International Journal of Sustainability and Innovation in Engineering (IJSIE)
https://www.doi.org/10.56830/IJSIE2026
Vol. 4 2026

Storage Virtualization for Hybrid Cloud: A Policy-Driven Object Storage Layer with Placement, Compliance, and Cost Constraints

https://www.doi.org/10.56830/IJSIE202615

Chiranjevi Sai Venkat Kaushik Dhulipudi
Software Engineer, USA
Email: dhulipudichiranjevi@gmail.com

Sashank Siwakoti
Forward Deployed Engineer, USA
Email: sashanksiwakoti1@gmail.com

Received:20 July, 2026, Accepted:30h October, 2026, Published: 11th November, 2026

Abstract
The study proposes a policy-based S3-compatible virtualization layer that can be extended to both private MinIO/Ceph and public object storage services. It also analyzes five representative traces (T15, T29, T65, T78, and T79), containing about 17.4 million requests, within a larger set of about 1.6 billion requests that represent 342 million unique objects from a collection of traced IBM Cloud Object Storage requests. The 5 reported modeled-cost comparisons have a low of 38.7% modeled cost reduction compared to static tiering, a high of 47.2%, and an unweighted mean of 43.7%. These numbers refer to differences in the modeled totals, and do not represent statistically significant differences or newly confirmed cloud bills. Geography or regulatory classifications are not recorded with the dataset, and assumptions about compliance and causation of egress are made during simulation but not included in the dataset. HMAC-SHA-256 is used to validate the authenticity of policy metadata, rather than physical residence. Allows for repeatable assessments such as measurable tariff snapshots, baseline definition, fault-injection tests, and more.

Keywords;
Storage virtualization, Hybrid cloud object storage, Policy-driven placement, S3-compatible storage, Data residency compliance.

1. Introduction
Hybrid-cloud storage is a prerequisite for any organization that stores data on-site and wants to keep some in the cloud to provide elasticity and burst compute functionality. The issue is that object placement can affect latency, storage cost, transfer cost, data locality, and retention requirements. The IBM Cloud Object Storage trace collection was large enough to illustrate the scale; 98 traces, each representing a week of operation, generated a collection of about 88 GB of logs that contained about 1.6 billion REST object-storage requests with 342 million distinct objects. These workloads become expensive and harder to manage with static placement.

This provider pricing motivates policy-aware placement. However, AWS S3 Standard is about $0.023 per GB-month for the first 50 TB, and cross-region transfer is about $0.02 per GB. Google Cloud Standard storage costs ~$0.020/GB-month in us-east1, US internet egress is initially up to ~$0.12/GB, and inter-region transfer in the North America region is

~$0.02/GiB. Azure charges about $0.02 per GB to transfer data from North America to Europe, and premium-network internet egress starting at 100GB is $0.087 per GB. This means that the more of this sort of transfer there is, the more might be spent in cheaper storage overall.

Typical storage concepts and lifetime tiering routines usually address only one aspect of these added complications. Current methods use largely static destinations and optimize performance and cost across heterogeneous backends. However, both solutions are flawed; the lowest-cost location may not be ideal for latency, egress charges, or local residency and retention requirements. Common APIs are not the end of the hybrid cloud. It must be discoverable and queryable across private and public object stores under the same namespace; responsive to workload behavior; able to transfer storage-provider-specific storage properties and transfer pricing; provide simple primitives for residency and retention before placement; and support a low-overhead data path. This need requires a dedicated, policy-driven hybrid object storage control plane that can optimize performance, compliance, and cost simultaneously.

There are several goals to this study. The goal is to build an S3-compatible virtualization software stack comprising object-storage backends, such as on-premises or public cloud, with built-in operational efficiency. It also aims to design an Adaptive Record Placement controller considering the frequency of access, storage cost, egress fees, residency, and retention criteria. The goal is also to set up the architecture and test it in a fenced experiment that has appropriate IBM COS traces and stress workloads to validate the architecture in a real environment. It also measures and baselines cost, cross-boundary traffic, latency, throughput, metadata overhead, and Migration performance.

The results were collected from an S3-compatible object storage system, so there was no virtualization of block or traditional file storage. It took into account private storage using MinIO/Ceph like devices, tiers exposed to the public cloud, and the GET, PUT, HEAD, and DELETE operations from IBM. These metrics include cost, latency, residency and retention, placement in EU versus US, local placement, trace replay, and synthetic benchmarking. Not quite as comprehensive as it is for all cloud providers or for all jurisdictions, or for all workloads.

