than DH8 with dedicated slim drives and Everspan with customized multiple-heads drives. Since all drives can read/write data on discs in parallel, ROS relies on deploying more drives to increase its overall bandwidth of accessing discs.

In ROS, both disc burning and disc load/unload operations have long delays at the minute level. The current standard write speed of a drive for 100GB discs is 4X, or about 20MB/s. It takes more than an hour for a drive to complete burning a 100GB disc at up to 6X speed. For 25GB discs and 50GB discs, the empirical write speed under a pair of well-matched drive and disc can reach 10X, or 45MB/s. It takes more than 10 minutes to burn a 25GB disc. Additionally, the time spent on loading requested discs into free drives is about 70 seconds. When all drives are not free, it will take another 70 seconds to unload discs from drives. These time-consuming burning and mechanical operations cannot be in the critical I/O path of the foreground applications. Traditional optical and tape libraries use dedicated backup systems to hide these long delay time, which also renders applications unable to access data directly on the libraries.

ROS uses a number of disks as the write buffer and read cache for optical discs, constituting a two-level tiered storage structure. ROS deploys at least two SSDs configured as the RAID-1 mode to serve as a metadata volume. The rest of the disks are configured into multiple RAID-5 arrays. Data transferred to ROS are first written to these arrays, and then reorganized and burned to discs asynchronously. These arrays also act as read cache to absorb data read requests. Since each hard disk has almost 150MB/s read/write sequential throughput, each RAID-5 can not only tolerate single disk failure but also offer more than 1GB/s throughput. In fact, ROS can deploy more disks according to requirement. Two or more independent RAIDs can effectively support multiple I/O streams such as writing data from clients onto the disk buffer and burning data from the disk buffer to discs, avoiding potential performance degradation caused by the interference between these concurrent I/O streams.

4. Optical Library File System

4.1 OLFS Overview

Based on the mechanical and electronic platforms, we further design and implement an optical library file system, OLFS for short, as a global virtualized file system on the tiered storage architecture. OLFS is able to orchestrate all electronic and mechanical resources effectively to serve user's I/O requests, as well as hide the internal tiered storage and complicated electromechanical behaviors sufficiently. To this end, OLFS presents several novel techniques: metadata and data decoupled storage, preliminary bucket writing, unique file path, regenerating update, and delayed parity generation.

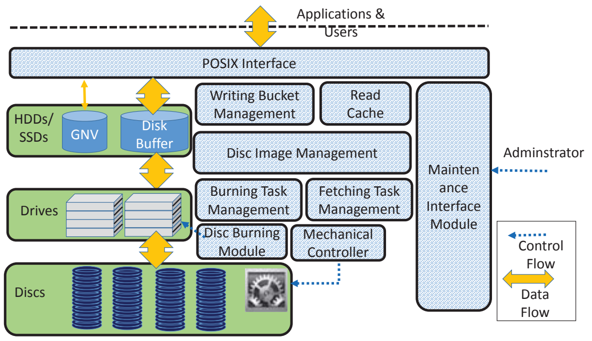


Figure 3. OLFS modules and data flow in ROS. OLFS provides a POSIX file system interface and hides the internal disc and mechanical operations.

OLFS is comprised of nine software modules as shown in Figure 3. OLFS provides a *POSIX Interface* module (PI) as a uniform file/directory external view for users in the network-attached storage (NAS) mode. The fact that this NAS mode has been the most widely used in storage services makes ROS easily deployable in most storage systems. OLFS stores all files' mapping information in a small and fast volume, referred to as Metadata Volume (MV). OLFS first generates the global metadata of all incoming files in MV, and then writes their actual data into a couple of updatable buckets formatted in UDF (universal disc format) in the disk write buffer. A fully filled and closed bucket is transformed into a data disc image that is a basic switchable data unit between the write buffer and discs. The *Writing Bucket Management* module (WBM) creates, uses and deletes buckets. The *Disc Image Management* module (DIM) generates parity disc images from a set of disc images according to predefined redundant strategies. The *Burning Task Management* module (BTM) creates a new disc burning task when a set of unwritten disc images are ready. A new disc burning task cannot perform immediately until its requested hardware resources are free. When these resources are available, the disc burning task further invokes the *Mechanical Controller* module (MC) to perform the relevant mechanical operations to load empty discs into drives. Then the *Disc Burning* module (DB) burns images onto the discs. Hence, the file writing process and disc burning process can perform asynchronously.

OLFS considers a disc image as a basic container to accommodate files. Each disc image has the same capacity as the disc and has an internal UDF file system. Therefore, disc images as a whole can swap between discs and disks. Each disc image has a universal unique identifier. After being directly mounted to a OLFS internal directory, disc images can be accessed by VFS (virtual file system). OLFS defines a disc array index *DAindex* to maintain the state of each disc array in one of the three states, ”Empty”, ”Used”, ”Failed”. Initially, all entries in *DAindex* are marked as Empty. Then $DAindex_i$ will be modified to ”Used” when disc array i is used. When the disc burning task for disc group j has failed, $DAindex_j$ will be set to ”Failed”. OLFS also uses a disc image location index *DILindex* to record each disc image identifier and its own physical location.

