Project Silica: Towards Sustainable Cloud Archival Storage in Glass

Patrick Anderson, Erika Blancada Aranas, Youssef Assaf, Raphael Behrendt, Richard Black, Marco Caballero, Pashmina Cameron, Burcu Canakci, Thales de Carvalho, Andromachi Chatzieleftheriou, James Clegg, Rebekah Storan Clarke, Daniel Cletheroe, Bridgette Cooper, Tim Deegan, Austin Donnelly, Rokas Drevinskas, Alexander Gaunt, Christos Gkantsidis, Ariel Gomez Diaz, Istvan Haller, Freddie Hong, Teodora Ilieva, Shashidhar Joshi, Russell Joyce, Mint Kunkel, David Lara, Sergey Legtchenko, Fanglin Linda Liu, Bruno Magalhaes, Alana Marzoev, Marvin McNett, Jayashree Mohan, Michael Myrah, Truong Nguyen, Sebastian Nowozin, Aaron Ogus, Hiske Overweg, Antony Rowstron, Maneesh Sah, Masaaki Sakakura, Peter Scholtz, Nina Schreiner, Omer Sella, Adam Smith, Ioan Stefanovici, David Sweeney, Benn Thomsen, Govert Verkes, Phil Wainman, Jonathan Westcott, Luke Weston, Charles Whittaker, Pablo Wilke Berenguer, Hugh Williams, Thomas Winkler, Stefan Winzeck*

Microsoft

Abstract

Sustainable and cost-effective long-term storage remains an unsolved problem. The most widely used storage technologies today are magnetic (hard disk drives and tape). They use media that degrades over time and has a limited lifetime, which leads to inefficient, wasteful, and costly solutions for long-lived data. This paper presents Silica: the first cloud storage system for archival data underpinned by quartz glass, an extremely resilient media that allows data to be left in situ indefinitely. The hardware and software of Silica have been co-designed and co-optimized from the media up to the service level with sustainability as a primary objective. The design follows a cloud-first, data-driven methodology underpinned by principles derived from analyzing the archival workload of a large public cloud service. Silica can support a wide range of archival storage workloads and ushers in a new era of sustainable, cost-effective storage.

CCS Concepts: • **Information systems** → **Cloud based storage**; **Information storage technologies**; • **General and reference** → *Measurement*; **Design**; • **Hardware** → **Emerging optical and photonic technologies**.

*All authors are listed in alphabetical order and contributed while working at Microsoft.

SOSP '23, October 23–26, 2023, Koblenz, Germany

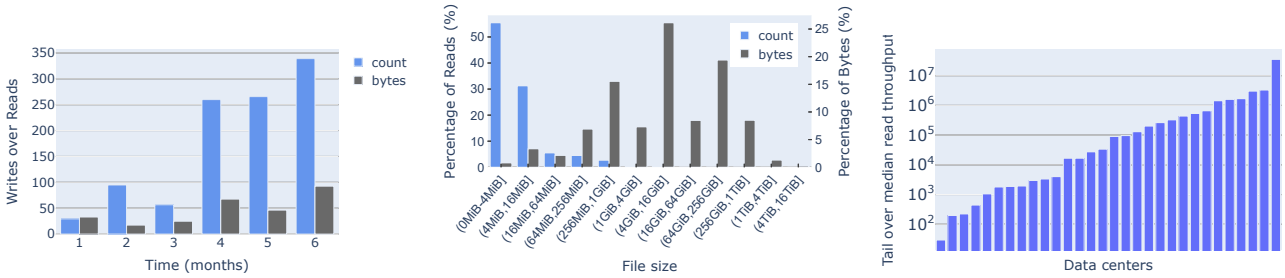
ACM ISBN 979-8-4007-0229-7/23/10.
https://doi.org/10.1145/3600006.3613208

Keywords: Storage, Cloud Storage, Cold Storage, Disaggregation, Data Center, Archival, Sustainability, Glass

1 Introduction

Demand for archival storage for high-value, long-lived data continues to grow exponentially [26]. Cloud archival storage providers exploit statistical multiplexing of many customers' workloads onto a shared storage platform to offer the lowest-cost storage. Today, in all large-scale cloud providers, the most cost-effective media is magnetic, usually combining hard disk drives (HDDs) and tape. *All* magnetic media has limited lifetimes, and the data on it degrades over time, so it has to to be migrated. In particular, HDDs have a ~5-year and tape a ~10-year lifetime. Since bit rot occurs in both over time, storage systems use scrubbing [5, 38, 39] to guarantee data integrity. Further, both have significant media costs, requiring garbage collection and defragmentation over time. As a result, while it would be intuitive to assume that data storage costs are a function of the customer access rates, for most archival workloads, the background management accesses associated with the *refresh cycle*, *data integrity checking*, and *garbage collection* dominate user accesses. Hence, they also dominate the costs associated with storing data. Ironically, since most archival data is rarely accessed, the environmental and financial costs of storing archival data on magnetic media *increase* over time.

Another common misconception is that archival storage is a *disaster recovery* workload, where reads are rare and access large volumes of data. Tape and tape libraries were designed to support this workload: a modern tape is over 1km long, spooling takes over a minute, and read drives provide high throughput (~360 MB/s). Tape library robots are prone to failures leading to media unavailability and are designed to perform tape load/unload operations assuming minutes of IO per tape. While tape offers the lowest $/GB equipment cost, all these challenges translate into runtime complexity. Section 2 shows



(a) Ratio of writes over reads per month.

(b) Percentages of read bytes (right y-axis) and count (left y-axis) vs. file size buckets.

(c) Tail over median read throughput across data centers.

Figure 1. Cloud archival workload characteristics.

that cloud archival workloads are different to the disaster recovery scenario: in a large cloud archival service, the workload is dominated by many small read IOs (80th percentile reads < 10MiB) and the majority of total read bytes (80th percentile) correspond to less than 2% of all read requests.

Silica is a first attempt to explore a clean-slate archival storage system, designed to service modern cloud archival workloads sustainably and efficiently. The media that Silica uses is quartz glass (fused silica). Using glass provides an extremely low-cost Write-Once-Read-Many (WORM) media with no bit rot over more than 1000 years. This latter property means that the media removes the need for integrity checking, minimizing energy usage over time. Since the media is low-cost, it also removes the need to perform active garbage collection. Finally, our design of the storage system around glass media removes the need for the refresh cycle, allowing data to be left in situ, and dramatically reducing the carbon footprint of long-lived data.

The hardware (media, drives, and library) for Silica has been *co-designed* along with a software stack to operate as a cloud-first storage platform. We employ *full disaggregation* [14] of both the hardware and the software: we design each part of the system to enable *elasticity* and maximize utilization without sacrificing maintanability and flexibility.

Each component of the software stack of Silica is designed as a user-level microservice that is fault-tolerant, can be scaled independently, and runs on the compute hardware that best matches it. The lack of dependence on kernel-level functionality also allows for greater flexibility in maintenance and evolution. These include the data preparation pipeline for writing (using CPUs with parallel vector instructions), the data decode pipeline for converting signal from the analog into the digital domain (using efficient ML inference hardware), as well as the control plane of the media library (using low-power CPU cores). Thus, the system is *resource-proportional*: the software stack *right-sizes* to the amount of customer load at any given time, and the hardware is designed to allow independent scaling of read throughput, write throughput, and robotics.

The Silica library does not use gantry robotics as a traditional tape library, but instead, uses a set of free roaming shuttles inspired by state-of-the-art warehouse robotic systems. Our read and write processes use different technologies; and to minimize stranded resources (and lower the storage costs), we use physically different read and write drives. This allows independent scaling of read and write throughput. Additionally, this design allows us to create the first storage system that offers true *air-gap-by-design*: the robotics are unable to insert a glass platter into a write drive once the glass media has been written.

We have learned that building a new storage technology from the media up consists of three stages: *(i)* understanding and controlling the physics; *(ii)* scaling mechanisms to operate at the desired scale and speed; and *(iii)* operationalizing the technology into a deployable form. We have a strong understanding and control of the physics. We also understand the scaling mechanisms, have prototyped read and write technologies that can achieve the required throughputs and density, and have prototyped a full-scale media library that provides long-term platter storage. This work spanned research and innovation in physics, machine learning, computer systems, and industrial design. We are actively working on the third stage, as the production hardware is not yet fully operational.

In this paper, we provide details about all parts of the system and focus particularly on the computer systems aspects. We describe the basics of reading and writing, the library, shuttle management, the software stack, how we tolerate failures, our data layout policies, and data pipelines for writing, reading, and processing in an elastic storage stack. To show that the system is able to service cloud archival workloads, we benchmark our large-scale prototype and present results from a full-system, discrete event simulator in Section 7.

2 Cloud Archival Workloads

This section examines workload characteristics for a set of geographically distributed tape libraries that implement part of a multi-tenant cloud archival storage service. The tape libraries need to support a service-level objective (SLO) on the order of 10+ hours. A *data center* contains one or more tape libraries, with a variable number of tapes, frames, and tape

drives. We log all I/O operations performed across all tape libraries, and filter to include I/O operations that are performed as a direct consequence of user requests. Here, we focus on the *read* and *write* operations over a period of six months in 2022, which represents tens of billions of operations.

The first characteristic we look at is the writes over reads per month in terms of data volume and number of operations, shown in Figure 1(a). This shows that, on average for every MB read there are 47 MBs written, and for every read operation there are 174 writes. We can see some variation across months, but writes always dominate by over an order of magnitude.

Next, we examine the read request sizes, where intuitively, one may expect the workload to be a small number of large reads. Figure 1(b) shows the percentage of volume of data read (grey, right y-axis) and the percentage of read operations (blue, left y-axis) vs. ranges of file sizes. Small files dominate the workload, with 58.7% of the reads for files of 4 MiB or smaller. However, these reads only contribute 1.2% of the volume of data read. Files larger than 256 MiB comprise around 85% of bytes read but less than 2% of total read requests. Additionally, there is a long tail of request sizes: there is ~ 10 orders of magnitude between the smallest and largest requested file sizes.