For this research, the paper has been divided into different chapters. The study is grounded theoretically and technically in Chapter 2, which outlines the previous works on storage virtualization, adaptive placement, cloud cost, and cloud compliance. Chapter 3 describes the experiments' data sets, sampling strategy, experimental testbed, analysis procedures, and, finally, the evaluation metrics and baselines. The new control plane and its integration of the S3 naming namespace, compliance with the new policy, and zero-downtime migration designs are addressed in Chapter 4. The results of the experiments done and the comparison on a point-to-point basis with the work done in previous chapters are discussed in Chapter 5. The research is summarized, and future research directions and limitations are suggested in the conclusion.

2. Literature Review

2.1 Storage Virtualization and Hybrid Cloud Object Storage

Data storage virtualization has evolved from primary virtualizations linked directly to the media or provider to logical virtualization, where applications can interact with data without being directly tied to the media or provider (Carrizales-Espinoza, Sanchez-Gallegos, Gonzalez-Compean, & J. Carretero, 2024). This would make Object Storage more suitable, since it is more effective with massive sets of unstructured data and exposes objects using an

identifier-based approach. A persistent private dataset can be deployed on its own infrastructure and expanded elastically, and a public cloud could serve as a hybrid regional deployment solution (Azumah, Maciel, Sørensen, & S. Kosta, 2023).

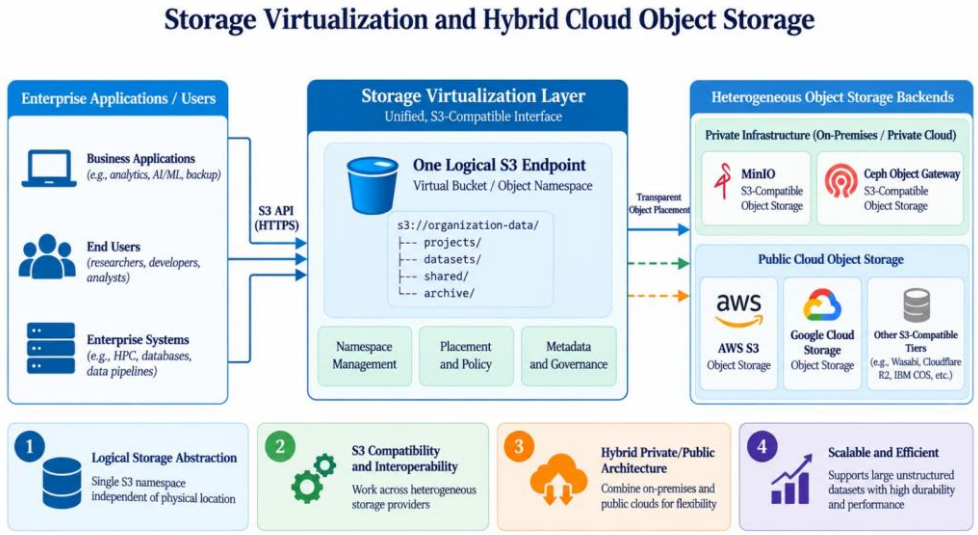


Figure 1: Securely integrates unified S3 access on private and public object storage backends in a storage virtualization architecture.

Figure 1 presents the hybrid cloud object storage virtualization architecture. The S3 API is the endpoint for enterprise applications, users, and systems; it represents a single logical S3 endpoint. The virtualization layer manages namespaces, placement, and metadata, and seamlessly and transparently routes objects between MinIO, Ceph, AWS S3, Google Cloud Storage, and other S3-compatible backends. This design ensures interoperability, flexible placement, scalability, and reliable, location-independent access.

Ceph Object Gateway and MinIO can mimic this model, both providing S3-like object access to distributed storage (Poat, Lauret, & T. Rao, 2024). In an S3-compatible application, applications can access data via homogeneous interfaces, while placement services route objects to their corresponding backend services. Secure access to and protection of data and services are part of coordinated data access and protection that ensures proper data and service management and resilience, similar to platform research on secure and resilient platforms (Sardana, Hariharan, & R. S. Pudi, 2025). The physical location of a Physical Object can change without changing the Namespaces the client sees.

2.2 Unified Namespace, Proxy Routing and Control/Data-Plane Separation

The unified namespace abstracts away the storage systems using a virtual bucket, logical object identifiers, and a backend-independent API. Metadata services hold bindings of logical service names to their physical replicas, and routing components forward requests to the right service endpoint. This design removes the application's reliance on provider resources' physical location and allows the service provider to change without changing the client application code (Ristov, Brandacher, Hautz, Felderer, & R. Breu, 2024).