For a file read request, OLFS looks up its index file in MV to determine its image identifier. This image can be stored in buckets, disc images in the disk buffer, or discs in drives or in the roller. In the last case, OLFS has to invoke mechanical operations to fetch the requested discs. Once the requested disc image is ready, it is mounted and accessed to read the requested file. Considering that recently and frequently read data are likely to be used again according to data life cycles, Read Cache (RC) retains some recently used disc images according to a LRU algorithms. Once requested data miss in RC and buffered disc images, the *Fetching Task Management* module (FTM) generates a new fetching task to capture the relevant discs into drives. The current design of OLFS only considers a disc image as a cache unit, sufficiently exploiting spatial locality. Considering that the size of the disk buffer is more than 50TB, its capacity miss rate is relatively low. Certainly, the read cache also can use in finer grain as files or prefetch some files according to specific access patterns[10]. A specific cache algorithm should sufficiently exploit the potentials of such large buffer and access patterns. We leave these works to the further combining empirical use cases.

Note that MC not only communicates with PLC, but also schedules disk burning and fetching tasks to optimize the usage of mechanical resources. Finally, OLFS also offers a *Maintenance Interface* module (MI) to configure and maintain the system by an interactive interface for administrators.

4.2 Global Namespace Mapping

Different from current tape and optical libraries, OLFS provides a shared global logical namespace for all users based on the standard POSIX interface. This means that clients can use the absolute path of a file in the global namespace to effectively locate its actual data in ROS. This mapping process is illustrated in Figure 4.

OLFS uses MV to maintain updatable maps between millions of files and thousands of discs. When a file write operation comes, its global metadata is created and updated in the MV. Meanwhile its data is written into buckets on the disk write buffer. Traditional local file systems such as Ext4 use file inodes to record their metadata such as data block addresses and attributes, as well as employ directory inodes to maintain a hierarchy namespace. In these scenarios, both inode blocks and data blocks are in the same block address space. In contrast, ROS files are physically distributed among thousands of disc images on disk write buffer or discs while their global medadata are stored in MV. This means that metadata and data storage can be physically decoupled,

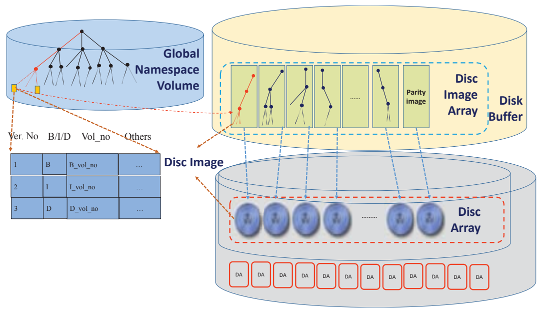


Figure 4. Mapping from the global namespace to discs. The global namespace and all index files are stored in MV. Each data image contains a part of the global namespace.

avoiding that the high performance of accessing small metadata files is hampered by the data storage with long I/O latency.

Any entry in the global namespace, including file and directory, has its corresponding index file with the same file name in MV. However, MV index files do not have actual file data, but only record the locations of their data files in the form of bucketID, image ID, or disc ID. This design sufficiently leverages the maturity and advantages of ext4 and UDF, reducing the development complexity.

The index file is organized in the Json standard format for its ease of processing and translation[5]. Its typical size is 388 bytes. In addition to the Json tag, the location information is 128 bytes. This index file structure can ensure platform independence and interchangeability. In order to support file appending-update operations, multiple file version entries for a file can be recorded into the index file. Each entry takes 40 bytes. To reduce storage wastage, the block size of MV can be set to 1KB offering about 15 historic entries. Meanwhile the inode size in MV is set to the smallest 128 bytes. MV with 1 billion files and 1 billion directories only needs about 2.3 TB, which is only 0.23% of the overall 1PB data capacity.

MV files and directories are relatively small, frequently accessed and updated. To accelerate the metadata access, MV is built on a small RAID-1 formatted as ext4 file system. The small RAID-1 can further be configured upon a pair of SSDs. Furthermore, the well optimized ext4 file system adopts buffer cache, dentry cache, and journaling mechanisms, to ensure the performance of small files and directory operations on MV[7].

Besides index files, all system running states and maintenance information are also stored in MV in the Json format. Once ROS crashes, OLFS can recover from its previous checkpoint state with all state information stored in MV.

OLFS must ensure the durability and consistency of its global file namespace since the file-directory structure is critical for looking up their actual storage locations. More importantly, the global file namespace also embodies the semantic relationship among files. For WORM discs, OLFS is actually a traceable file system and records the creating and updating tracks of files. This information of files is helpful in analyzing data evolution. Therefore, MV is periodically burned into discs. Once MV fails, the entire global namespace can be recovered from discs. This mechanism is very essential for long-term data perseveration. As an experiment, ROS took half an hour to recover MV from 120 discs.

This namespace mapping mechanism can also be extended to support other mainstream access interfaces such as key-value, objected storage, and REST. OLFS can also provide a block-level interface via the iSCSI protocol. Certainly, in the future, if required, we can design dedicated mechanisms to simplify the mapping between these specific interfaces and discs.