To understand how homogenous or heterogeneous the read workload is, we extracted the 30 most read-active data centers. For each, we calculate the the median and 99.9th percentile (tail) hourly read rates (in MB/s). Figure 1(c) shows the data centers ranked by 99.9th percentile rates, normalized to the median. We observe a variability in the workload within data centers, with up to 7 orders of magnitude difference between the median and the tail, as well as large variability across different data centers.

To understand the burstiness of the ingress rate, we measure the volume of data written per data center in different time windows, varying from 1 to 60 days. Within each rolling window, we calculate the average ingress rate based on the total volume of data written over that period. Figure 2 shows one representative data center, but the trend is quantitatively similar for all. We plot the peak over mean rate across all the rolling windows at different time aggregations. At the granularity of a day, the peak daily rate is ~16x higher than the mean daily rate. As the aggregation time increases beyond 30 days, the peak over mean ratio decreases significantly down to only ~2, indicating that the average write rate is similar across different 30-day windows. The write rate gets smoother as we increase the aggregation time, resulting in a much more predictable ingress rate over a longer window.

Summary and key system design observations. To summarize, as expected for archival storage, the workload is *heavily write-dominated*. However, unexpectedly, the IO operations are dominated by *small file accesses*. There is *variability* in the workload for both reads and writes, both within and across different data centers; and the read bandwidth needs to be provisioned to at least handle the burstiness. However, while writes are *bursty* at the granularity of a day or week and have a high peak to mean ratio up to ~16, across windows of a month, the write workload has a peak to mean ratio close to 2.

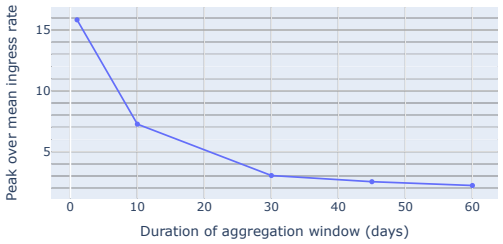


Figure 2. Peak over mean of average ingress rate across rolling windows as a function of the aggregation time.

These insights have big implications on the design of an archival storage tier. The library and read drives need to be able to handle a large volume of I/O requests for small files, so we need to minimize any mechanical overheads involved before the media is ready for reading. This includes the media transfer, mounting, and seeking delays. The average read is for less than 4 MiB, so raw per-drive read throughput for the customer load is less important. For the small fraction of large files, sharding across multiple platters can help. There is variability across data centers and it is important to enable read, write and library robotics to scale independently at different sites: a one-size-fits-all approach would lead to stranded capacity at most sites. Write bandwidth would normally be provisioned for peak, but with these access patterns, that would lead to low average utilization over time. In Silica, we smooth the write load over time with relatively small volume of staging prior to writing. This allows us to reduce costs by making the peak only a little higher than mean, so write utilization remains high.

3 Storing Data in Glass

We provide a high-level overview of how data is written and read from glass to understand how we exploit these properties in the design of Silica. Different technologies are used for writing and reading. Fused silica is the ideal media for archival storage because it has three suitable properties: *(i)* it is low-cost; *(ii)* it is chemically inert, so, durable and resilient to the environment (e.g. temperature, moisture, and electromagnetic interference); and *(iii)* the structures used to store data in it have lifetimes of over 1000 years [50].

The glass platter used to store data in Silica is a square that is approximately the size of a DVD. Unlike traditional optical discs, data is stored by making permanent physical modifications inside the pure glass platter (rather than thin film sandwiched in a pre-fabricated media) [51]. The modifications are not susceptible to bit rot or corruption, so no scrubbing is required. Therefore, fused silica is sustainable and provides a Write-Once-Read-Many (WORM) media. The platters are stored in the library and are not encased with any mechanics or other materials.

The permanent modifications change the physical structure of the glass [51]. They are created using femtosecond-scale ($\sim 10^{-15}$ seconds) high power pulses from an ultrashort pulse laser. We refer to these modifications as *voxel*s. To create a voxel, pulses are focused into the glass, and a single voxel can encode multiple bits (on the order of 3 or 4) by modulating the polarization of the laser beam and the pulse energy during voxel creation [23, 40, 41, 50, 51]. Voxels are written side-by-side in 2D *layer*s across the XY plane. As in [51] the 3D volume of the glass is used, and in the Z axis 100s of layers of these XY planes can be created. The refractive index of a voxel depends on the orientation of the light's electric field [45] and this is called *form birefringence*.

Writing an entire platter of glass happens in a single load into the write drive; we write from the deepest to the top layer in the platter. Prior to writing, user data is buffered in an online storage tier. Relevant coding is performed on the data (see Section 5). In contrast to existing optical storage technologies, there is no spinning media in Silica. This is mainly driven by the read requirements: handling and stabilizing circular spinning media prior to performing small IOs is complex and induces additional delays. The write drives can achieve sufficient write throughput with low enough power to make a production-scale write drive feasible and cost-efficient.

Overwrites are handled logically by versioning in metadata, as the media is WORM and low-cost. Similarly, deletes are handled by encryption key deletion [46] for the file and removing pointers to it from the metadata. If a platter no longer contains live data, it can be melted down and sustainably recycled as blank media.

Reading does not use a femtosecond laser; it uses polarization microscopy [42] to image the platter. A polarized light beam is focused on the 2D plane of voxels of interest inside the glass, and the resultant electric field is measured onto a camera sensor [51]. The read drive does not decode these images internally, but generates a sequence of images of the voxels. These are then processed for decoding (see Section 3.2). The power of light used during reading cannot affect the physical structure of written voxels so cannot corrupt, overwrite, or erase any data stored, even if buggy, faulty, or under malicious control.

A platter consists of *sectors*: a 2D group of voxels in an XY plane. A sector is dimensioned such that a read drive can image an entire sector in one image, is rectangular, and contains over 100,000 voxels (upwards of 100 kB of data). A 3D stack of sectors in the platter is called a *track*. Overall, we achieve high volumetric densities, and even with engineering overheads to support reading and decoding, we can still achieve multiple TBs of user data stored per platter.

When a read is performed, the required platter is mechanically loaded into the read drive and the platter *mounted*. The read drive *seeks* on the XY plane to locate the desired track, and reads *an entire track*'s sectors in a single fast scan in the Z direction. Remember that the workload is dominated by small reads; many read operations can be serviced by reading a single track. The read technology allows the throughput of the read drive to scale in multiples of 30 MB/s, and there is a cost-performance trade-off. It is possible to have read drives with different throughputs in the same library. In Section 7, we evaluate the impact on system performance of varying the throughput between 30 and 210 MB/s.

3.1 Data Verification

Data verification is a critical part of the Silica design. As different technologies are used to read and write, after a platter is written it must be fully read using the same technology that will be used to read it subsequently. This happens before a platter is stored in the library and any staged write data is deleted.

This design has an interesting consequence: during the period when user data is being written into the library, the workload is going to become *read*-dominated. Every byte written must be read to be verified, in addition to the user reads. The read bandwidth has to be provisioned for peak user read rate, however the read workloads are very bursty, so read drive utilization is extremely low on average. Thus, the verification workload simply utilizes what would otherwise be idle read drives. To enable high drive efficiency, two platters can be mounted simultaneously in a read drive: one undergoing verification, and one servicing a customer read. Customer traffic is prioritized over verification, with the read drive switching away when a platter is mounted for a customer read. As soon as the customer platter stops being accessed, the read drive has the ability to quickly switch to the other platter and continue verification. This is similar to avoiding head-of-line blocking of mice flows by elephant flows in networked systems.

3.2 Data Decode

All storage requires signal conversion from the analog to the digital domain during readout, followed by error correction to achieve a persistent and reliable storage media. In Silica, to decode the data stored in each sector, we process the images captured by the read drive using machine learning, as it provides us with more accurate signal conversion to the digital domain than traditional signal processing techniques. Machine learning models are able to better learn and account for any noise properties inherent in the end-to-end write and read processes including: inter-symbol interference between adjacent voxels in the glass, scattered light from neighbouring layers during readout, variability between optical components, and more. By contrast, traditional signal processing techinques require extensive understanding of all these characteristics and careful (often hand-crafted) processing to remove their effects. Designing and prototyping the hardware in-house also means we have essentially unlimited training data to further improve the accuracy of the machine learning decoder.

We pose the problem as a classification task: the network must classify every voxel into its most likely symbol value. For each sector, the network takes the set of images captured by the read drive as input, and outputs a 2D array of probability distributions over the encoded symbols for all voxels

in the sector, one distribution for each voxel. These distributions are subsequently used as inputs into the per-sector error-correction code (Section 5).

Our decode stack evolved over the years from using a simple VGG-style [43] network that decoded a single voxel at a time to a custom fully-convolutional U-Net [37] network that decodes an entire sector at a time. The decode stack uses a microservices architecture and is elastic in its resource usage. It supports SLOs ranging from seconds to hours, and exploits that to allow time-shifting of processing to periods of lowest compute costs. The ML model can also be updated as it evolves without the need for firmware updates to the read drives or the library. The processors used for inference can also be upgraded over time to minimize costs, as processor technologies evolve.

4 The Glass Library

The Silica library design is driven by four key objectives. Data stored on platters at rest should not consume any resources, keeping costs and energy requirements low. The library should be able to store platters for at least the lifetime of the data center without requiring large-scale transfer of platters between library generations. The read and write bandwidth should be provisioned independently and it should be possible to scale them on the order of months according to the workload demand. Failures in the library mechanics should minimize impact on unavailability and performance.