However, architectures with a centralized control service that proxies payloads can create throughput bottlenecks and tail latency. The proposed design separates metadata routing from high-volume data moves. The control plane determines policy and object location, and the data path transfers object bytes to a lightweight proxy (Flinn, et al., 2022). Table 1 presents some dimensions for considering a hybrid or multi-cloud approach, such as

unified namespace, dynamic data placement, cost awareness, residency constraints, retention awareness, and control-versus-data-path separation.

Table 1: Comparison of Hybrid and Multi-Cloud Object Storage Approaches

System / Approach	Unified Namespace	Dynamic Placement	Cost Awareness	Residency Constraint	Retention Awareness	Control/Data-Path Separation
Ceph Object Gateway	Yes	Limited	No	Limited	Yes	Partial
MinIO	Yes	Limited	No	Limited	Yes	Partial
SkyStore / GlobalStore	Yes	Yes	Yes	Limited	Limited	Partial
Skyplane	No	Yes, for transfer routing	Yes	No	No	Yes
Centralized Proxy-Based Storage Virtualization	Yes	Limited	Limited	Limited	Limited	No
Proposed Policy-Driven Framework	Yes	Yes	Yes	Yes	Yes	Yes

2.3 Policy-Driven and Adaptive Object Placement

Adaptive placement can optimize static tiering by rethinking object placement over time as workloads change. Such signals include, but are not limited to, popularity, access frequency, recency, the size of the requested object, the request origin, latency to the destination, the number of replicas, and replica placement. SkyStore and GlobalStore boast of combining multi-cloud APIs and adaptive time-to-live placement that can deliver up to 6x cost improvement over published baselines (Liu, et al., 2025).

The optimization of bulk inter-cloud transfer routes is another layer, and here the company reports that they have achieved up to 4.6x throughput for a single cloud and 5.0x throughput for transferring data between cloud infrastructures. The discovery suggests it will help produce a more effective distributed cloud by increasing transparency in cloud decisions. Further, an investigation on multi-cloud automation demonstrated that deployment conditions, cloud provider infrastructure differences, and operational feedback are crucial for performance and must not be omitted (N. M. K. Koneru, 2025). Cost and performance optimization cannot be guaranteed for compliant placement.

2.4 Egress Cost, Storage Tiering and Workload Access Patterns

The ratio between Storage and networking usage, and thus each type of usage's share, depends on workload composition. Published analyses of IBM Cloud Object Storage traces document substantial variation in object sizes and access patterns (Eytan, Harnik, Ofer, Friedman, & R. Kat, 2020). Therefore, object count and byte volume measure different pressures. Static Hot/Cold tiers can simplify storage management but will not adapt to changing access patterns. As repeated network operations become more expensive than maintaining another copy, cost-aware caching/replicate-on-read policies reduce the number of GET operations sent to the remote node (Gefen, Zuck, Bransky, Hershcovitch, Harnik, & D. Tsafrir, 2025).

Some workloads have demonstrated as much as 85% savings in the cloud for caching in Hybrid-cloud studies at the high end. Therefore, placement analysis should consider factors such as storage price, egress asymmetry, migration cost, request frequency, and the lifetime of predicate replicas. If access is frequent across borders and overall network costs are higher, the most cost-effective storage site may not be the most cost-effective operating site (Khan, Matskin, Prodan, Bussler, Roman, & A. Soylu, 2024).

