International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Preallocated Media Storage in Large-Scale Archival Systems: Benefits, Misconceptions, and the Hidden Cost of Unreleased Space

Bhanuprakash Naidu Basani

Abstract: A media archive pipeline may simultaneously handle sustained write throughput aspects such as video ingest, image payloads with on-disk indexing, and lower-latency read paths to application retrievals. Rewind-free appends can be made more performant for the drive by preallocating storage blocks before the file grows, a common best practice that also reduces the odds of later on not being able to append due to lack of free space on the segment. Even so, preallocation is usually based on false assumptions, most notoriously of how far it can guarantee physical contiguity and thus superior sequential read performance. Nonetheless, modern distributed object-based storage systems suffer high metadata overhead and placement complexity where preallocation interacts with the allocation topology in ways that cannot be reduced to simple contiguity guarantees [1]. Hence, in a cost-conscious tiered storage hierarchy, from hot SSD to cold HDD, unreclaimed preallocated space leads to silent but wasteful consumption of the premium tier. Thus, a preallocation lifecycle that combines early reservation and deterministic reclamation at finalization is the only approach that provides a balanced trade-off between reliability and structural efficiency. The measurements show that reclamation can eliminate SSD tier pressure, recover hot-index locality, reduce the reserved-but-unwritten footprint by the majority, and lower preview retrieval tail latency by one to two orders of magnitude on a normalized basis. The results also indicate that preallocation without reclamation is not a complete solution and can degrade storage efficiency in a tiered storage environment at scale.

Keywords: File Preallocation, Tiered Storage Efficiency, Extent-Based Allocation, Reserved-But-Unwritten Capacity, Archival Ingest Pipelines

1. Introduction

Media archival systems that are expected to store continuously streamed video and still images, produce preview JPEGs, and build their metadata on the fly typically satisfy the write-path reliability constraint, meaning that segment writes to the system complete deterministically without late failure from lack of free space during an append. The second requirement on these systems is the read-path responsiveness constraint: all operations visible to the users, such as preview retrieval, browsing within a clip, or seeking metadata queries, should show low latency, even while ingestion is happening and for long retention horizons. File preallocation, in which reserve disk blocks are allocated before a file enters its active growing phase, satisfies this requirement by committing to a given bound on the size of the file, which reduces the probability that the in-flight write sequence will encounter an out-of-space condition. .

Large-scale distributed object-based storage systems face several placement, replication, and metadata-management challenges. The metadata structures typically managed by distributed object stores scale with the number of objects in the store, and the decisions made by the system when creating an object influence the free-space topology both at the object and system level [1]. In environments with millions of media objects of varying duration and bitrate, such a decision has fleet-wide implications for structural properties that cannot be reversed without a non-trivial service outage.

Independent Researcher, USA

Improper allocation lifecycle management across tiers incurs extra costs. By isolating hot data on a high-performance SSD tier while spinning down other assets on a low-cost ultra-high-capacity HDD tier, operators gain a 40-60% reduction in storage costs while maintaining acceptable retrieval times for high-frequency use cases in AI/ML and media workloads [1]. However, the above cost model assumes that each tier is always filled with data equal to its capacity. When reserved-but-not-yet-written allocations fill an SSD tier, the cost model fails. Instead of servicing the performance, premium capacity is exhausted in empty regions that do not contribute value to the content being retrieved. The condition is then explained by the preallocation mechanism, why it is misattributed, and how to avoid it using a design pattern [1][2]. .

2. Preallocation Mechanics: Filesystem Allocation Behavior

2.1 Write Reliability as the Primary Benefit

In fact, a more accurate version of preallocation's benefits is that preallocation is a write-path reliability mechanism: the earlier the preallocation, the less likely an append would chunk out of space later in the file's lifetime. For ingest pipelines that perform writes of variable segment sizes, this reservation operates as an operational guardrail. Once a segment is written, the pre-allocated extent commitment has a high likelihood of reaching a clean segment boundary before any free space pressure is applied.