A *Silica library* is a sequence of contiguous write, read, and storage racks interconnected by a platter delivery system. Along all racks there are parallel horizontal rails that span the entire library. We refer to a side of the library (spanning all racks) as a *panel*. A set of *free roaming* robots called *shuttles* are used to move platters between locations. A shuttle is slightly taller than the vertical spacing between two rails and, in the default configuration, is attached to two rails. Each shuttle is untethered and battery-powered, and it can move across the entirety of the panel which maximizes the potential paths in the library that it can use. In the horizontal direction, shuttles move along rails. To move vertically between rail tracks, shuttles *crab*. When crabbing, the shuttle releases one rail, pivots around the other rail, and then grips onto the target rail. Crabbing allows shuttles to move between rails at any point without need for dedicated vertical rails. Shuttles have a mechanism for handling platters called a *picker*; a shuttle uses the picker to *pick* any platter from the shelf between its rails. The shuttle can then carry the platter to another location in the panel and *place* it at its destination (i.e., a read drive or storage slot).

From left to right, a default Silica library deployment has a write rack, then a read rack, and then sufficient storage racks to fit all the platters produced by the write drive over its lifetime. Finally, another read rack is placed at the end. The separation of read drives helps minimize the distance shuttles travel, and also enables availability if storage racks become partitioned due to shuttle failures. The read, storage, and write racks can be independently scaled if needed.

The storage rack is designed for cost-effective long-term storage and efficient automated platter handling. Platters are stored vertically in *slots* on a set of *shelves* similarly to books on a bookshelf. There is a shelf between each pair of contiguous rails in the panel, which enables direct access to all platters from the panel. The storage racks have no active mechanical or electrical components: combined with the properties of glass, they do not need to be powered or cooled, and platters are securely held using gravity without any active locking mechanism. Storage racks do not need replacement for the lifetime of the data center.

A read rack contains multiple read drives. Each read drive is independent and has slots into which platters are inserted and removed. The number of shuttles active on a panel is limited to twice the number of read drives in the panel. The write drive is full-rack-sized and writes multiple platters concurrently. Written platters are collected by shuttles and delivered to read drives for verification. The *air gap* property is enforced by guaranteeing that the mechanics cannot move a platter from the eject bay back to the write drive (i.e., a one-way system), and the empty platter supply is not reachable by the shuttles. Both read and write racks require cooling, power, and network connectivity. Visuals of a Silica library are available at aka.ms/Silica.

4.1 Controller

The library controller runs as a service that manages the resources of the library. At a high-level, it oversees writing platters, schedules verifications, monitors the battery level of shuttles, and assigns platters to slots in the storage racks. Importantly, the controller schedules and manages shuttles, which we focus on in this section. This is critical for the read operations, and the controller is designed to maximize the library aggregate read throughput while optimizing the utilization of the read drives and the shuttles. The controller has three logical tasks: *scheduling*, *traffic management* and *load balancing*.

The *scheduler* manages the request queue, while monitoring the expected tail latency for reading data (as opposed to mean). The scheduler maintains a queue ordered on request arrival time and maintains a separate structure that groups all requests for the same platter. By default, once a platter is inserted into a read drive all the requests for that platter are serviced since the fetch time dominates. Doing so amortizes a fetch across many reads when possible. We could optimize the read order to minimize seek latency, but seek latency is one of the lowest overheads in the system.

Platter fetch selection is based on work-conserving fairness. The platter selected has the earliest queued read *among the platters that are accessible*. Due to work preservation, the selected platter might not have the earliest queued request overall. For example, if the earliest queued request is for a platter currently obscured by a shuttle picking another platter, it

will not be serviced next. However, the scheduler guarantees that it will be executed as soon as the required resources are available.

Once a platter is selected, the *traffic manager* assigns a free shuttle to fetch it. Shuttles have full freedom of motion on a panel, so the design space of possible traffic management policies is very large. This is akin to the Multiple Agent Path Finding (MAPF) problem which is known to be NP-hard in the general case [49]. The traffic manager ensures shuttles motions do not cause conflicts on shared rails within the panel. Effectively it minimizes *congestion* which means a shuttle needs to slow down (or stop) to give way. The manager minimizes congestion as a two stage process: optimize paths globally for all shuttles, then dynamically mitigates localized congestion.

To reduce complexity and keep traffic management simple and predictable, the traffic manager splits the storage racks and read drives in the panel into n logically partitioned rectangular segments, where n is the number of active shuttles on the panel. Each logical partition must contain one shuttle and at least one read drive slot. Under normal operation, shuttles do not move outside of their logical partition, which eliminates congestion at the read drives. Congestion can occur at the boundaries between logical partitions and is resolved by a localized conflict resolution mechanism prioritizing the shuttle with the highest identifier. In Section 7, we show that this is rare and does not lead to imbalance across partitions. During normal operation, shuttles move along shortest paths within their partition. If a partition contains multiple read drives, we select the one that minimizes the time to mount the platter, which is the time it takes to move to the read drive and wait for it to become free.

Under some workloads the static partitions can lead to load imbalance, for example if a read drive is idle but there are no requests for platters in its logical partition. The traffic manager uses a work stealing scheme where shuttles from lightly loaded partitions can temporarily move outside of their assigned partition to pick platters from an overloaded partition. The controller monitors the amount of data to be read from each partition and triggers the work stealing scheme if the difference between the least loaded and the most loaded partition exceeds a defined threshold. Motion outside a partition is done using shortest path routing, but this may create additional congestion.

5 Error Correction

In this section, we outline two coding techniques used in Silica to implement a persistent and reliable storage layer: intra-sector error correction (LDPC), and inter-sector erasure coding using Network Coding (NC).

LDPC. There are two classes of error modes that can impact individual sectors: Write- and read-time errors. Write-time errors could occur if the laser energy is nonoptimal during writing (for example, if sufficiently-large particulates are present in the optical path during writing), resulting in voxels missing from the sector. These types of errors are expected to be rare, as data center environments are clean and active monitoring and control of the laser energy is expected. Read-time errors are typically non-deterministic and mostly the result of stochastic read sensor noise, as with most storage media. They result in a small number of random voxels in the sector being decoded incorrectly.

To protect against sector-level errors, we use low-density parity-check (LDPC) codes [30], a common class of codes used in other storage devices such as hard disk drives and SSDs [52]. We also employ per-sector checksums to verify that the result of the LDPC decode procedure is correct. We provision the amount of LDPC overhead per-sector empirically, based primarily on the expected error rate resulting from read-time errors. Write-time errors that result in the loss of more voxels in the sector than the LDPC code can cope with are handled using the following erasure coding schemes.

Erasure Coding. Failures that cannot be resolved using LDPC turn into a sector erasure. In Silica, we rely on network coding [22] to handle such cases.

We define a *network group* of sectors to be $I+R$ sectors composed of I information and R redundant sectors, such that any I sectors in the group can be used to construct any other sector in the group. Conversely, each group can tolerate up to R sector failures. Information sectors in a network group contain user data whereas redundant sectors are encoded as linear combinations of the information sectors. Due to the properties of the binomial distribution, the probability of being unable to recover a group falls rapidly with the size of the group ($I+R$).

We design our network coding scheme to provide both (i) fast recovery in the common case of independent sector failures, and (ii) durability through large erasure code groups for other cases. Note that glass media is truly WORM, and as it is freed from the constraint of expensive coding updates, Silica can use group sizes in the *tens of thousands* as opposed to the small groups in existing storage systems (e.g., 10+4).

Within-track NC. Recall that a track is the minimum read unit. To cope with independent sector failures within a single track, which we expect to be the common case of sector failures, we rely on *within-track network coding*. We partition sectors within a track to $I_t = O(100)$ information sectors and $R_t = O(10)$ redundancy sectors. From a track read, only I_t sectors need to be decoded successfully to recover it completely, providing durability against sector failures within a track at no additional read cost.

Large-group NC. *Correlated* sector failures within a single track that cannot be resolved using within-track coding can be handled using large coding groups across independent tracks in the same platter. For each platter containing user data, we group $I_l = O(100)$ *information tracks* in the platter and generate $R_l = O(10)$ large-group *redundancy tracks* per group. Each of these groups have thousands of redundant

sectors for tens of thousands of information sectors in the platter.

Cross-platter NC. The previous two levels of network coding protect against decode errors within a given platter. To handle *platter unavailability* due to shuttle or read drive failures, we use cross-platter network coding. We define a *platter-set* to have I_p+R_p platters composed of I_p *information platters* and R_p *redundant platters*. User data is stored in information platters, and redundancy platters are accessed only when a platter from their set is unavailable.

Within a platter set, one track from each platter is organized into a network group. That is, sectors of R_p tracks on the redundant platters are encoded for information sectors of I_p tracks on the information platters. Note that this is significantly stronger than simply grouping matching sectors in the platters; whilst at the same time, if a platter is unavailable, a read of a track is only inflated to the I_p matching tracks within the platter-set. The redundancy tracks of redundancy platters are additionally encoded to protect the information tracks in the information platters.

Durability. In summary, the immutability and durability of the media enables Silica to disaggregate the error handling, tune each technique, and combine them to deliver efficiency. Errors that impact voxels randomly are handled with LDPC and uncorrelated sector errors are handled with the in-track erasure code. Any residual correlated sector errors are handled with the very large network coding groups across many independent tracks within a platter and unavailability of platters is handled with the cross-platter erasure code.

Note that we verify every platter after writing, and so we know for every sector both whether it is recoverable, and the available LDPC margin. Together with the expected read error rate over time, we can determine whether to record a file as durably stored in the media. If a file cannot be recovered from a platter during verification, it can simply be kept in staging and rewritten onto a different platter later. Since the media is used as part of a cloud service as opposed to data or software distribution, the entire platter need not be rewritten due to rare write-side problems that affect a subset of its files.