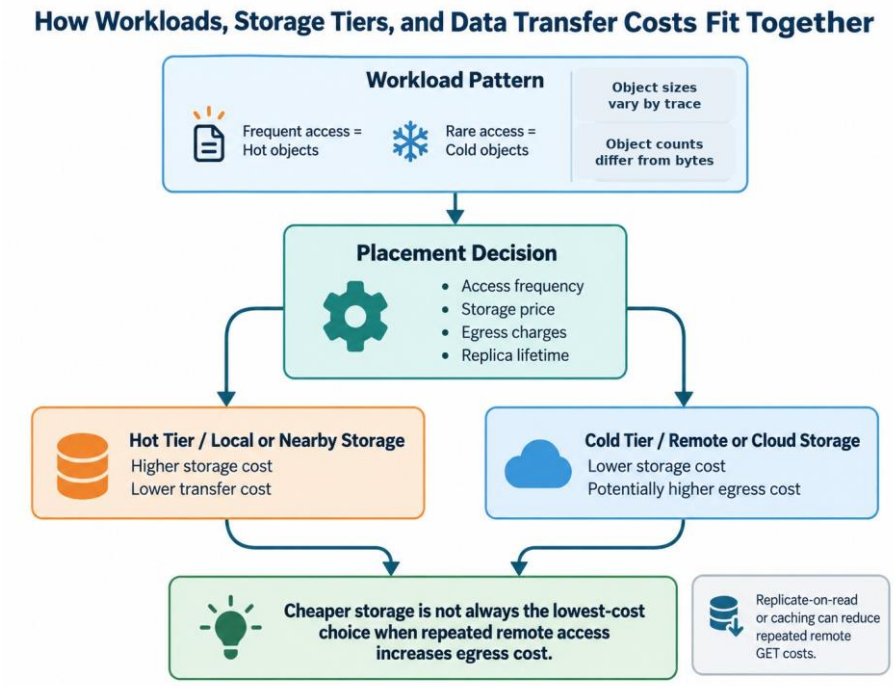


Figure 2: Decisions about workload access patterns also affect the storage tier, egress costs, replication, and overall operating expenditure.

Figure 2 shows how workload access patterns, a data object's storage location, and storage transfer costs relate in a hybrid cloud. Frequently accessed objects are moved to hot or nearby storage, while infrequently used objects can stay in colder, cheaper storage. Placement decisions take into account access frequency, storage pricing, egress fees, and replica lifetime. The chart also illustrates that frequent remote access can drive up the costs of lower-priced storage due to total egress charges.

2.5 Data Residency, Retention and Policy Enforcement

Data residency refers to the physical location of data, and data sovereignty refers to jurisdiction over the data. Retention is the window of time that information is required to be held, and immutability is the window of time that information cannot be altered or removed. With cross-border transfer and geographic replication, constraints are added and applied to other replicas that may be replicated to jurisdictions that were not approved for specific data classes. Article 44 of GDPR caters to this by stating that transfers may be made to third countries when appropriate safeguards are implemented (Juliussen, Kozyri, Johansen, & J. P. Rui, 2023).

Amazon S3 Object Lock has governance and compliance retention modes, while Open Policy Agent has policy-as-code decision evaluation (Pallewatta & M. A. Babar, 2024). Signed HMAC metadata can be used to attest to the integrity and authenticity of a securely managed claimed residency label but, alone, may not show the physical whereabouts of an object. Moreover, there are various aspects on which physical residence assurance will rely, including trusted provider-region configuration, endpoint-replication inventory audit, and endpoint-replication and failover control.

2.6 Limitations and Research Gaps in the Literature

The literature provides a wealth of information, but four limitations arise from the approaches reviewed. Many systems concentrate on cost or performance and not on hard-coding compliance within a modeling domain. Business continuity and preservation are also

dependent upon providers and can be difficult to achieve in a hybrid infrastructure with a portable environment. Typically, policy engines can only turn an action on or off, and are not used for adaptive multi-objective storage placement per se according to workload and tariff signals. There are also multi-cloud cost optimization systems, such as SkyStore, that do not necessarily involve residency notification and the data-path separation, control, and retention. The gaps provoke questions about policy-based placement and how it will help lower costs, prevent cross-boundary data traffic, enforce data-residency compliance, keep latency in check, and minimize disruption when moving data around.

3. Methods and Techniques

3.1 Data Collection Methods

The study used three data sources. The main workload venue was the SNIA IOTTA IBM Cloud Object Store trace workload, which contained 98 week-long traces, 88 GB of trace records, ~1.6 billion requests, and ~342 million unique objects. Logged fields included time trends, REST operations, de-identified object identifiers, object size, and, optionally, byte ranges for GET/PUT/HEAD/DELETE operations. The full collection comprises 98 week-long traces; the five selected traces form the workload subset described below.

Five representative traces were selected for replay with an intention to have variation in the read/write ratio, object size, access frequency, as well as recency. In T15, 57% of requests were GET, and 43% were PUT, out of 1.6 million tests. T29 had 13.0 million requests, 70% of which were GETs; T65 had 0.3 million requests, 99% GET; T78 had 2.4 million requests, 95% GET; and T79 had 0.1 million requests, 89% GET. In total, the replay sample had 17.4 million requests. Table 2 summarizes the data sets, rate-card variables, and compliance-policy variables used in the tables. The reported workload counts and GET/PUT ratios are consistent with the previously published SkyStore trace characterization (Liu, et al., 2025).