4.3 Preliminary Bucket Writing

In order to effectively eliminate long disc burning delay and file format transformation for file writes from clients, ROS introduces a preliminary bucket writing (PBW) mechanism. The actual data of an incoming file is written into an updatable UDF bucket on the disk write buffer, its index file is simultaneously created in MV to record the bucket image ID. As soon as the file data have been completely written, OLFS immediately acknowledges the completion of the file write.

OLFS initially generates a series of empty buckets, each of which is a Linux loop device formatted as an updatable UDF volume. When an empty bucket begins to receive data, OLFS allocates an image ID to it. After the bucket is filled up, it will transit into a disc image with the same image ID. The bucket can be recycled by clearing all data in it. The unburned disc images should be stored in the disk write buffer before they are burned into discs according to certain pre-defined burning policies. The tiered writing process is illustrated in Figure 5.

PBW is very suitable to the features of WORM discs. The buckets can not only absorb all write and partially updated data, but also reorganize these data in the UDF format designed for discs. PBW supports the write-all-once mode that can achieve the highest performance and reliability since sequentially burning discs can reduce seek operations of optical head and improve burning quality.

4.4 Unique File Path

A file in OLFS has two file paths, a logical path in the global namespace and a physical path in the UDF file system on disc images. In order to avoid transformation between these two paths, OLFS uses the global file path of a file as its internal file path in disc images, referred to as unique file path. For a file, its ancestor directories up to the root in the global namespace should be generated as UDF directories in the disc image. As a result, each image has a root directory and a relevant directory subtree of an entire global directory tree.

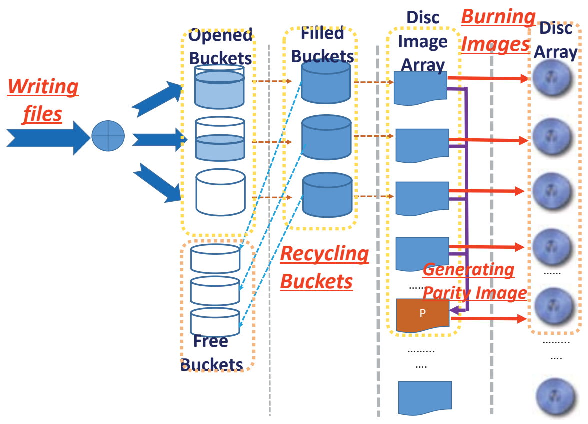


Figure 5. File writing process on the tiered storage. Data are first written to empty buckets. The full buckets are closed and become disc images, then parity image is generated asynchronously. Finally, these disc images are burned to a free disc array.

We argue that the unique file path approach based on redundant directory is worthwhile. Although it slightly increases directory data in images, this approach simplifies the content of the index file and the process of accessing files. The index file only needs to record its image ID, reducing the capacity requirement for MV. OLFS can use the unique path to locate a file in both MV and images. Once the image is mounted, the requested file can be located by executing lookup operations of the UDF file system. Additionally, this approach allows discs to preserve the semantic information as the relationship among files by copying subtree from the global namespace. Even if all electronic and mechanical components failed, all or partial data can be reconstructed by scanning all survived discs. This is very essential for long-term data preservation.

4.5 Partitioning Files into Buckets

OLFS should automatically partition datasets into a series of separated images. The default policy is first come first service. This means that all incoming files are sequentially written into an opened and not-full bucket. In order to create the unique file path of a writing file, its ancestor directories should be created or updated. Since unclosed buckets are updatable, the write process modifies the corresponding directory information dynamically. In the UDF file system the basic block size is 2 KB and cannot be changed. Hence, each file entry size is allocated at a minimum of 2KB. In the worst case scenario where all files are less than 2KB plus extra corresponding 2KB file entry, the actual space to store data is only half of the bucket. Note that before the bucket is closed, its UDF metadata can be update-in-place in disks and its defined metadata zone is not fixed. Once the free space of a bucket is insufficient to accommodate both a new file and its directory, the bucket is marked as full and closed, guaranteeing that the image size does not exceed the disc capacity.

Note that OLFS does not know the actual size of an incoming file ahead of time under the POSIX write semantics. When an incoming file data exceeds the free space of the current bucket, the writing process has to choose another empty bucket to sequentially write the rest of this file, causing the file to be divided into two subfiles and written into two consecutive images. Therefore, its corresponding index file needs two relevant entries to record the subfile locations. To correctly ensure the reconstruction of the global namespace without VM, OLFS also creates a link file on the second subfile image to point to the first subfile.

4.6 Updating Mechanism

OLFS based on WORM discs can still implement file update in its global file view. If an updating file is still in an opened bucket with sufficient free space, the file can be simply updated. Otherwise, instead of update-in-place, OLFS has to create a new file replacing its old file. When this modified file is generated in another bucket, its index file also adds an entry to record the new location with an increasing version number. As mentioned above, an index file with 2 KB can store up to 15 entries. When all 15 entries have been used up, the first entry will be overwritten. After burning current MV into discs, all index files will only store the recent entries.

Note that an old file in images is not actually removed, since OLFS can obtain any of its foregoing versions by searching the old entry in the current index file or old index file in MV images. Hence, OLFS can conveniently implement data provenance and data audit.