When choosing the size of the network groups for each of these levels of network coding, there are trade-offs between recovery effort, probability of decode failure, and storage overhead. We discuss these trade-offs along with how data in a Silica library is laid out in Section 6.

6 Data Layout and Management

The data layout impacts both the performance and availability of the system in handling read requests. As described in Section 2, it is possible to smooth the write rate over 30 days or more. Files can be staged in the storage front-end before being written to platters; we can use this time to make decisions about data placement to maximize the expected read performance and availability. Within a deployment, we address data placement at four separate levels:

Assignment of files to platters. Like other storage systems, we want to pack files that we expect to read together to the same platter. This minimizes the costs of platter travel, load, and unload. We can use the (opaque) customer account identifiers, file write times, and historical access trends to make informed decisions on which files should be packed together. To ensure time-efficient read of large files, we shard them into multiple platters to parallelize their reads.

Placement of files within a platter. Given a set of files to place inside an information platter, we need to decide on which information sectors contain which file's data and which sectors contain redundant network coding data. To do so, we need to consider the ordering of files and the storage overhead of redundancy within the platter.

The minimum read unit is a single track and the read drive can read adjacent tracks in serpentine sector-order without an additional seek. Thus, we want to locate a file, and co-locate groups of files that are likely to be read together, within a single or adjacent tracks. Additionally, from a single track, we want to obtain both the requested data and enough redundancy to recover that data in the common case of independent sector failures. This is possible to achieve within a single track for the majority of read requests since they are for small files (Section 2).

Since an entire track is read at a time, we need to focus on the *probability of failure to decode a whole track*. Recall from Section 5 that each track forms a network sub-group of hundreds of sectors. This large group size provides excellent protection from independent sector failures, even if the probability of sector failure is high. With a redundancy overhead of $\sim 8\%$, and a sector (LDPC) failure probability of 10^{-3} (which is what we observe in our prototype), the probability of failure to decode a track is less than 10^{-24}. In addition, large-group NC across tracks provides protection against correlated sector failures within a track, with around $\sim 2\%$ additional overhead. For manageability, we assume that every information platter in Silica has the same partitioning of information and redundancy sectors, so, the structure of the network groups do not need storing for each platter and the within-platter redundancy overhead is the same for all information platters.

To conclude, given a set of files to place in the platter, we order them such that related files are next to each other. Starting from the first track, moving in serpentine sector-order, we place files into the information sectors while calculating and placing the associated redundant sectors. Since we do not investigate the optimal packing of files to tracks with respect to their sizes, sectors related to an individual file may be spread across one more track than the optimal. However, in that case, the extra track is adjacent so the read cost is minimal.

Partitioning platters to platter-sets. Given a set of information platters, we need to partition them to form platter-sets and to create redundancy platters. Similar to the assignment

of files to platters, we want to group information platters that contain files that are likely to be read together. Since many platters from the same platter-set need to be loaded and read in case of a shuttle or read drive failure, such partitioning allows us to streamline the travel and mechanical costs of recovery with those of a regular request.

Placement of platters within a deployment. We want to place platter-sets such that we maximize the read-availability of a deployment, as well as the utilization of its write drives, read drives, and shuttles. First, we assume that the deployment consists of a single library which is the minimum deployment unit (MDU).

As in Section 5, every platter-set consists of I information and R redundancy platters and increasing R increases a library's tolerance to shuttle and read drive failures. There are various failure cases for these components such as unresponsive shuttles or read drives. Additionally, although we do not expect shuttle-bound collisions due to our hardware and shuttle management measures, for robustness of placement, we consider even two-shuttle collision as a failure case. Note that the library controller can reliably detect shuttle or read drive failures.

Each failure case in our analysis has a corresponding *blast zone*, which is the area of the library that is inaccessible due to the failure, specified at the granularity of one shelf of one rack. When a failure occurs, any platter stored in the blast zone will be temporarily unavailable. In addition, zero to two platters may be inaccessible within the failed components, e.g., one within a read drive or one each within two collided shuttles. We want to minimize the number of platters from the same platter-set that may be unavailable due to a single failure, so, we place platters such that no two platters from the same platter set can be within a blast zone. Thus, a single failure can make at most three platters from the same platter-set unavailable. In Silica, we fix R to 3, so that a library can serve all read requests while a worst-case failure is being resolved. Note that the number of concurrent failures that can be tolerated could be improved using more complex traffic management policies that make sure a shuttle does not travel around platters from the same set as the one it carries, but this would come at the cost of flexibility and performance.

Given a fixed R, there are multiple considerations for deciding I, namely, redundancy overhead at the write drive, the size of the library, and recovery effort. Increasing I decreases the ratio of the redundant work done at the write drive, which we would like to minimize for cost-effective and sustainable writing. However, it can increase the minimum number of storage racks required in a library since each platter from the same set needs to be placed in a sufficiently separate area of the library. In addition, increasing I increases the number of platters that need to be loaded and read in case recovery is needed. However, as mentioned above, we can reduce some of the recovery cost by forming platter-sets carefully so that we take advantage of the temporal locality of the requests and the many-hour SLO.

Table 1. Write time redundancy overhead and minimum storage racks for different platter-set configurations.

I+R (platter sets)	Redundancy overhead at write drive	Storage racks needed
12+3	25 %	6
16+3	18.8 %	7
24+3	12.5 %	10

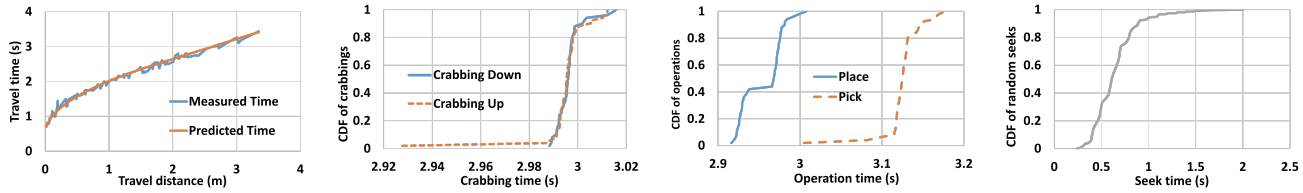
Table 1 summarizes the redundancy overhead at the write drive and the minimum number of storage racks needed in a library for various values of I. Note that based on our design, a library needs at least six storage racks. Given a number of platters, I, and R, we have a method of determining the required number of storage racks based on binary integer programming. For brevity, we omit this method and present the results here. For our MDU as described in Section 4, we choose platter-sets of 16 information platters and 3 redundancy platters.

When placing platters from the same platter-set in a multi-library deployment, we spread them out within and across libraries as much as possible, while maintaining the invariant that at most one of them is in any potential blast zone. Libraries are independent of each other and do not share drives or shuttles, so spreading platters from a platter-set across libraries increases the deployment's robustness against unforeseen failures. As an added advantage, because we assign files that we expect to read together to the same platter-set, spreading them across libraries leads to better load-balancing and higher utilization of libraries at read-time.

Note that the platter locations in Silica are fixed: after a read, a platter is replaced in its initial location. This reduces the complexity of the controller since the validity of the placement of a platter-set, or the optimality of the library-wide placement need not be examined after each platter access. The only exception to this rule is if a platter's initial location is unavailable after it is read due to a shuttle failure, in which case, the platter is temporarily stored in a different slot until the failure is resolved.

To sum up, given a platter-set with 16 information and 3 redundancy platters, we place each platter in the set such that no two platters from the set are in the same potential blast zone. While choosing a slot for the platter, we prioritize slots that are in areas of the deployment least occupied. If a number of tracks in a platter needs to be recovered due to its unavailability, the corresponding tracks need to be read from 16 other platters in that platter-set.

Metadata management. The naming and indexing of files in the Silica service is similar to Azure Cloud storage [10]. All mappings above such as within-library and within-platter addresses are stored as additional metadata per file in a separate, highly-available storage service, backed by warmer media such as HDDs. During a normal read operation, the metadata service is first accessed to locate the file in the Silica service.



(a) Horizontal motion (b) Vertical motion (crabbing) (c) Picking and placing (d) Random seeks

Figure 3. Benchmarks from the library prototype.

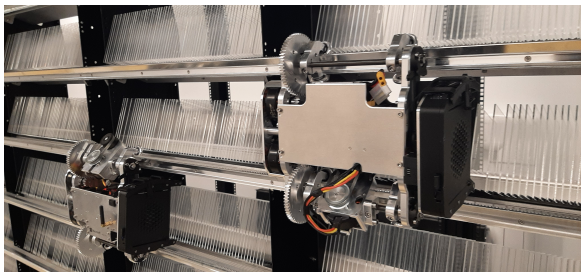


Figure 4. Shuttle prototype [1].

However, each platter is self-descriptive and its header contains the list of files on it. Therefore, a file can still be located within the service after a platter-level scan of libraries, should the metadata service be unavailable.

7 Evaluation

In this section, we provide an evaluation of the system design of Silica libraries. We simulate real cloud workloads using a full-system discrete event simulator, a digital twin of the library that is configured with the mechanical latency observations from our prototype. We aim to answer the following questions:

- What are the requirements from library components to sustain present cloud archival workloads?
- Can Silica efficiently and flexibly serve different profiles of workloads while keeping component utilization high?
- Can Silica libraries remain performant under future workloads or when requested platters are unavailable?

7.1 The Library Prototype

We start with a brief description of racks and shuttles in the prototype Silica library. Then, we measure the latency of the mechanical operations involved in serving read requests.

Racks and shuttles. We assume that storage racks have 10 shelves per panel and that a read rack can fit up to 10 drives and should have at least two drives for availability. In our experiments, we assume fully populated read racks. Shuttles are bound to their panel and can move and handle only one platter at a time. Figure 4 shows a photo of a Silica shuttle on a storage rack.