Figure 7 therefore labels the five modeled-cost comparisons Case 1 through Case 5. Although the workload sample comprises traces T15, T29, T65, T78 and T79, the available plotting records do not establish a verifiable association between an individual cost case and a specific trace. The public IBM COS traces contain request and object attributes but do not identify geographic request origins, statutory residency classes or sensitivity labels. Any geographic analysis therefore depends on an explicit synthetic scenario rather than observed client geography. The original geographical assignment distribution, random seed, and sensitivity results are not recoverable, so that no causal relationship can be stated between the trace dataset as such and the reduction in cross-border egress or empirical compliance with jurisdiction**.**

Table 2: Research datasets, cloud rate-card sources, workload samples, and compliance-policy inputs used in the experimental evaluation.

Data Source	Dataset / Input	Scope or Sample	Key Variables / Fields	Role in the Study
Industrial workload traces	SNIA IOTTA IBM Cloud Object Store trace collection	98 week-long traces, 88 GB of records, approximately 1.6 billion requests, and about 342 million unique objects	Timestamp, REST operation, anonymized object ID, object size, optional byte range, GET, PUT, HEAD, DELETE	Descriptive characterization of real object-storage workloads and selection of representative traces for replay
Experimental trace T15	IBM COS trace T15	1.6 million requests	57% GET, 43% PUT, object size, access frequency, recency	Represented a comparatively balanced read-write workload
Experimental trace T29	IBM COS trace T29	13.0 million requests	70% GET, 30% PUT, object size, access frequency, recency	Represented the largest selected workload and a read-dominant access pattern
Experimental trace T65	IBM COS trace T65	0.3 million requests	99% GET, 1% PUT, object size, access frequency, recency	Represented a highly read-intensive workload
Experimental	IBM COS trace T78	2.4 million requests	95% GET, 5% PUT, object	Represented another strongly

Data Source	Dataset / Input	Scope or Sample	Key Variables / Fields	Role in the Study
trace T78			size, access frequency, recency	read-intensive workload
Experimental trace T79	IBM COS trace T79	0.1 million requests	89% GET, 11% PUT, object size, access frequency, recency	Represented a smaller read-dominant workload
Replay sample total	Five purposively selected IBM traces	17.4 million requests	GET/PUT balance, access frequency, object size, recency	Supported maximum-variation experimental replay across contrasting workload characteristics
Cloud-provider pricing	AWS, Google Cloud, and Azure rate cards	Original capture date and archived SKU snapshot not supplied; public current rate cards can only provide reference prices.	Provider, region, storage class, unit price, request category, egress charge, inter-region charge, transfer direction, currency	Supported calculation of storage, request, egress, inter-region, and migration costs
Compliance requirements	GDPR Article 44	Cross-border and regional data-placement requirements	Permitted region, prohibited region, transfer condition, residency status	Defined reproducible data-residency constraints for policy evaluation
Retention and immutability rules	Cloud-provider retention and immutability documentation	Applied to object-retention scenarios	Retention period, immutability state, deletion eligibility, policy status	Defined machine-readable allow or deny conditions before placement, migration, or deletion

The second source was described as AWS, Google Cloud, and Azure rate cards used at the experiment date; however, the date and historical SKU-level snapshot are unavailable for verification. These included storage, request, egress, and inter-region charges. The third source covered GDPR Article 44 requirements and supplier retention and immutability policies to create replicable residency and retention policies. To ensure the records are reproducible, the rate-card records also contained provider, region, storage class, unit price, operation category, transfer direction, and currency. The study mapped the compliance inputs to machine-readable allow-or-deny conditions to replay for every placement strategy during compliance testing (Nogueira & A. Resende, 2026).

In this model, costs are subdivided as C_total = C_storage + C_requests + C_transfer + C_migration + C_retained_replica so that no costs overlap. The following occur: Billable traffic amounts in bytes are present in C_storage, but staging and separate replicas kept are not included; migration ordinary object tasks are present in C_requests and C_transfer, including ordinary Billable traffic, but migration-involved traffic amounts in bytes are not included. C_migration includes incremental copy/retrieval, request, network, and temporary staging charges not elsewhere reported. C_retained_replica includes additional occupancy information that is only present because of retention periods or legal holds, representing occupancy not used for any other categories after switchover. Rate considers the region, direction, storage class, quantity, length of storage, retrieval costs, minimum length of storage, as well as the GB/GiB counting principle. The sum totals are not preserved, so they cannot be used to determine that there was no double-counted total.