4.7 Redundancy Mechanism

The sector error rate of archive Blu-ray discs is generally 10^{-16}. A disc array of OLFS typically consists of 12 discs with 11 data discs and 1 party disc organized in a RAID-5 schema. In this case, the whole error rate of a disc array is about 10^{-23}, which can satisfy the reliability demand for enterprise storage. In order to obtain lower error rate under some rigid circumstances, OLFS can further configure higher redundancy with 10 data discs and 2 parity discs organized in a RAID-6 schema to obtain about 10^{-40} system error rate.

OLFS employs this system-level redundancy mechanism instead of forced write-and-check approach that almost halves the actual write throughput. Actually, ROS attempts to effectively tradeoff among reliability, storage resources and actual write throughput.

Additionally, OLFS does not generate parity data synchronously when data are written into images. On the contrary, parity disc images are generated only when all data disc images in the same disc array have been prepared. In this case, the parity strips can be created according to the corresponding data strips and written into a parity image sequentially. Note that the parity image is not a UDF volume. Only after a set of data and parity disc images being prepared, OLFS sequentially burn all images in the array into discs concurrently. On the other hand, disc sector-error checking can be scheduled at idle times and can periodically scan all the burned disc arrays to check sector errors. When sector errors occur, data on the failed sectors can be recovered from their parity discs and the corresponding data discs in the same disc array under the given tolerance degree. The recovered data can be written to new buckets and finally burned into free disc arrays.

OLFS generates parity disc images asynchronously. However, the parity generating process is I/O intensive. It requires reading all stripes in its corresponding 10 or 11 data disc images from disks, and then creating the parity stripe and writing it into the parity disc image on disks, which produces intensive read and write streams concurrently. Therefore, there exists four kinds of concurrent intensive data flows on disks: (1) User writes data to disks, (2) the parity maker reads data from disks, (3) the parity maker writes parity data back to disks, (4) optical drives read images from disks to burn discs. These four I/O streams might interfere each other to worsen overall performance. To avoid this problem, ROS can configure disks into multiple volumes of independent RAIDs and further schedule these I/O streams to different volumes at same time.

4.8 Implementation

To accelerate the development of ROS and to ensure system maturity, the current OLFS is implemented based on FUSE[21] to customize file operations effectively under the POSIX interface. Additionally, the FUSE framework is a user-space file system implementation that allows us to debug and test OLFS in much easier way than kernel development. However, FUSE introduces substantial kernel-user mode switching overhead since its file system operation is implemented in the user space.

By default, FUSE flushes 4KB data from the user space to the kernel space each time, resulting in frequent kernel-user mode switches and significant overheads. OLFS sets the mount option *big_writes* to flush 128 KB data each time, thus improving the write performance.

In order to further eliminate FUSE performance penalty in some performance-critical scenarios, we provide a direct-writing mode where incoming files are directly transferred to the SSD tier at full external bandwidth through CIFS[13] or NFS, then asynchronously delivered into OLFS.

For all file writes and file reads that hit on disks, the overall performance is primarily affected by OLFS, FUSE, UDF, ext4, and the underlying storage because the burning of discs and mechanical fetching are executed asynchronously without involving the critical I/O path. But for file reads that miss on disks, their response times can increase by orders of magnitude depending on their stored locations: (1) a disc in drive; (2) a slot with free drives; (3) a slot while all drives are just occupied; (4) a slot while all drives are being burned.

In the fourth case, there are two policies. One is waiting for the burning task to complete, where the residual recording might take from several minutes to more than an hour. The other is immediately interrupting the current disc array burning process and switching this array out of the drives with the appending-burn mode. After the requested files are read into disks, the interrupted disc array is re-loaded into the drives and continued to be burned. This policy needs to format the reserved metadata zone on discs ahead of time.

For file reads that miss in both disks and drives, the long mechanical delay might lead to read timeout. We also design a forepart-data-stored mechanism to store the forepart (eg. 256KB) of data files in their corresponding index file in MV. It ensures that the first word of the file can quickly respond within 2 ms. After that, OLFS sends the forepart data continuously at a slow but controllable rate until the requested disc is fetched into drives. This approach avoids read timeout continuously but introduces extra storage overhead. We will leave further designs to our future work, optimizing for empirical workloads.

5. Evaluations

In this section we will evaluate and analyze the performance of ROS.

5.1 Experimental Setup

Our ROS prototype deploys two rollers with *6120 100*GB optical discs each, a server controller, 24 optical drives, 14 hard disks and 2 SSDs. Thus, the ROS prototype has a total capacity of *1.16* PB. SC runs Ubuntu Server 14.04 and has two Xeon processors, 64GB DDR4 memory, two 10Gbps NICs, 4 PCIe3.0*4 slots and 4 PCIe3.0 HBA cards connecting SSDs, hard disks and optical drives. The idle and peak powers of ROS are 185W and 652W respectively. The two 240GB SSDs and fourteen 4TB hard disks are configured into 3 RAID sets, respectively consisting of 2 SSDs, 7 HDDs, and 7 HDDs. The 2 SSDs are configured as a RAID-1 to store system metadata as MV. The two sets of HDDs are each configured as a RAID-5. The 24 optical drives are the Pioneer BDR-S09XLB with SATA interface and supporting up to 128GB BDXL disc each with peak power 8W. Clients are connected to ROS via a 10GbE network.