[1]Videos of the prototype library are available at aka.ms/Silica.

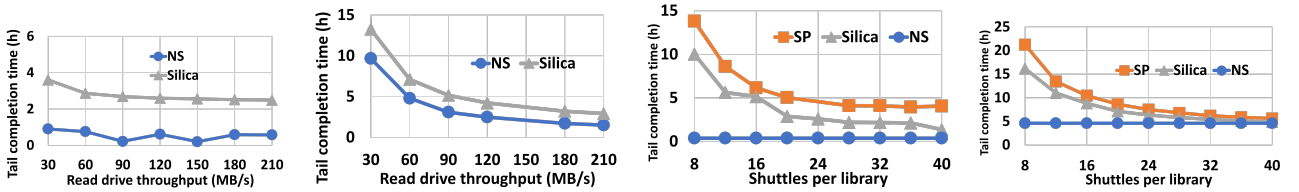
Benchmarking. We provide observed latencies from our prototype library for shuttle- and read drive-bound mechanical overheads. There are up to six mechanical operations involved in a read operation: horizontal and vertical shuttle motion, picking and placing of a platter, mounting a platter and seeking within a platter.

- The horizontal motion of the shuttle has two distinct phases. First, the shuttle performs a fast linear movement to the target location, fully defined by acceleration / deceleration and top speed. Then, fine tuning of the position is performed to align with the target, which always takes ~0.5 s. Our observed behavior from the prototype is shown in Figure 3(a) along with the predictions of the motion model in the digital twin.
- Figure 3(b) shows that vertical motion is predictable: the difference between the fastest and slowest crabbing operation is only 88 ms, and crabbing up and down have similar medians and tails, with 86% of operations completing in 3 s or less, with a maximum of 3.02 s.
- Each shuttle has a picker that can also carry one platter. Figure 3(c) shows the performance of pick and place operations. Picking is on average 170 ms slower that placing due to the weight of the platter.
- Our prototype read drives do not have a fully automated mounting mechanism; we assume a *conservative* constant overhead of 1 s for every mount or unmount operation, as well as for fast switching (see Section 3.1).
- We use our prototype read stage to benchmark seek delays. Figure 3(d) shows the distribution of random seek times with a median and maximum of 0.6 and 2 s respectively.

We configure the digital twin to sample mechanical operation durations from the abovementioned distributions.

7.2 Methodology

The primary metric we focus on is the **completion time** of a read request, which is the delay between the reception of a read request and the last byte read and sent from the library. We focus on 99.9th percentile, i.e., the *tail* completion time. We assume an SLO of 15 hours to the last byte, which is in line with current archival services [7, 31]. The completion time does not include the disaggregated decode, however, decode requests can be submitted with high priority to the ML stack for reads that complete close to the SLO.



(a) Per-drive throughput (IOPS) (b) Per-drive througput (Volume) (c) Increasing shuttles helps (IOPS) (d) Shuttles improve load balancing (Volume)

Figure 5. Performance of Silica with different read drive throughputs and different number of shuttles per library.

Workload. We evaluate Silica using different read traces from a representative data center in a large cloud archival service. We aim to show that Silica can serve as the backend to that service, which is currently backed by tape libraries.

To choose the read traces to simulate, we consider 12-hour rolling intervals across six months in the data center. We choose intervals with (i) the highest volume of data read (*Volume*), (ii) highest number of read requests (*IOPS*), and (iii) a *Typical* interval. To put the workloads into context: compared to Typical, IOPS has approximately 10x more reads per volume read, while Volume has a 25x higher volume read, but only 5x more reads by count. For each of these three 12-hour intervals, we create a workload trace which also includes previous (warm-up) and subsequent (cool-down) read requests and record statistics about the requests issued in the interval. Unless otherwise mentioned, we do not assume data locality, i.e., we distribute the read requests to platters stored in the library uniformly. However, we investigate the impact of skewed read request placement in Section 7.5. If a file consists of multiple tracks, we assume that the tracks are adjacent as described in Section 6.

We do not replay write requests since writes are buffered, disaggregated, and do not impact read completion times. We assume a platter to be verified is always mounted in the drive and measure the time each drive is able to spend on verification.

Provisioning. The ingress rate can be smoothed as described in Section 2 and serviced if the libraries have enough aggregate write throughput. Hence, we compute the ingress rate at trace time and use the rate to determine the number of libraries (MDUs) to provision.

For each trace, we compute the number of platters in the libraries based on the amount of data stored in the backend at trace time. This leads to data centers where Silica libraries are not full, which we expect to be the case in a library that is in early stages of its lifecycle. To evaluate performance of a full library, we use synthetic traces as described in Section 7.7.

Baselines for comparison. To understand the overhead of shuttle mechanics and management, we use a baseline called *No Shuttles (NS)*. NS can start loading the next platter into the read drive as soon as it becomes available. Although this system is infeasible since it assumes infinitely fast shuttles, it provides a proxy to the lower bound of the shuttle overhead.

To evaluate our shuttle management policy, we compare it with the strawman baseline called *Shortest Paths (SP)*. This baseline has no partitioning of panels and any shuttle can move anywhere in the panel using shortest path routing. Note that for all evaluated systems, we use the same workload, number of read drives and shuttles, and physical layout.

7.3 Exploring Component Requirements

The number of shuttles, as well as per-drive throughput play a key role in the performance of the library. The following section explores the impact of each of these parameters using the digital twin and workload traces.

7.3.1 Per-drive read throughput First, we want to investigate the necessary throughput per read drive to sustain the range of workloads we have. We fix the number of read drives in the library to 20 (the maximum possible in the MDU), and the number of shuttles in the library to 20 (one per read drive). We explore a range between 30 and 210 MB/s of throughput per drive, in increments of 30 as described in Section 3. We plot the tail completion time vs. per-drive read throughput for the IOPS and Volume workloads in Figure 5(a) and Figure 5(b) respectively. It might be expected that bandwidth-intensive (Volume) workloads require tape-scale (hundreds MB/s) read throughput per drive. However, we see that for both IOPS and Volume workloads, 30 MB/s read drives complete the requests within the SLO.

More specifically for the IOPS workload, high throughput read drives do not significantly reduce tail completion time. For example, the tail completion time for NS plateaus at about 15 minutes for 60 MB/s and above. For Silica, where shuttles add extra mechanical overhead, it is well under SLO at 30 MB/s per drive, just under 3 hours at 60 MB/s per drive, and plateaus at around 2.5 hours for higher throughputs. Although not plotted, the same trend holds for the Typical workload, with plateaus at 2.6 and 48 minutes for NS and Silica respectively.

For bandwidth-intensive workloads, the aggregate read throughput delivered by a library is critical, so we expect the throughput of the read drive to be more significant. For the Volume workload, we observe that the tail completion time is

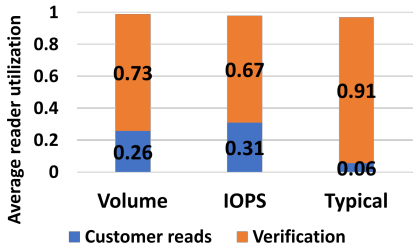


Figure 6. Read drive utilization, IOPS, 20 shuttles, 40 read drives.

dropping as per-drive throughput increases, though improvements beyond 60-120 MB/s tail off. This trend is consistent for both NS and Silica, showing that the ultimate bottleneck for this workload is neither the shuttle overhead nor the per-drive throughput, but the drive mechanics (loading, mounting and seeking) that reduce the *goodput* of the drives.

Summary. Silica library components are designed for the cloud with the goal of minimizing mechanical overheads, as informed by the bursty cloud workloads. Additionally, the glass media is random-access. Accordingly, we can see that Silica libraries can service present IOPS-intensive workloads within SLO even with 30 MB/s read drives. For bandwidth-intensive workloads, 60 MB/s read drives yield the best trade-off between per-drive throughput and tail completion time. In both cases, the throughput requirements from the read drives are way under the expectations from tape drives. For tractability, in the rest of the evaluation, unless mentioned otherwise, we assume 60 MB/s read drives, and 20 read drives per library.

7.3.2 Number of shuttles in a library

To speed up platter retrieval and delivery to read drives, we can increase the number of shuttles per library. In this experiment, we want to investigate the optimal number of shuttles for a variety of workloads. We fix the per-drive read throughput to 60 MB/s, the number of read drives to 20, and we vary the number of shuttles from 8 to 40 (two per read drive). We plot the tail completion time vs. number of shuttles for the IOPS and Volume workloads in Figure 5(c) and Figure 5(d). Note that NS has no shuttles, and so for each trace, its tail completion time is constant across the sweep.

For the IOPS workload, the tail completion time of NS is 22.2 minutes. For both SP and Silica, the large number of IOPS incurs additional shuttle overheads, but both systems remain within the SLO for all shuttle provisionings. Notably, shuttles are really instrumental for reducing the tail completion time for Silica, from 10 hours with 8 shuttles to 1 hour and 20 minutes with 40 shuttles, with diminishing returns from 20 shuttles onwards. For the Volume workload, the tail completion time of NS is 4.6 hours, a Silica deployment needs at least 12 shuttles to service requests within SLO, and the overhead from shuttles diminishes when 20 are provisioned per library.

Summary. Silica shuttles can efficiently cope with different types of workloads with completion times substantially below the SLO when enough are provisioned. For the IOPS-intensive workload, shuttles are critical in platter delivery, and thus increasing shuttles steadily reduces tail completion time for Silica. For the volume-intensive workload, although the important constraint is the aggregate read throughput, increasing the number of shuttles still leads to performance improvements through load balancing between read drives. In the following experiments, we assume 20 shuttles per library.