3.2 Data Analysis

The study performed trace record cleaning for invalidated timestamps, REST operation labels were normalized, object identifiers were tested/written to a unified format, and sizes of objects written to an extent were converted to a uniform value in bytes and, if present, the byte range was preserved. Each trace was analyzed to calculate the percentage of GET and PUT. It retrieved access frequency, time elapsed between two successive GETs, one-hit, cold access, warm access, hot access, and object-size class. This trace cleaning and workload classification support descriptive characterization of request mix, access recency and size distributions, without establishing causal effects (E. Ofer, 2021).

The study includes monthly equivalent charges for storage and request charges, egress charges, migration traffic charges, and inter-region transfer charges. The study assumed that private cluster infrastructure costs remain the same across policies and that local infrastructure

is the benchmark for all policies (Ke, et al., 2026). Therefore, the difference in the approaches' performance was attributed to policy behavior rather than an arbitrary gap in private storage valuation.

Because private-cluster capital expenditure, electricity, labor, maintenance and depreciation were treated as constant across configurations, the analysis concerns modeled variable storage, request and network charges, and not demonstrated savings in private infrastructure or total organizational operating expenditure. The reported record contains neither per-trace billable transferred bytes nor replay-time compression factors, live bucket transfer receipts, or complete replica-duration accounting. These omissions prevent independent reconciliation of the large modeled amounts against the workload request counts and object-size distribution.

3.3 Experimental Testbed and Baseline Configurations

The hybrid testbed comprised an in-house distributed 4-node MinIO cluster with members having at least 32 GiB of RAM, NVMe- or SSD-backed storage, and a 10GbE internal network. A secondary endpoint was set up to perform S3 checking in the study. The experimental testbed incorporated live AWS S3 and Google Cloud Storage buckets in United States and European regions as public-cloud storage targets (Khan, A. Q.; Matskin, M.; Prodan, R.; Bussler, C.; Roman, D.; A. Soylu, 2024). The live Azure replay was not included by researchers owing to cost control of experiments, with the Azure pricing used for the sensitivity analysis.

Two workload classes were used in the study. The first ran the five IBM traces, which had 17.4 million requests. Saturation Throughput and p99 latency were measured when running a 2nd loaded MinIO Warp with a combination of S3 workload types. Four different placement methods were investigated based on the same workloads: static-tiering, cost-only placement, nearest-region latency placement, as well as the policy-driven proposed controller. It was also able to resynchronize the clocks of the clients before each run, initialize the objects in the buckets to initial values required for each run, and remove the objects that were previously used in trials from the trials. The study further varied placement policies to mitigate workload-order bias and maintained the same sequence of objects and timing of requests across policies (Hardt, Nahhas, Müller, & K. Turowski, 2025).

Four baseline definitions establish the required comparison semantics: static tiering fixes placement in the initial policy-permitted tier; unconstrained cost-only placement minimizes the same complete storage, request, network, migration and replica-retention cost objective without residency constraints; nearest-region selects the lowest-latency permitted destination; and policy-driven placement minimizes the complete objective subject to residency, retention, capacity and latency eligibility. All configurations require the same workload, initial replicas, price table, forecast horizon and charging rules. Under a globally optimized common objective, adding constraints cannot produce a strictly lower mathematical minimum than removing constraints. Because the archived cost-only algorithm and cost components are unspecified, the figure does not support a claim of superiority over an unconstrained cost optimum. SkyStore and Macaron are relevant system baselines, but their measured performance cannot be inferred from the available experiments.