5.2 Evaluation Methodology

As ROS is a tiered storage system, file write operations can be fully absorbed by the disk tier, making ROS file write latency equal to that of disk access via file system. However, file read operations in ROS may have latency ranging from several milliseconds to minutes as shown in Table 1. To further analyse the latency, all ROS operations can be divided into three types, namely, *disk operations* involving file writes and reads that hit on disks, *disc operations* involving disc writes (i.e., burning) and reads, and *mechanical operations* involving fetching discs mechanically. These three types of operations are relatively independent and exhibit large delay discrepancies. In fact, the latency for file read/write can be primarily determined by these three types operations and their combinations based on the requested file location.

File location	Read Latency(s)
Disk bucket	0.001
Disc image	0.002
Disc in optical drive	0.223
Disc array in the roller with free drives	70.553
Disc array in the roller and drives are not working	155.037
Disc array in the roller and all drives are busy	minutes

Table 1. Read latency from different file locations.

In order to evaluate the performance overhead of OLFS, we use filebench[8] with the singlestream workload(default 1 MB I/O size) to compare the throughput under five configurations, namely, *ext4+FUSE* - FUSE based on ext4 with the requests routed through an empty FUSE module, *ext4+OLFS* - OLFS based on ext4, *samba* - samba on ext4, *samba+FUSE* - FUSE via samba, and *samba+OLFS* - OLFS via samba. The baseline throughput is ext4 on one of the RAID-5 volume.

To accurately measure the I/O latency caused by OLFS precisely, we add timestamps in OLFS code to trace the internal OLFS operation when file I/Os takes place on *ext4+OLFS* and file I/Os take place on *samba+OLFS*. We write and read a file with 1 KB data through OLFS 50 times to measure the average execution time of each OLFS internal operations. To ensure consistency of index file and data, the current OLFS uses direct I/O without cache. It means that each OLFS operation leads to an actual I/O of underlying storage. We choose to write and read file with 1 KB size to eliminate effect of the data transfer time of file, thus get the accurate latency brought by index file.

The current reference recording speeds for 25GB and 100GB discs are respectively 6.0X and 4.0X. The recording speed is strictly limited to ensure the recording quality at high storage density. In ROS, we make drives work at their peak throughput regardless of recording quality because OLFS introduces the inter-disc redundancy and delayed check mechanisms to enhance the overall reliability. We measure the recording speed of single disc and disc array with 25GB and 100GB disc respectively.

To quantitatively assess the performance impact of the mechanical operations, we test the latencies of critical mechanical operations: loading and unloading disc arrays between the drives and trays in the roller. As the start position of the robot arm is near the uppermost layer, we choose to measure the average operation time of loading/unloading trays in the uppermost layer which takes the shortest operation time and lowest layer which takes the longest operation time.

In section5.3, we evaluates the OLFS performance of file writes and reads that hit in disks. Then, the disc burning and read performances are discussed in Section 5.4. Finally, the mechanical delays is presented in Section 5.5.

5.3 File Writes and Reads in Disks

Note that ROS has a large disk capacity used as write buffer and read cache with more than one hundred TB. As a design objective, OLFS aims to buffer all file writes and most file reads on disks. Therefore, in most cases, the overall performance of ROS is mainly determined by disk operations when file writes and reads hit in disks. In this case, as the current OLFS implementation is stacked on the ext4 file system via the FUSE and UDF file systems, the latency of file access is heavily affected by three primary factors: (1) the underlying storage; (2) the overhead introduced by FUSE and UDF; (3) the OLFS-specific processes. OLFS can directly benefit from the performance improvement of the underlying storage, FUSE and UDF without any modification.

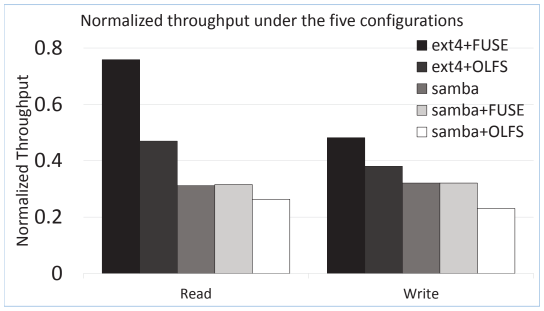


Figure 6. Throughput under the five configurations, normalized to that of ext4.Filebench uses the singlestreamread and singlestreamwrite workloads. The average I/O size is 1 MB.

The throughput of ext4 on the underlying RAID-5 volume is 1.2GB/s for read and 1.0GB/s for write. The throughput results of the five configurations, normalized to throughput of ext4, are shown in Figure 6. The results indicate that *ext4+FUSE* underperforms ext4 in throughput by 24.1% for read and 51.8% for write due to kernel-user mode switches. *Ext4+OLFS* further causes 28.9% read and 10.1% write performance loss compared to *ext4+FUSE*. In NAS mode, *samba* leads to about 68.9% read and 68.0% write throughput degradation of ext4. We can see that *samba* and FUSE introduces most of the performance loss which can be further optimized. For the most common user case *samba+OLFS*, OLFS can provide throughput of 236.1 MB/s for read and 323.6MB/s for write, which is acceptable for long-term storage system[2].