7.4 Read drive utilization

In Silica, read drives support two concurrent workloads: customer reads and verifications. We now evaluate the ability of fast switching (see Section 3.1), to avoid head of line blocking of customer reads while keeping drives at high utilization We measure *read drive utilization* which we define as the fraction of time a drive spends executing read requests or verifies, including mounting, unmounting and seeking, but excluding fast switching.

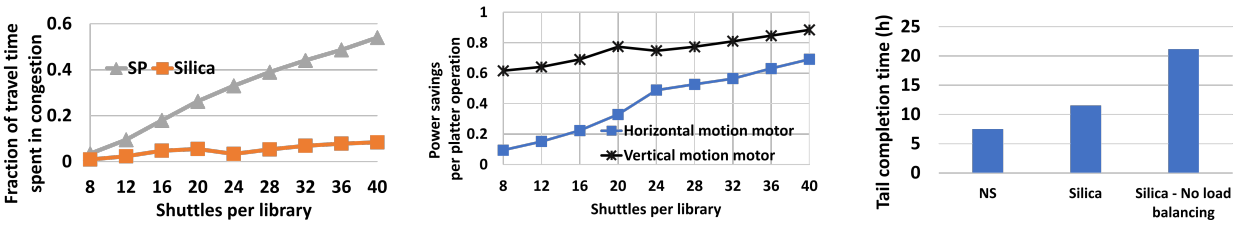
Figure 6 shows the average read drive utilization for Silica across all workloads. The average drive utilization is very high: above 96% for all workloads, showing that fast switching is an efficient mechanism for enabling the responsiveness of the system while keeping drives highly utilized. Read drives spend most of their time on verification for all workloads. Surprisingly, drives spend on average more time to service the IOPS workload than Volume: 31% vs. 26% respectively. This is due to more frequent mounting and unmounting of platters for IOPS. For the Typical workload, only 6% of the drive time is spent on reads vs. 91% for verifies.

Summary. Verifies must be preempted to service read requests in a timely manner. While such context-switching can normally reduce read drive utilization, fast switching allows read drives to remain at high utilization across different workload profiles. Moreover, read drives can efficiently serve verifies even during bursts, in addition to the high verify rate during typical periods.

7.5 Shuttle Management

Next, we focus on evaluating the shuttle management policies described in Section 4. Our goal is to show that the policies perform efficient platter retrieval, scale well with the number of shuttles, and are energy-efficient.

Partitioning. We now evaluate the impact of dividing the panel into logical partitions and restricting shuttle motion to remain within partitions (Section 4). For that, we compare Silica to SP which is the strawman policy that does not have partitioning. We use the IOPS workload, where shuttle movement is maximized. In Figure 5(c), we see that Silica enables substantially lower tail completion times than SP: 2.8 vs. 5 hours. We quantify the *congestion overhead* per travel as the time difference between (i) observed travel time and (ii) expected travel time in the absence of stopping due to obstructing shuttles. Figure 7(a) shows congestion overheads for SP and Silica as a function of the number of shuttles. For SP, any shuttle can move anywhere in the panel, which means



(a) Scalable congestion management (b) Sustainable routing (c) Skewed request distribution

Figure 7. Performance of shuttle management.

that increasing the number of shuttles leads to more conflicts and linearly increases congestion. In contrast, Silica is able to scale out, with congestion remaining within 10% for any number of shuttles. This is because under normal operation, shuttles remain within their partitions and do not conflict with each other.

An additional benefit of partitioning the shuttle management is the much lower power consumption per travel due to shorter travel distances and fewer acceleration / deceleration cycles. This impacts the duty cycle of battery-powered shuttles, as well as the sustainability of the library. We measure the average power consumed by the horizontal and vertical motion motors of the shuttle per platter operation. Figure 7(b) shows the power saving per platter operation of Silica over SP. The results show substantial energy-efficiency improvements for any number of shuttles, consuming on average 20%-90% less power per platter operation. Furthermore, energy savings improve as the number of shuttles in the library increase since each shuttle travels less on average. Contrarily, as the number of shuttles increase, SP is hit by congestion-induced stop/start overheads.

Load balancing. While partitioning the panels of the library for congestion avoidance has performance and sustainability benefits, if the read load across partitions is skewed, restricted flexibility can result in performance degradation. Therefore, an appropriate fallback from strict partitioning is necessary. Silica achieves this fallback by using the work-stealing scheme as described in Section 4.

We want to evaluate the impact of workloads where read requests are distributed skewed across partitions. We show results of the Volume workload, and we use a Zipfian distribution to assign requests to platters. This results in a highly-skewed request layout: i.e., the most accessed platter has an order of magnitude more data read than the second most accessed platter. Figure 7(c) summarizes our results. Without load balancing, Silica does not meet the SLO with a tail completion time of over 21 hours, because load is uneven across read drives. The results show the load balancing of work stealing, which has a tail completion time of 11.5 hours. This is achieved at the expense of extra travel delays as shuttles occasionally move outside their partitions. The tail travel time increases

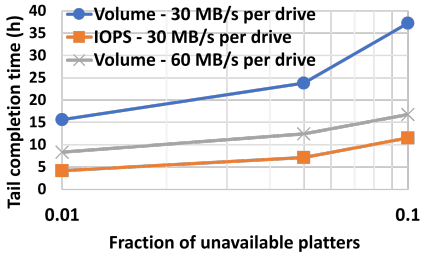


Figure 8. Impact of platter unavailability, 20 read drives, 20 shuttles.

from 29.4 seconds without load balancing to 76 seconds with load balancing. NS, which does not have any shuttle overhead and is able to flexibly load any platter into any read drive in the panel, has a tail completion time of 7.5 hours.

Summary. Silica provides flexible shuttle management that can cope well with different load distributions. In cases where requests are distributed balanced across a library or libraries, the partitioning scheme leads to efficient platter delivery. In addition, it supports our goal of sustainable design by reducing power waste. Where there is skew in the placement of requests, work stealing allows the shuttle management to adapt and service requests within the SLO.

7.6 Performance with unavailable platters

As described in Section 6, some platters in the library may be unavailable due to the failure of a shuttle or read drive. In that case, the cross-platter network coding is leveraged to retrieve the data. We use 20 shuttles and 20 read drives per library, we obtain results for different read drive throughputs for IOPS and Volume traces. We choose the unavailable platters uniformly randomly, similar to how we distribute files to platters.

Figure 8 shows the tail completion time for up to 10% of unavailable platters, which corresponds to several hundreds of platters, or multiple blast zones in an early library. The results for the IOPS workload indicate the efficiency of shuttle travel and management in the library: with 16x read amplification for 10% of the platters, Silica can service requests within SLO even with 30 MB/s readers. With bandwidth-intensive workloads such as Volume, read amplification is more pressing, since the aggregate throughput of the library is a bottleneck. We can see the benefit of higher throughput read drives here, going from

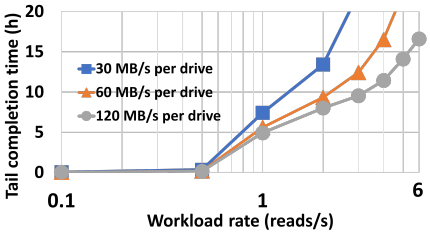


Figure 9. Performance of a full library.

30 MB/s to 60 MB/s drives reduces the tail completion time with 10% platter unavailability from 35 hours to ~15 hours.

Summary. For the IOPS-intensive workload, the Silica library can comfortably service requests within SLO even under 10% platter unavailability. With bandwidth-intensive workloads, the aggregate library throughput is critical: drive throughput of over 60 MB/s or more read racks per library would yield better performance under failures.

7.7 Full library

Thus far, we have focused our analysis on real archival data and workload traces, which lead to libraries that are not full. Nevertheless, we want to analyze performance in a fully populated library. To do so, we create synthetic traces based on a steady read rate and Poisson arrival. We assume each file is around 100 MB, which is the average file size obtained from our workload analysis. We focus on the completion time of requests that arrive in a 6-hour window (that is padded with warm-up and cool-down periods as before). We fix the number of read drives and shuttles in the library to 20 and distribute requests uniformly randomly across a full library. We obtain results for read drives with per-drive throughputs of 30, 60, and 120 MB/s (Figure 9).

To put the read rates into context: the mean read rate per Silica library in the early deployment that we simulate above is 0.3 reads/sec. Assuming a periodic deletion rate of 5% and a cool-down rate of 10%, we expect a mean rate of 1.6 reads/sec for a similar library 9-age-folds into the future. We see that 60 MB/s read drives can service such a workload with a tail completion time of around 8 hours.

Summary. A full library can service mean expected behaviour with similar read drives used to service workloads today. Should expectations change or for more challenging future workloads, the aggregate throughput and performance of a library can be improved by adding read racks to the end of the library as it ages.

8 Related Work

Technologies like DNA storage offer the promise of an extremely dense media for long-term data storage. However, the high costs and low throughputs of oligonucleotide synthesis and sequencing continue to hamper the technology's feasibility. The total amount of data demonstrably stored in DNA remains on the order of MBs [13], and building a functional storage system that can offer reasonable SLAs underpinned by DNA is a substantial challenge. Alternative DNA-based technologies like dNAM [20] attempt to bypass costly sequencing and synthesis steps, sacrificing density down to densities comparable with magnetic tape [20].

Optical archival storage have been proposed in the past using optical discs. ROS [48] is an optical disc library designed for data preservation. Facebook [33] also explored Blue-Ray discs for archival storage. These are based on single frames containing drives, mechanics and 1000s of optical disks. The key challenge for them is the optical disc capacity, today around 500 GB, which is significantly below tape per unit of volume. Glass can support very high densities [51] and even in early generations the density per mm^3 will be higher than production tape. Holographic storage [17] is not used as it too suffers from low volumetric densities.