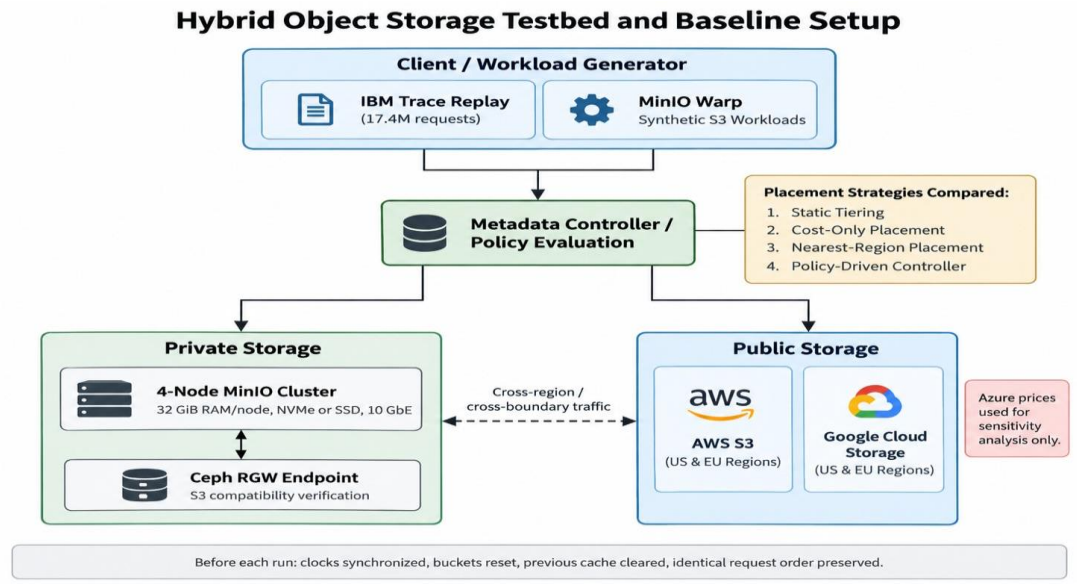


Figure 3: Object storage testbed showing workloads, policy controller, private storage, public clouds, and baselines.

Figure 3 illustrates the experimental hybrid object storage testbed that is used to evaluate object placement strategies. It shows how workloads from IBM trace replay and MinIO Warp are routed through a metadata and policy controller to private MinIO or Ceph storage regions, or to public AWS S3 or Google Cloud Storage regions. The figure also shows cross-region traffic, Azure pricing sensitivity analysis, and pre-run controls to standardize benchmarking across all configurations.

3.4 Policy Model and Placement Decision Logic

Signed policy metadata was attached to each object, which included the residency, retention_until, sensitivity, replica_min, preferred_latency_class, and a policy-version identifier. The controller authenticated placement decisions using HMAC-SHA-256 on the residency tags. The controller identified destinations that did not meet residency, retention, capacity, or latency requirements and removed the first. The controller ranked all eligible destinations by projected storage, egress cost, frequency of use, recency, request-origin geography, measured latency, current replica count, and migration cost (Venkatesh, Chandrashekar, Rao, Sridhar, & S. Rajanna, 2023). Therefore, simply because it is cheaper, an EU-residency object should not be moved to a United States backend.

For a reproducible implementation, let B(o,t) denote feasible backends for object o at time t after authentication and enforcement of residency, retention, replica-count, capacity and latency restrictions. Over forecast horizon H, choose b* = argmin[b in B(o,t)] J(o,b,t), where J is the sum of anticipated capacity-time storage cost, GET/PUT/HEAD operation charges, origin-weighted egress charges, inter-region transfers, one-off migration expense and temporary retained-replica cost. Candidates are ordered by J with a deterministic backend-ID tie-break. Decisions occur on a relevant object or policy event and at a configurable period Δ.

Because the objective J already includes one-off migration expenses, the controller migrates between eligible backends only if the complete-objective improvement J(o,b_current,t) − J(o,b_candidate,t) exceeds ε and the object has remained in its current eligible placement for at least τ. Migration cost is therefore charged only once. If the current placement becomes prohibited by a policy change, the controller must prioritize an eligible destination instead of delaying corrective action for economic hysteresis. H, Δ, ε, τ, popularity estimator, and backend prices are configuration inputs and should be published with the code.

This specification is for explicit reproduction, not a description of the implementation of the controller used to calculate the totals that are displayed.