Figure7 illustrates the breakdown and execution order of the internal OLFS calls corresponding to four OLFS operations. File write operation of *ext4+OLFS* can be divided into five consecutive OLFS internal operations: (1) stating index file; (2) no file found, creating a new index and data file; (3) stating file again; (4) writing to data file on UDF; (5) releasing and closing the files. Likewise, the OLFS file read operation can also be broken into three sequential internal steps: (1) stating index and data file; (2) reading data file; (3) releasing and closing the files. Each internal operation in OLFS takes almost 2.5ms in average and there are kernel-user mode switch time between each two internal operations. As a result, the latency of file read and write for OLFS are 9ms and 16ms respectively. In the case of *samba+OLFS*, writing new file increases extra 7 stat operations. The average latencies of file write and read increase to 53ms and 15ms accordingly.

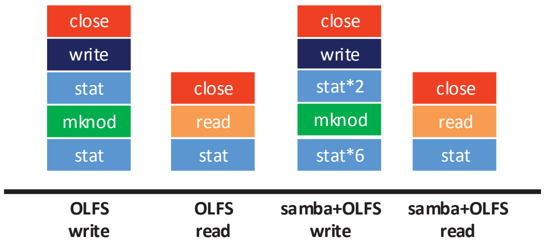


Figure 7. Four OLFS operations and their respective internal OLFS operations. File I/O through OLFS invokes multiple internal OLFS operations, resulting in long latency.

Although OLFS causes performance loss of the underlying storage due to currently used FUSE implementation, the latency for accessing a file is lower than 60ms regardless of file size, which is far better than conventional archival system which has minutes-level latency[20]. In fact, OLFS still has sufficient design space to further optimize its performance.

5.4 Optical Drive Performance

When a file read misses on disks and hits on discs in drives, the disc read is in the critical I/O path. The delay of reading file on discs is further broke down into three parts as drive mounting disc with about 2 seconds delay, mounting disc into local VFS with about 220ms delay, and seeking files on discs with about 100ms delay. The first two procedures occur only when the drive is in the sleep state.

Disc capacity (GB)	Single drive speed for reading (MB/s)	Aggregate drive speed for reading (MB/s)
25	24.1	282.5
100	18.0	210.2

Table 2. Optical drive read speeds for 25GB and 100GB discs. Although each drive works independently, the aggregate read speed is close to 12X of the single drive read speed.

In general, with the exception of the above case, drives only need to read and burn discs in an asynchronous and batched manner, which is not in the critical I/O path. Its performance is determined by actual drive access throughput. The recording speed of single drive and 12 drives of 25GB optical disc are respectively shown in Figure 8 and Figure 9. Note that not all drives start to burn data at the same time.

When burning 25GB optical discs, the default burning quantity detection mechanism of the writing-and-checking mode is turned off to achieve high recording speed. The single drive recording speed is gradually increased from 1.6X in the inner tracks to 12.0X in the outer tracks. The average recording speed is 8.2X. All 12 drives concurrently work to reach the peak recording speed of about 380MB/s. But this peak speed is maintained for only a short period of time. The average recording speed for 25GB disc array is 268MB/s. Since these drives are not completely synchronized, the average recording time for a single 25GB disc is 675 seconds, but finishing recording a disc array needs 1146 seconds.

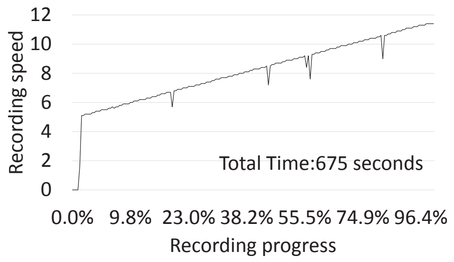


Figure 8. Single drive recording process for 25GB disc. The burning speed is increasing from 4X to alomost 12X.

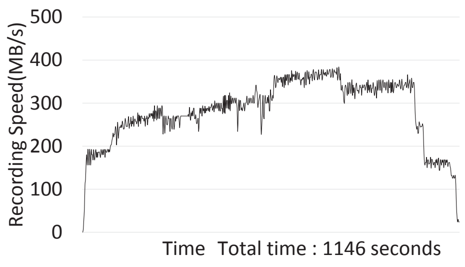
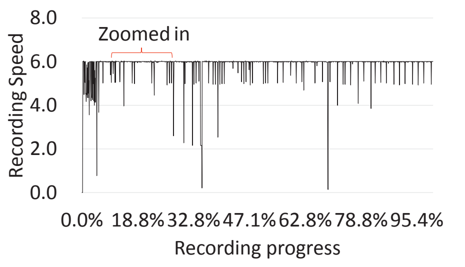
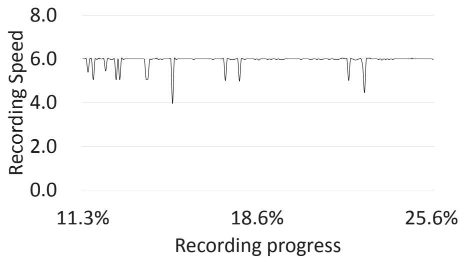


Figure 9. Aggregated throughput of 12 drives burning 25GB discs.