Asymmetric file systems for disaggregated persistent memories further illustrate that fault tolerance and performance must be co-designed rather than simply derived from lower-level storage properties. These offload the metadata operations to a shared log that consists of a metadata log (mlog) and a data log (dlog). Measured persistence latency for an 8-byte mlog is 5.48 µs with one RDMA_CAS and one RDMA_READ in parallel (RDMA_CAS takes 4.8 µs and RDMA_READ takes 3.2 µs). For a 64B dlog write, the latency is 4.32 µs (for a 1024B dlog write, 5.53 µs, and 7.46 µs for a 4096B dlog write). The final write isolation and log replication time for the 8-byte DRAM mlog insertion is 5.32 µs, slightly lower than the baseline due to the DRAM arena design. dlog persistence times are 4.4 µs, 6.15 µs, and 8.97 µs for 64B, 1024B, and 4096B object sizes, corresponding to an average 11.10% latency overhead. Additional latency of additional write isolation and log replication mechanisms is bounded and predictable, and unlike other optimizations of these mechanisms, not open-ended. This shows that write-path commitment and reliability are not mutually exclusive in principle and allows them both to be jointly applied [3].

2.2 Why Contiguity Cannot Be Assumed

A long-lived myth about preallocation was that it would guarantee contiguous physical block placement, helping predictable sequential reads. Modern filesystems have long satisfied allocation requests with extents, variable-length ranges of contiguous blocks laid out according to current free-space topology, concurrent writer activity, deferred allocation policies, and background compaction operations. A single large reservation may instead be split in non-contiguous extents if the device's free space is already fragmented when the reservation is created.

Storage device bottleneck analysis of SSD performance models has found that the behavior of storage devices running mixed workloads is considerably more complicated than that of running sequential workloads, as internal SSD parallelism, channel contention, garbage collection interference, and write amplification compete to determine throughput and latency as a function of distributions of access patterns, queue depth, and device utilization, rather than simply based on logical block proximity. With sustained random write workloads and queue depths common in multi-stream media ingest, IOPS may drop 15 to 30 percent of peak rated performance. This is due to GC and RB pressure, not fragmentation of the logical allocation. Contiguous preallocation does not protect against these device-internal mechanisms, weakening the argument for contiguous read-path allocation [4]. Figure 1 shows the average control plane shared log operation latencies with no reliability (baseline), write isolation, and log replication enabled, respectively. In the experiment, over-provision the number of write/RDMA_CAS and read/RDMA_READ operations to be issued in parallel in order to minimize the total persistence latency, Dlog latency in the short (64B), medium (1024B), and long (4096B) file path scenarios. Replication overhead is the additional latency incurred due to write isolation and log replication as an average over all dlog sizes. Note that all latency values are given in microseconds (µs). [3]