Most cloud based archival storage services likely use HDDs in some role [6, 32, 36], but lowering the cost of storage is important. Systems like MAID [16] and Pelican [8] are HDD-based storage optimized for archive. They reduce the total cost of ownership but not to the level that can compete with tape.

Control of robotic libraries is well understood, but using free roaming robotics in a shared environment has many trade-offs[44]. A planning algorithm that provides completeness and path optimality [29] for real-time usage is hard and requires trade-offs [9, 19, 34]. Learning based approaches can lack predictability[15]. We have prioritized real-time operation and used logical partitioning in Silica to make this feasible. We plan to use the current platform to explore more sophisticated path-finding and scheduling algorithms that may be better.

9 Discussion

Cost and sustainability are of paramount concern in the cloud. Today, the only cost-competitive option for storing archival data is magnetic tape, as it offers the lowest $/TB cost. The unique properties of the glass media and the clean-slate, cloud-first co-design of the hardware and software allow Silica to be fundamentally more sustainable and achieve significantly lower costs for archival data than magnetic tape. Table 2 compares different aspects of magnetic tape and Silica technologies on cost.

Glass is the ideal media for archival storage. It is low-cost and readily available to manufacture at scale due to its use of sand as the raw material. In contrast, while magnetic tape is a more sustainable media than HDDs [47], manufacturing magnetic tape still requires significant amounts of energy and water, and results in higher cost, greenhouse gas, and air pollutant emissions [2, 3, 27], compared to fused silica glass [1][2]. Operationally, Silica also has fundamental advantages over magnetic tape. The durability and resilience of glass media means that Silica libraries can operate in a standard data center environment and do not spend any energy on scrubbing

[2]We expect a public report on this in due course.

Technology		Tape	Silica
Media Manufacturing	Financial cost	H	L
	Environmental impact	H	L
Media Maintenance	Scrubbing	M	L
	DC Environmentals	H	L
Drive Operations	Read process	M	L
	Write process	M	H
	Processing compute	M	L

Table 2. Cost comparison between different aspects of magnetic tape and Silica technologies. L: Low, M: Medium, H: High.

any data. In contrast, magnetic tapes are notorious for being susceptible to any changes in environmental conditions such as humidity, temperature, and dust. As a result, tape libraries need to operate in tightly-controlled environments separate from the rest of the data center, increasing their deployment and maintenance costs [4, 11, 25, 35]. Another consequence is that tape libraries need to scrub their media over its entire lifetime, further adding to environmental and financial costs [11].

The Silica read drives use polarization microscopy, which is a commoditized technique widely used in many applications [12, 18, 24, 28] and is low-cost. Currently, system cost in Silica is dominated by the write drives, as they use femtosecond lasers which are currently expensive and used in niche applications. This highlights the importance of resource proportionality in the system, as write drive utilization needs to be maximized in order to minimize costs. As the Silica technology proliferates, it will drive up the demand for femtosecond lasers, commoditizing the technology. As tape technology continues to mature [14], magnetic write and read processes will require continued investment and significant complexity to improve, requiring advancements in both head and media technology [21].

9.1 Future Work

Our current design uses known techniques for request scheduling and library traffic management. While our evaluation shows that these techniques already perform well, more sophisticated algorithms that better exploit the properties of the system might yield further performance and robustness benefits. In particular, learning-based approaches for shuttle scheduling could yield improvements to request completion times and reduce the number of shuttles required in a library. Similarly, our data layout techniques, namely the placement and assignment of files to platters and of platters to physical locations, could be improved using further knowledge of the workload.

The Silica system is air-gap-by-design: once a platter is written it is no longer accessible by a write drive, and read drives cannot modify the platter, leading to a physically immutable storage medium. There are several systems and security opportunities that such a medium presents. For instance, glass media provides a natural fit for append-only data structures such as blockchains. Additionally, there may be system-level benefits from evolving data archive design for a media that is naturally log-structured. Finally, the durability and immutability offered by the technology ensure and protect the integrity of data at a physical level, as opposed to software-protected integrity through, e.g., checksums. This stronger integrity property enables exploration of new security guarantees for data-at-rest.

Beyond traditional systems concerns, there is future work for the broader community and data preservation ecosystem. Engaging with many organizations with a vested interest in data preservation has highlighted the importance of considering the end-to-end data preservation workflow beyond the storage system itself. Once media and data durability at the byte level is a solved problem, there are concerns around file format and application-level obsolescence that need to be addressed. This broader consideration of how human knowledge is preserved presents opportunities for cross-disciplinary research and includes the ways in which users interact with a data preservation workflow (and indeed, what such a workflow even looks like), but also how data archives are curated and maintained, in order to maximize their utility to future generations.

10 Conclusion

Silica has been designed from the glass media up to support cloud archival storage. It uses design principles derived from understanding the archival workload in a large public cloud with the hardware and software co-designed to achieve them. In this paper, we focus on the computer system aspects of Silica and evaluate it using a prototype library and a full-system discrete event simulator. We show that Silica libraries can efficiently and sustainably support the cloud for long-term storage.

Acknowledgements

We would like to thank the SOSP reviewers and our shepherd, Simon Peter, for all their valuable feedback and suggestions. We'd also like to thank the following people for their contributions to Silica over the years: Janhavi Agrawal, Ibrahim Ahmed, Ben Arslan, Sarah Benaissa, Stefano Bucciarelli, Marco Faltelli, Zhonghe Feng, Michaela Florinda Picardi, Raluca Georgescu, Mark Green, Philip Heard, Katja Hoffmann, Marie Hoffmann, Krzysztof Jozwik, Joe Jun, Kirill Kalinin, Pavlo Kliuiev, Vamshidhar Kommineni, Qinbiao Li, Shirley Liu, Camilla Longden, Indeera Munasinghe, Douglas Phillips, Jorge Rodriguez Armas, Ashok Thillaisundaram, Nick Trim, Liana Valdes, Phillip Wright, Kaiuye Wu, Gyanendra Yadav, Francis Yu, and Siana Shekova.

References

[1] 2023. Life Cycle Assessment of Different Storage Media. Report prepared for Microsoft by WSP Consulting.

[2] U.S. Environmental Protection Agency. 1990. Magnetic Tape Manufacturing. In *AP 42, Fifth Edition, Volume I*. 4.2.2.13.

[3] U.S. Environmental Protection Agency. 2006. Magnetic Tape Manufacturing Operations: National Emission Standards for Hazardous Air Pollutants (NESHAP). https://www.epa.gov/stationary-sources-air-pollution/magnetic-tape-manufacturing-operations-national-emission-standards

[4] National Digital Stewardship Alliance. [n. d.]. Checking Your Digital Content. http://hdl.loc.gov/loc.gdc/lcpub.2013655117.1

[5] George Amvrosiadis, Alina Oprea, and Bianca Schroeder. 2012. Practical scrubbing: Getting to the bad sector at the right time. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*. 1–12. https://doi.org/10.1109/DSN.2012.6263919

[6] AWS. 2023. Amazon S3 Glacier storage classes. https://docs.aws.amazon.com/AmazonS3/latest/userguide/storage-class-intro.html

[7] AWS. 2023. What is Amazon S3 Glacier? https://docs.aws.amazon.com/glacier/index.html

[8] Shobana Balakrishnan, Richard Black, Austin Donnelly, Paul England, Adam Glass, Dave Harper, Sergey Legtchenko, Aaron Ogus, Eric Peterson, and Antony Rowstron. 2014. Pelican: A Building Block for Exascale Cold Data Storage. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. USENIX Association, Broomfield, CO, 351–365. https://www.usenix.org/conference/osdi14/technical-sessions/presentation/balakrishnan

[9] Zahy Bnaya and Ariel Felner. 2014. Conflict-Oriented Windowed Hierarchical Cooperative A*. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Robotics and Automation Society, 3743–3748. https://doi.org/10.1109/ICRA.2014.6907401

[10] Brad Calder, Ju Wang, Aaron Ogus, Niranjan Nilakantan, Arild Skjolsvold, Sam McKelvie, Yikang Xu, Shashwat Srivastav, Jiesheng Wu, Huseyin Simitci, Jaidev Haridas, Chakravarthy Uddaraju, Hemal Khatri, Andrew Edwards, Vaman Bedekar, Shane Mainali, Rafay Abbasi, Arpit Agarwal, Mian Fahim ul Haq, Muhammad Ikram ul Haq, Deepali Bhardwaj, Sowmya Dayanand, Anitha Adusumilli, Marvin McNett, Sriram Sankaran, Kavitha Manivannan, and Leonidas Rigas. 2011. Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles (SOSP '11)*. Association for Computing Machinery, New York, NY, USA, 143–157. https://doi.org/10.1145/2043556.2043571

[11] Germ Cancio, Vladim Bahyl, Daniele Francesco Kruse, Julien Leduc, Eric Cano, and Steven Murray. 2015. Experiences and challenges running CERN's high capacity tape archive. *Journal of Physics: Conference Series*. 4 (2015).