A regular drive burns 100GB discs at 4.0X. While we use a dedicated Pioneer BDR-PR1AME drive to burn 100GB optical disc at 6.0X. Compared with the 25GB discs, the recording speed for 100GB discs is almost constant. The result is shown in Figure 10. The drives have automatic speed deceleration during recording for fail-safe purposes. This fail-safe function is operated by detecting the disturbance of the servo signal of recording beam. If the disturbance is detected, drive will reduce the speed from 6.0X to 4.0X. Without disturbance, drive will restore the speed to 6.0X. The average recording speed is 5.9X. The recording time for single 100GB disc is 3757 seconds. When 12 drives burn together, all drives can reach the 6.0X speed simultaneously.



(a) Single drive recording process for 100GB disc



(b) Part of the recording process for 100GB disc

Figure 10. the single drive recording process for 100GB disc.

The read speed of optical disc are shown in Table 2. Each drive is working independently so the aggregated read speed of 12 drive is about 12X of single drive read speed.

5.5 Performance Impact of Mechanical Operations

In very rare cases when the requested files are located only on disc array in the rollers, mechanical operations must be invoked and they are in the critical I/O path, worsening the response time of external file read requests. Otherwise, the mechanical delay can be sufficiently hidden from the foreground applications. The average loading and unloading time of disc arrays are shown in Table 3. The roller rotation time is less than 2 seconds and takes up to 5 seconds to move the robotic arm vertically between bottom and top layer, separating 12 discs into 12 drives takes almost 61 seconds, fetching discs one by one from drives takes 74 seconds. Because the starting position of robotic arm is at the uppermost layer of the roller, it has to move down to bottom before loading or unloading a disc array in the lowest layer, so operation of the lowest layer takes about 5 more seconds than that of the uppermost layer.

The mechanical operations have very long delays that by themselves are difficult to be drastically reduced. The results clearly demonstrate how essential a tiered storage like ROS is to drastically reduce the performance impact of mechanical operations by fully hiding it to provide an illusion of inline accessibility.

6. Related work

There are several previous studies using disks, tapes and optical discs to build systems for long-term data storage.

Slot location	Disc array loading time(s)	Disc array unloading time(s)
Uppermost layer	68.7	81.7
Lowest layer	73.2	86.5

Table 3. Mechanical latency

IBM LTFS[20] provides a POSIX file system interface, thus makes linear tape an random access media. But LTFS is built on a single tape and its performance is limited by linear seek latency of the tape media. In contrast ROS provides a global file system view on all discs and uses tiered storage to hide the performance impact of the low disc throughput.

Facebook cold storage system[16] used Blu-ray disc. Its storage rack is packed with thousands of discs and provides a storage volume of petabytes. But Facebook has never provided any software and hardware design details in the public domain, making it impossible to evaluate and compare its performance.

Pelican cloud cold storage system[2] is designed to offer a thin-provisioned solution for cloud cold data. Its basic unit is rack too. In Pelican, all resources are carefully provisioned to reduce total cost of ownership with performance to support cold data workload. But its hard disk based design is not for long-term data preservation and still needs data migration towards the end of a hard disk's life cycle, which will bring extra complexity and cost. In addtion, foreground applications needs the dedicated interface to access Pelican.

SONY developed a massive media optical storage system with Blu-ray disc[25]. However it didn't provide a common file system interface thus requiring users to install extra software to manage the storage volume. Moreover, user needs to maintain namespace in each disc using file system specified software. This property makes it very difficult for the optical storage system to be transplanted in different file systems. ROS uses FUSE to trade performance for portability.

Panasonic Data Archiver[4] is a scalable optical disc library system. The basic storage module is a disc magazine with 12 discs. It uses 100GB optical discs so each magazine can store 1.2TB of data. In one 42U rack it can install 6500 disc which is half the capacity of our design. The library can be accessed by the Data Archiver Manager and CIFS network protocol. The basic unit has 12 drives and provides a 216MB/s data transfer rate. These optical libraries tend to be used like tape library, in contrast, ROS attempts to provide inline accessibility.

Optical storage arrays are introduced in [11]. This research only presents an assumption of using ultra high density optical discs to build a long-term storage system and discussed the advantage in theory without design details.

7. Conclusions

ROS is designed to store data in the long term by using optical discs. ROS provides a solution to hide the performance impact of individual optical disc's limitations on performance, capacity and specific operations. In this paper we have presented how the mechanical, hardware and software of ROS are co-designed to provide inline accessible data storage for long-term data preservation.

Acknowledgments

We sincerely thank EuroSys reviewers and Dr. Ant Rowstron for their constructive and insightful comments and advices that helped improve the paper. This work is supported in part by the National Basic Research Program of China (973 Program) under Grant No.2011CB302303, Wuhan National Laboratory for Optoelectronics Fund under Grant No.0106187015 and No.0106187027 , US NSF under Grant No.CCF-1629625 and the Amethystum Storage Technology Co., Ltd.