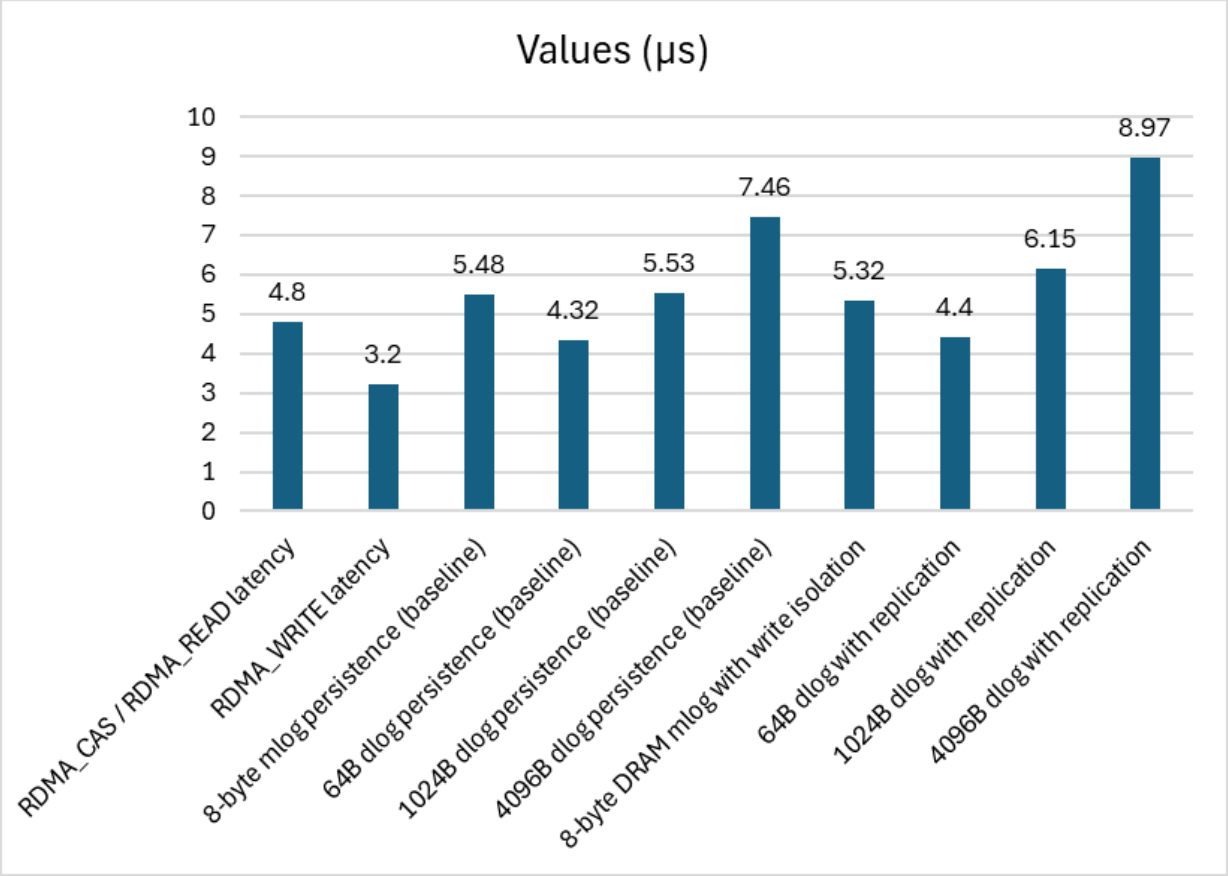

Fig. 1: Measured Log Persistence Latency in an Asymmetric File System on Disaggregated Persistent Memory [3]

3. Reserved-but-Unwritten Capacity: Fragmentation and Cost Amplification

3.1 How stranded allocations amass and increase pipeline costs

When files are finalized smaller than the upper bound preallocations, due to variable segment boundaries, measured bitrate change, duplicate detection, and decompression performance, these ranges are kept unless explicitly reclaimed by the application. The amount of allocated ranges wasted is directly visible for inline deduplication systems working on HDD-based primary storage. In a similar way, when analyzing RedHat's disk optimization feature called Virtual Disk Optimizer (VDO) while it is under a sequential write workload with 75% data reduction potential, the achieved I/O amplification factor is close to unity (10%) although reducing the logical size by 63%. The I/O pattern clearly explains this behavior: the I/O volume generated by the data after deduplication and compression constitutes 36.6% of the total I/O volume, the volume induced by duplicate verification reads is 45%, and the metadata I/O volume constitutes 8.2% [5]. As a result, most of the time that is spent on physical I/O is overhead, similar to reserved-but-unwritten allocations, which reserve space in physical memory but add little storage value.

Aside from its performance disadvantages, VDO's design introduces holes in the allocation because a physical block is immediately reserved for any I/O operation it receives, regardless of whether the data is new or even if it can be compressed. If the data turns out to be a duplicate, the reservation is then freed. This early reservation can be circumvented by on-demand allocation and block reservations only for confirmed unique, non-compressible data. For workloads with 90% data reduction potential, such optimizations reduce the I/O of verification reads by 67%, highlighting the cost amplification unreleased reservations impose on storage pipelines [5].

3.2 Fragmentation is a Long-Term Structural Tax

Fragmentation in large object repositories is an increasing degradation mechanism. Considering

read throughput, fragmentation in large object workloads has been evaluated to degrade read throughput by about 50% in case of object sizes of 256 KB up to 1 MB with increasing age of the storage device [6]. Volume occupancy affects fragmentation most: at 50% occupancy, as free space is limited on a volume, the fragmentation intensity more than doubles, and at the same workloads smaller or more densely filled volumes have considerably worse fragment-per-object ratios than larger volumes [6].