[12] Robert Allen Carlton. 2011. *Polarized Light Microscopy*. Springer New York, New York, NY, 7–64. https://doi.org/10.1007/978-1-4419-8831-7_2

[13] Luis Ceze, Jeff Nivala, and Karin Strauss. 2019. Molecular Digital Data Storage using DNA. *Nature Reviews Genetics* (May 2019). https://www.microsoft.com/en-us/research/publication/molecular-digital-data-storage-using-dna/

[14] Andromachi Chatzieleftheriou, Ioan Stefanovici, Dushyanth Narayanan, Benn Thomsen, and Antony Rowstron. 2020. Could cloud storage be disrupted in the next decade?. In *12th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 20)*. USENIX Association. https://www.usenix.org/conference/hotstorage20/presentation/chatzieleftheriou

[15] Lin Chen, Yaonan Wang, Yang Mo, Zhiqiang Miao, Hesheng Wang, Mingtao Feng, and Sifei Wang. 2023. Multiagent Path Finding Using Deep Reinforcement Learning Coupled With Hot Supervision Contrastive Loss. *IEEE Transactions on Industrial Electronics* 70, 7 (2023), 7032–7040. https://doi.org/10.1109/TIE.2022.3206745

[16] D. Colarelli and D. Grunwald. 2002. Massive Arrays of Idle Disks For Storage Archives. In *SC '02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*. 47–47. https://doi.org/10.1109/SC.2002.10058

[17] Hans J Coufal, Demetri Psaltis, and Glenn T Sincerbox (Eds.). 2000. *Holographic Data Storage*. Springer Series in Optical Sciences, Vol. 76. Springer. https://doi.org/10.1007/978-3-540-47864-5

[18] Michael W. Davidson and Gary E. Lofgren. 1991. Photomicrography in the Geological Sciences. *Journal of Geological Education* 39, 5 (1991), 403–418. https://doi.org/10.5408/0022-1368-39.5.403

[19] Boris de Wilde, Adriaan W. ter Mors, and Cees Witteveen. 2013. Push and Rotate: Cooperative Multi-Agent Path Planning. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '13)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 87–94.

[20] George Dickinson, Golam Mortuza, William Clay, Luca Piantanida, Christopher Green, Chad Watson, Eric Hayden, Tim Andersen, Wan Kuang, Elton Graugnard, and William Hughes. 2021. An alternative approach to nucleic acid memory. *Nature Communications* 12 (04 2021). https://doi.org/10.1038/s41467-021-22277-y

[21] Simeon Furrer, Mark A. Lantz, Peter Reininger, Angeliki Pantazi, Hugo E. Rothuizen, Roy D. Cideciyan, Giovanni Cherubini, Walter Haeberle, Evangelos Eleftheriou, Junichi Tachibana, Noboru Sekiguchi, Takashi Aizawa, Tetsuo Endo, Tomoe Ozaki, Teruo Sai, Ryoichi Hiratsuka, Satoshi Mitamura, and Atsushi Yamaguchi. 2018. 201 Gb/in^2 Recording Areal Density on Sputtered Magnetic Tape. *IEEE Transactions on Magnetics* 54, 2 (Feb. 2018), 1–8. http://ieeexplore.ieee.org/document/7984852/

[22] Christos Gkantsidis and Pablo Rodriguez Rodriguez. 2005. Network coding for large scale content distribution. In *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM, Vol. 4)*. IEEE, 2235–2245. https://doi.org/10.1109/INFCOM.2005.1498511

[23] E. N. Glezer, M. Milosavljevic, L. Huang, R. J. Finlay, T.-H. Her, J. P. Callan, and E. Mazur. 1996. Three-dimensional optical storage inside transparent materials. *Opt. Lett.* 21, 24 (Dec 1996), 2023–2025. https://doi.org/10.1364/OL.21.002023

[24] Chao He, Honghui He, Jintao Chang, Binguo Chen, Hui Ma, and Martin Booth. 2021. Polarisation optics for biomedical and clinical applications: a review. *Light: Science & Applications* 10 (09 2021), 194. https://doi.org/10.1038/s41377-021-00639-x

[25] IBM. 2023. TS4500 Tape Libray Documentation. https://www.ibm.com/docs/en/STQRQ9_1.9.1/pdf/ts4500-tape-library-1.9.1-documentation.pdf

[26] IDC. 2018. The Digitization of the World From Edge to Core. https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf

[27] RTI International. 2005. *Hazardous Air Pollutant Emissions From Magnetic Tape Manufacturing Operations – Background Information for Technology and Residual Risk Review*. Technical Report. https://nepis.epa.gov/Exe/ZyPURL.cgi?Dockey=9100923X.TXT

[28] Adib Keikhosravi, Michael Shribak, Matthew Conklin, Yuming Liu, Bin Li, Agnes Loeffler, Richard Levenson, and Kevin Eliceiri. 2021. Real-Time Polarization Microscopy of Fibrillar Collagen in Histopathology. *Scientific Reports* (05 2021). https://doi.org/10.21203/rs.3.rs-558678/v1

[29] Edward Lam, Pierre Le Bodic, Daniel Harabor, and Peter J. Stuckey. 2022. Branch-and-cut-and-price for multi-agent path finding. *Computers & Operations Research* 144 (2022), 105809. https://doi.org/10.1016/j.cor.2022.105809

[30] David J. C. MacKay. 2003. *Information Theory, Inference, and Learning Algorithms*. Copyright Cambridge University Press.

[31] Microsoft. 2023. Azure Archive Storage. https://learn.microsoft.com/en-us/azure/storage/blobs/access-tiers-overview

[32] Microsoft. 2023. Azure Archive Storage. https://azure.microsoft.com/en-us/products/storage/

[33] Rich Miller. 2015. Inside Facebook's Blu-Ray Cold Storage Data Center. https://www.datacenterfrontier.com/cloud/article/11431537/inside-facebook8217s-blu-ray-cold-storage-data-centerl

[34] Keisuke Okumura and Sébastien Tixeuil. 2022. Fault-Tolerant Offline Multi-Agent Path Planning.

[35] The Council on Library and Information Resources. [n. d.]. Magnetic Tape Storage and Handling: A Guide for Libraries and Archives. https://www.clir.org/wp-content/uploads/sites/6/pub54.pdf

[36] Google Cloud Plattform. 2023. Storage classes. https://cloud.google.com/storage/docs/storage-classes#archive

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (Eds.). Springer International Publishing, Cham, 234–241.

[38] Bianca Schroeder, Sotirios Damouras, and Phillipa Gill. 2010. Understanding Latent Sector Errors and How to Protect against Them. *ACM Trans. Storage* 6, 3 (sep 2010). https://doi.org/10.1145/1837915.1837917

[39] T.J.E. Schwarz, Qin Xin, E.L. Miller, D.D.E. Long, A. Hospodor, and S. Ng. 2004. Disk scrubbing in large archival storage systems. In *The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004. (MASCOTS 2004). Proceedings.* 409–418. https://doi.org/10.1109/MASCOT.2004.1348296

[40] Yasuhiko Shimotsuma, Masaaki Sakakura, Peter Kazansky, Martynas Beresna, Jianrong Qiu, Jiarong Qiu, Kiyotaka Miura, and Kazuyuki Hirao. 2010. Ultrafast Manipulation of Self-Assembled Form Birefringence in Glass. *Advanced materials (Deerfield Beach, Fla.)* 22 (09 2010), 4039–43. https://doi.org/10.1002/adma.201000921

[41] Manabu Shiozawa, Takao Watanabe, Eriko Tatsu, Mariko Umeda, Toshiyuki Mine, Yasuhiko Shimotsuma, Masaaki Sakakura, Miki Nakabayashi, Kiyotaka Miura, and Koichi Watanabe. 2013. Simultaneous Multi-Bit Recording in Fused Silica for Permanent Storage. *Japanese Journal of Applied Physics* 52, 9S2 (sep 2013), 09LA01. https://doi.org/10.7567/JJAP.52.09LA01

[42] M. Shribak and R Oldenbourg. 2003. Techniques for fast and sensitive measurements of two-dimensional birefringence distributions. *Appl. Opt.* 42, 16 (June 2003), 3009–3017.

[43] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014). http://arxiv.org/abs/1409.1556

[44] Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Roman Barták, and Eli Boyarski. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *Twelfth Annual Symposium on Combinatorial Search*. Association for the Advancement of Artificial Intelligence, 151–158. https://www.aaai.org/ocs/index.php/SOCS/SOCS19/paper/view/18341

[45] Wikipedia. 2023. Birefringence. https://en.wikipedia.org/wiki/Birefringence

[46] Wikipedia. 2023. Crypto-shredding. https://en.wikipedia.org/wiki/Crypto-shredding

[47] Velvet Wu. 2023. Tape Storage Might Be Computing's Climate Savior. https://spectrum.ieee.org/tape-storage-sustainable-option

[48] Wenrui Yan, Jie Yao, Qiang Cao, Changsheng Xie, and Hong Jiang. 2017. ROS: A Rack-Based Optical Storage System with Inline Accessibility for Long-Term Data Preservation. In *Proceedings of the Twelfth European Conference on Computer Systems (EuroSys '17)*. Association for Computing Machinery, New York, NY, USA, 161–174. https://doi.org/10.1145/3064176.3064207

[49] Jingjin Yu and Steven M. LaValle. 2013. Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI'13)*. AAAI Press, 1443–1449.

[50] Jingyu Zhang, Mindaugas Gecevičius, Martynas Beresna, and Peter G. Kazansky. 2014. Seemingly Unlimited Lifetime Data Storage in Nanostructured Glass. *Phys. Rev. Lett.* 112 (Jan 2014), 033901. https://doi.org/10.1103/PhysRevLett.112.033901

[51] J. Zhang, A. Čerkauskaitė, R. Drevinskas, A. Patel, M. Beresna, and P. G. Kazansky. 2016. Eternal 5D data storage by ultrafast laser writing in glass. In *Laser-based Micro- and Nanoprocessing X*, Udo Klotzbach, Kunihiko Washio, and Craig B. Arnold (Eds.), Vol. 9736. International Society for Optics and Photonics, SPIE, 97360U. https://doi.org/10.1117/12.2220600

[52] Kai Zhao, Wenzhe Zhao, Hongbin Sun, Xiaodong Zhang, Nanning Zheng, and Tong Zhang. 2013. LDPC-in-SSD: Making Advanced Error Correction Codes Work Effectively in Solid State Drives. In *11th USENIX Conference on File and Storage Technologies (FAST 13)*. USENIX Association, San Jose, CA, 243–256. https://www.usenix.org/conference/fast13/technical-sessions/presentation/zhao