References

[1] O. S. T. Association. Universal disk format specification. `www.osta.org/specs/pdf/udf250.pdf`, 2003.

[2] S. Balakrishnan, R. Black, A. Donnelly, P. England, A. Glass, D. Harper, S. Legtchenko, A. Ogus, E. Peterson, and A. Rowstron. Pelican: A building block for exascale cold data storage. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 351–365, 2014.

[3] S. Boyd, A. Horvath, and D. Dornfeld. Life-cycle assessment of nand flash memory. *Semiconductor Manufacturing, IEEE Transactions on*, 24(1):117–124, 2011.

[4] P. Corp. Data archiver lb-dh8 series. Technical report, 2016.

[5] D. Crockford. Javascript object notation. `http://www.json.org/`, 2016.

[6] G. Deepika. Holographic versatile disc. In *Innovations in Emerging Technology (NCOIET), 2011 National Conference on*, pages 145–146. IEEE, 2011.

[7] G. R. Ganger and M. F. Kaashoek. Embedded inodes and explicit grouping: Exploiting disk bandwidth for small files. In *USENIX Annual Technical Conference*, pages 1–17, 1997.

[8] V. T. George Amvrosiadis. filebench. `https://github.com/filebench/filebench/wiki`, 2016.

[9] B. Godard, J. Schmidtke, J.-J. Cassiman, and S. Aymé. Data storage and dna banking for biomedical research: informed consent, confidentiality, quality issues, ownership, return of benefits. a professional perspective. *European Journal of Human Genetics*, 11:S88–S122, 2003.

[10] M. Grawinkel, L. Nagel, M. Mäsker, F. Padua, A. Brinkmann, and L. Sorth. Analysis of the ecmwf storage landscape. In *13th USENIX Conference on File and Storage Technologies (FAST 15)*, pages 15–27, 2015.

[11] M. Gu, X. Li, and Y. Cao. Optical storage arrays: a perspective for future big data storage. *Light: Science & Applications*, 3 (5):e177, 2014.

[12] P. Gupta, A. Wildani, E. L. Miller, D. S. H. Rosenthal, and D. D. E. Long. Effects of prolonged media usage and long-term planning on archival systems. In *International Conference on Massive Storage Systems and Technologies*, 2016.

[13] C. R. Hertel. *Implementing CIFS: The Common Internet File System*. Prentice Hall Professional, 2004.

[14] N. Kishore and S. Sharma. Secured data migration from enterprise to cloud storage–analytical survey. *BVICAM's International Journal of Information Technology*, 8(1), 2016.

[15] S. Kumar and T. R. McCaffrey. Engineering economics at a hard disk drive manufacturer. *Technovation*, 23(9):749–755, 2003.

[16] R. Miller. Inside facebook's blu-ray cold storage data center. `http://datacenterfrontier.com/inside-facebooks-blu-ray-cold-storage-data-center/`, 2014.

[17] H. Minemura, K. Watanabe, K. Adachi, and R. Tamura. High-speed write/read techniques for blu-ray write-once discs. *Japanese journal of applied physics*, 45(2S):1213, 2006.

[18] B. Nikoobakht and M. A. El-Sayed. Preparation and growth mechanism of gold nanorods (nrs) using seed-mediated growth method. *Chemistry of Materials*, 15(10):1957–1962, 2003.

[19] Y. Okazaki, K. Hara, T. Kawashima, A. Sato, and T. Hirano. Estimating the archival life of metal particulate tape. *Magnetics, IEEE Transactions on*, 28(5):2365–2367, 1992.

[20] D. Pease, A. Amir, L. V. Real, B. Biskeborn, M. Richmond, and A. Abe. The linear tape file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–8. IEEE, 2010.

[21] A. Rajgarhia and A. Gehani. Performance and extension of user space file systems. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 206–213. ACM, 2010.

[22] P. Rattan et al. Disaster management and electronic storage media: An overview. *International Journal of Information Dissemination and Technology*, 2(1):1, 2012.

[23] A. Rosenthal, P. Mork, M. H. Li, J. Stanford, D. Koester, and P. Reynolds. Cloud computing: a new business paradigm for biomedical information sharing. *Journal of biomedical informatics*, 43(2):342–353, 2010.

[24] Sony. Sony everspan. Technical report, 2016.

[25] C. Thompson. Optical disc system for long term archiving of multi-media content. In *Systems, Signals and Image Processing (IWSSIP), 2014 International Conference on*, pages 11–14. IEEE, 2014.

[26] W. Wamsteker, I. Skillen, J. Ponz, A. De La Fuente, M. Barylak, and I. Yurrita. Ines: astronomy data distribution for the future. *Astrophysics and Space Science*, 273(1-4):155–161, 2000.

[27] A. Watanabe. Optical library system for long-term preservation with extended error correction coding. In *29th IEEE conference on massive data storage*, 2013.

[28] N. Yamamoto, O. Tatebe, and S. Sekiguchi. Parallel and distributed astronomical data analysis on grid datafarm. In *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, pages 461–466. IEEE, 2004.

[29] P. Zijlstra, J. W. Chon, and M. Gu. Five-dimensional optical recording mediated by surface plasmons in gold nanorods. *Nature*, 459(7245):410–413, 2009.