These dynamics apply to media archival fleet management, where a tier with reserved-but-unwritten capacity acts as if it were a volume at artificially elevated occupancy, incurring additional fragmentation without the added content value. Filesystem allocation policies, such as delayed allocation, can provide some best-effort mitigation of fragmentation by postponing block allocation until the target object size can be better determined. For example, the FFS reallocation policy reduced disk fragmentation by a factor of two in trace-based workloads [6]. However, fragmentation still remains an important performance bottleneck in scenarios with continuous allocation and deletion, such as long retention media storage archives. Freeing pools, which restores their free space to the real available space at finalization, directly decreases the fragmentation pressure due to occupied space [5][6]. The figure shows percentage-based read performance metrics obtained with two different workloads. First, the proportion of physical I/O volume broken down by type with VDO inline deduplication on a HDD for a sequential write workload with a data reduction potential of 75%. Second, the degradation of read throughput caused by long-term fragmentation in a large object repository.

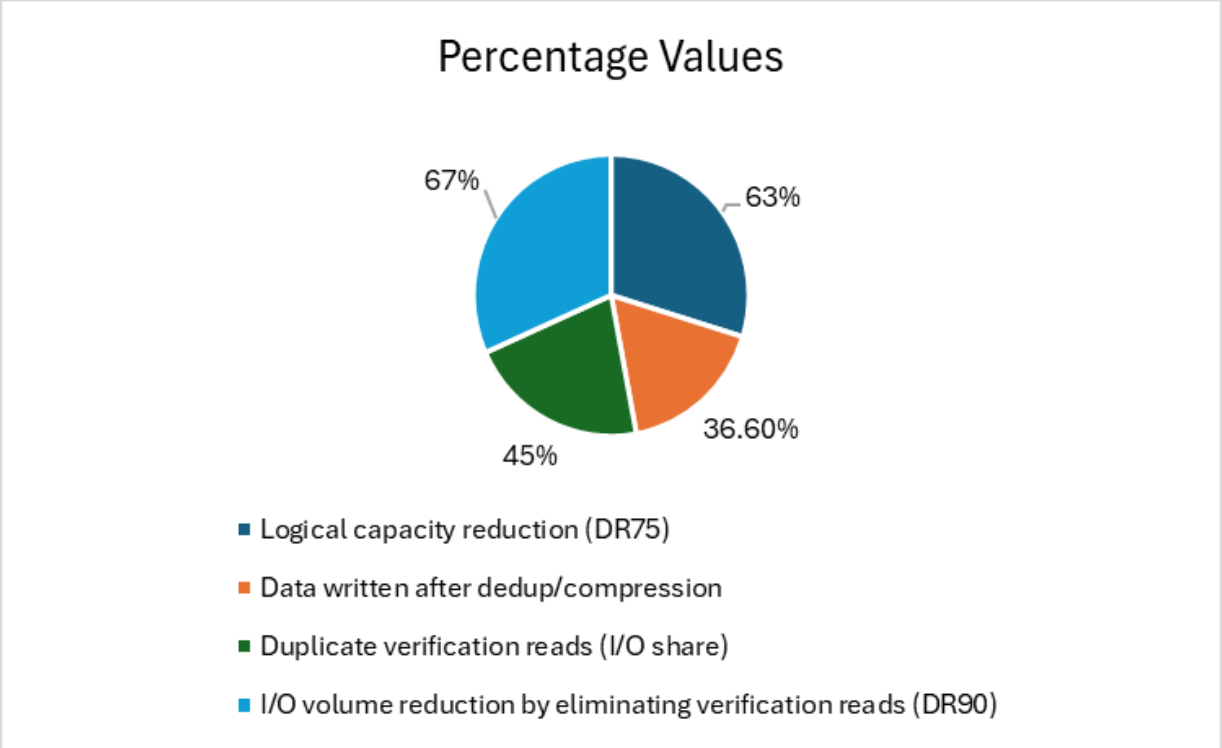

Fig. 2: Physical I/O Composition and Fragmentation Impact in HDD-Based Storage Systems [5]

4. Media Archival Architecture and Flash Storage Limitations

4.1 Hybrid Cloud Media Asset Management

Production media asset management solutions increasingly rely on hybrid cloud architectures that combine on-premises nearline storage with a cloud-based cold archive tier. Best-practice architectures store assets in nearline storage when they are accessed more frequently than a defined access threshold and store the remainder in a cloud-based cold archive tier. Accurate byte count accounting of actual vs. allocated bytes for storage footprint is operationally critical. Setting nearline tier boundaries based on preallocated rather than written footprints can deplete the nearline tier

prematurely and result in expensive cloud egress that wrecks the cost model [7]. Archival systems with telemetry separating preallocated bytes from written bytes enable the right tier boundary decision and prevent nearline overflow due to phantom consumption of capacity.

Due to lifecycle management, preallocation will interact even more heavily with the hybrid model. Realistically, assets need to be committed before being moved from nearline to cloud archive tiers. Unreclaimed allocations are moved to cold archive tiers. The cold tiers charge the reserved amount of used space, not the actual written amount, causing an equivalent increase in cold storage expenses as the ratio grows on on-premises storage. [7]

4.2 Tiered designs have restrictions due to flash memory

Because of the characteristics of flash memory storage systems, unreclaimed preallocation may cause the greatest cost at the SSD tier. A survey of host-side interaction with flash storage systems has presented write amplification, garbage collection overhead, and program/erase cycle wear as three of the primary mechanisms through which the allocation pattern affects SSD lifespan and performance. Write amplification factors from 2 to 10 have been measured for mixed read and write workloads with random write distributions due to erasing entire erase blocks before rewriting to pages in the same block [8]. Preallocated but not yet written pages undergo read amplification from scanning operations during background management tasks for integrity checking, backup enumeration, and tiering policy evaluation, thereby decreasing effective SSD endurance and using up P/E cycles even though the pages have no written content. Empty reservations in tiers consume the finite endurance budget of the SSD tier and negatively affect drive longevity, after which the cost is amortized by hot-data acceleration and not empty-reservation storage [7] [8]. Table 1 summarizes the architectural components, functionality, and storage requirements in hybrid-cloud media asset management and tiered flash deployments. It maps, in addition, to the unreclaimed preallocation constraint on the physical architecture, and to the requirements for accurate footprint accounting in the production archival systems.

Architecture Component	Tier Role and Function	Preallocation Obligation
Nearline on-premises storage	Holds assets with retrieval frequency above defined access thresholds for low-latency user-facing operations	Tier sizing must be based on the written footprint rather than the preallocated footprint to avoid premature tier exhaustion
Cloud cold archive tier	Absorbs assets that have aged beyond active production windows and serve long-term retention requirements	Assets must be finalized and reclaimed before migration to prevent reserved-but-unwritten capacity from inflating cold storage costs
SSD hot tier in tiered deployments	Accelerates access to hot metadata, index structures, and frequently retrieved media assets	Unreclaimed allocations on the SSD tier consume endurance budget and premium capacity on regions carrying no content value
Background scanning processes	Integrity verification, backup enumeration, and tiering policy evaluation traverse allocated regions regardless of content	Preallocated-but-unwritten regions contribute to read amplification and program/erase cycle consumption without informational return
Telemetry and footprint accounting	Distinguishes preallocated bytes from written bytes to support accurate tier boundary and capacity planning decisions	Systems lacking this distinction cannot detect stranded capacity accumulation until it manifests as user-visible latency or tier overflow

Table 1: Hybrid Archival Architecture Components, Tier Roles, and Preallocation Obligations [7][8]

5. Deterministic Reclamation and Tail Latency Restoration

5.1 Enforcing tail latency in tiered storage

The overt consequence of unreclaimed preallocation in tiered storage is inflation of tail latencies. Hot index structures and metadata for preview retrieval and clip navigation must be served by HDDs when SSDs are packed with stranded allocations. The difference in SSD and HDD latencies 2 to 3 orders of magnitude for random access, becomes more noticeable in the p95 or p99 user-visible latencies once these index structures are off the SSD tier.

Predictive control-theoretic frameworks for deterministic tail-latency enforcement in multi-tier storage architectures have been constructed with deep reinforcement learning that can satisfy latency SLO constraints under variable workloads by cleverly scheduling tier placements and I/O scheduling choices based on predictive horizons rather than customary reactive policies. Reinforcement learning agents trained on multi-tier storage architectures achieve p99 latency target adherence rates of over 95% under variable ingest workloads, compared to 60-75% for threshold-based reactive schedulers [9]. This assumes that tier occupancy is proportional to value-bearing content rather than stranded capacity. A control agent occupying a tier with empty allocations cannot make appropriate placement decisions since the occupancy monitor does not correlate well with actual demand for retrieval operations [9].

5.2 Observability as a Lifecycle Validation Tool

To enable a sound production-grade preallocation lifecycle, it is worthwhile to carefully check reclamation invariants and regressions. Benchmarking frameworks for storage observability can provide evidence that storage failure and performance regressions need monitoring metrics to distinguish logical and physical states throughout the storage stack. For the entire observability instrumentation, it needs to expose a few metrics: preallocated, written, and reclaimed bytes; reserved-but-unwritten ratio; used/reserved tier occupancy split; and p50/95/99 latency [10]. For example, if the system reports just the total capacity usage and not its split into writing and reserving a footprint, it will have no idea where stranded allocation is happening (the reservation of capacity with no writing) until the user-visible latency degradation occurs, which might be very severe. To detect reclamation failures before tier pressure exceeds acceptable levels, metrics export at segment finalization events is combined with the fleet-wide aggregation. Table 2 illustrates how the lifecycle of a complete preallocation-and-reclamation design pattern maps onto the controlling mechanism and production indicators for each of the pattern's lifecycle stages. It also discusses observability instrumentation requirements for supporting lifecycle validation and early regression detection, which derive from predictive tail-latency enforcement frameworks and storage observability benchmarking.

Lifecycle Stage	Control Mechanism	Production Outcome
Early reservation	Capacity committed at segment start before active growth, providing write completion assurance for the duration of the ingest window	Ingestion pipelines complete segment writes without encountering late-stage out-of-space failures under sustained throughput pressure
Finalization detection	State transition triggered by segment boundary closure, writer release, index commit completion, or retention policy evaluation	Reclamation executes deterministically at the correct point in the file lifecycle rather than being deferred indefinitely
Unused extent reclamation	Truncation to actual written size, deallocation of unused extents to create sparse holes, or rewrite into a compact finalized form	SSD tier occupancy reflects value-bearing content rather than stranded capacity, restoring intended hot-index locality
Tier placement accuracy	Predictive control frameworks use occupancy signals that reflect written content to make placement and scheduling decisions	Latency SLOs are maintained under variable ingest loads; reactive schedulers operating on inflated occupancy signals lose placement accuracy

Observability instrumentation	Continuous export of preallocated bytes, written bytes, reclaimed bytes, reserved-but-unwritten ratio, and latency distributions at p50, p95, and p99	Reclamation regressions are detected proactively at segment finalization events before tier pressure propagates to user-visible latency

Table 2: Reclamation Lifecycle Stages, Control Mechanisms, and Production Outcomes [9][10]

Conclusion

Reframing Preallocation's Operational Role

Preallocation is generally a good architectural choice for append-heavy media ingest pipelines, with performance gain, the real gain being that the chances of a write completing for in-progress segments under space pressure are improved. This is related to the distributed object storage patterns, which use allocation negotiation due to their complex metadata hierarchies. Fleet-scale reliability also ensures the operational reliability and the overhead correlated with retrying failures, which reduces the overall pipeline's complexity and the latency variation .

In practice contiguous physical layout does not ensure preallocation, as extent-based allocators do preallocation according to free-space topology at reservation time, contention with concurrent writers, and device utilization levels. Also, the serialization to adjacent (physically contiguous) locations (in the same physical strip) is less meaningful in SSDs due to garbage collection, write amplification, and contention within and between channels. Solid-state storage devices that rely on preallocation for high-speed sequential reads therefore have fragile, workload-dependent performance that degrades as the device is filled and ages .

The Structural Cost of Incomplete Lifecycle Management

The other and often greater drawback of incomplete reclamation is the waste of storage space. The unreclaimed space, consisting of preallocated but not actually reclaimed space, has an all-zero footprint that grows indefinitely. This wastes storage space, decreases deduplication and compression effectiveness (by increasing the zero-block metadata index), increases free-space fragmentation (by approaching high-occupancy limits), and consumes SSD program/erase cycle endurance on space that has no possible value. Further, hybrid cloud archival architectures push this stranded footprint onto cold archive tiers at the time of asset migration into the cloud, which inflates cloud storage costs proportionally to the reserved-but-unwritten ratio.

Design Requirements for a Full Lifecycle

A correct implementation requires three deterministic components; the first is early reservation in the file growth phase for append completion guarantees. Finalization must be a reliable state transition: a writer has signalized and closed a segment boundary, released a writer, committed an index, or passed a retention policy, not a race. Reclamation must be coupled with finalization: by truncating to the actual written size, deallocating unused extents to produce real sparse holes, or rewriting as a compact finalized form. This required observability instrumentation (e.g., recording preallocated, written, and reclaimed bytes at the segment level, distinguishing used versus reserved tier capacity) to verify lifecycle correctness in production and to catch reclamation regressions before they become user-visible latency.

Final Assessment

If reservation and reclamation aren’t handled as a single lifecycle, stabilizing ingest performance through preallocation can create hidden structural waste. This reduces tiering efficiency, raises infrastructure costs, and can significantly increase tail latency for user-facing retrieval workloads. Deterministic reclamation, in conjunction with preallocation, allows the default reclamation tax to be avoided, and the tier intent of tiered SSD/HDD deployments to be restored, without the reliability benefit of preallocation during ingest being paid for by sustained degradation across the retention lifecycle.

References

[1] Feng Wang, "STORAGE MANAGEMENT IN LARGE DISTRIBUTED OBJECT-BASED STORAGE SYSTEMS," UNIVERSITY OF CALIFORNIA, 2006. Available: https://ssrc.us/media/pubs/b478452bd61cc3cb3510ed6ea8750d5d93f2affd.pdf

[2] Joshua Silvia, "Tiered storage for AI: scalable performance and cost control," solved Magazine. Available: https://www.solved.scality.com/tiered-storage-for-ai-scalable-performance-and-cost-control/
[3] Miao Cai, et al., "Achieving Both Performance and Reliability in An Asymmetric File System on Disaggregated Persistent Memory," ACM Digital Library, 2026. Available: https://dl.acm.org/doi/epdf/10.1145/3760403
[4] Jihun Kim, et al., "SSD Performance Modeling Using Bottleneck Analysis," IEEE Computer Architecture Letters, 2018. Available: https://www.computer.org/csdl/journal/ca/2018/01/08126227/13rRUy3gmZo
[5] Patrick Raaf et al., "From SSDs Back to HDDs: Optimizing VDO to Support Inline Deduplication and Compression for HDDs as Primary Storage Media," ACM Digital Library, 2024. Available: https://dl.acm.org/doi/full/10.1145/3678250
[6] Russell Sears and Catharine van Ingen, "Fragmentation in Large Object Repositories Experience Paper," Conference on Innovative Data Systems Research, 2007. Available: https://www.cidrdb.org/cidr2007/papers/cidr07p34.pdf
[7] Kelly Messori, "Best practices: Archiving your media assets with hybrid cloud MAM," Iconik, 2025. Available: https://www.iconik.io/blog/best-practices-archiving-your-media-assets-with-hybrid-cloud-mam
[8] Jalil Boukhobza, et al., "A Survey on Flash-Memory Storage Systems: A Host-Side Perspective," ACM Digital Library, 2025. Available: https://dl.acm.org/doi/10.1145/3723167
[9] Torsten Jacob and Bluusun LLC, "Deterministic Tail-Latency Enforcement in Multi-Tiered Storage Architectures: A Predictive Control-Theoretic Framework via Deep Reinforcement Learning," ResearchGate, 2025. Available: https://www.researchgate.net/publication/399362419
[10] Duo Zhang and Mai Zheng, "Benchmarking for Observability: The Case of Diagnosing Storage Failures," BenchCouncil Transactions on Benchmarks, Standards and Evaluations, 2021. Available: https://www.sciencedirect.com/science/article/pii/S2772485921000065