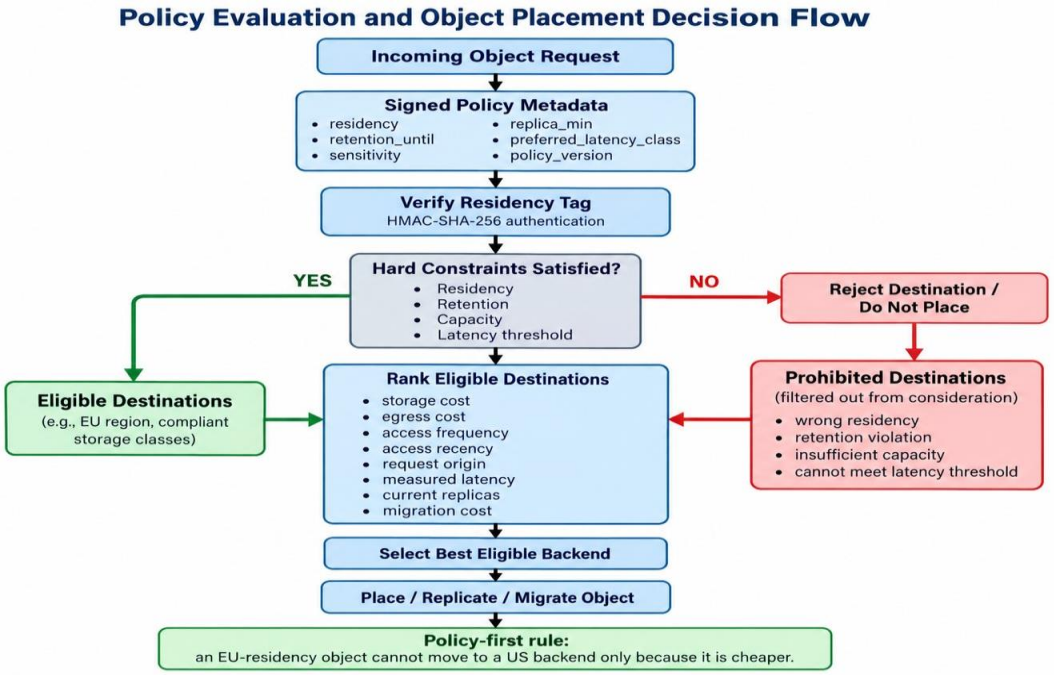


Figure 4: Policy-defined object placement workflow that checks compliance constraints prior to ranking eligible hybrid-cloud storage destinations for the run.

Figure 4 shows the hybrid storage controller's policy evaluation and object placement decision flow. Signed metadata for each request comprises Residency, Retention, Sensitivity, replica requirements, Latency class, and policy version. With HMAC-SHA256, the system checks residency tags before applying hard constraints. Policy-first control logic manages noncompliant destinations and ranks eligible backend placements, replicas, and backend migrations by cost, access behavior, backend latency, backend replicas, and backend migration cost.

3.5 Evaluation Metrics and Statistical Validation

These metrics included monthly-equivalent operating cost, cloud storage cost, cross-boundary egress (GB), cross-boundary egress expense, number of migrations, number of residency violations, number of retention violations, p50 latency, p95 latency, p99 latency, throughput, and metadata-routing overhead and migration downtime. The synthetic benchmarking protocol called for up to 30 repetitions where feasible, but run counts for individual configurations and the associated replicate logs are unavailable. Consequently, numerical latency summaries, standard deviations, interquartile ranges, and 95% confidence intervals cannot be independently reported or verified. Paired non-parametric comparisons across the five distinct traces were considered separately from synthetic repetitions, following established multi-dataset comparison principles (J. Demšar, 2006).

The five paired trace-level cost differences are reported descriptively ($n = 5$). With five nonzero pairs, the smallest possible exact two-sided Wilcoxon signed-rank p-value is 0.0625. All five displayed differences favor policy-driven placement; conditional on these five pairs and no ties or zero differences, the exact two-sided sign test likewise gives $p = 0.0625$. Thus the five-trace data do not establish significance at $\alpha = 0.05$. Independent repetitions of synthetic microbenchmarks are distinct experimental units and cannot be used

to enlarge the trace-level sample. Raw replicate data and 95% confidence intervals were not retained in the manuscript and are not estimated here.

For comparisons at trace levels, distributional assumptions for parametric testing were not defensible, so researchers used paired non-parametric tests. For each metric, static tiering served as the primary reference because it is the traditional placement approach without adaptations. Conformance was deemed a harsh measure, with all prohibitive residency/latency events considered violations regardless of whether cost increased or latency decreased (Dockara & M. Malhotra, 2026). The design enabled cost, compliance, and performance effects to be assessed in parallel, rather than treating any one measure as sufficient evidence of superiority.

4. Policy-Driven Hybrid Object Storage Virtualization Framework

4.1 Globally Unified S3-Compatible Virtual Namespace

The basic idea of the scenario is that the framework offers just a single logical S3 endpoint and that this endpoint is the same for all applications, and offers applications a single virtual bucket and object namespace on all the storage backends. Clients thus did not talk directly to S3-compatible tiers such as MinIO, Ceph, AWS S3, Google Cloud Storage, etc. Instead, a virtualization layer handled requests, abstracted the object name from the client, and tracked the object's location.

The additional key metadata forwarded requests to one or more backend replicas and enabled placement changes without changing the application code or the object identifier (Sadavarte, et al., 2026). This design was intended to preserve S3 request semantics, although concurrent-operation correctness was not experimentally established and allowed flexibility when dealing with private and public storage resources under a single logical system, with various components. It provided version information and, during failover, maintained backend independence. If one endpoint fails, clients can connect to another endpoint within the acceptable range without changing the request format (Goggin, Lekshmanan, Valverde, & E. Bocchi, 2025). This allowed seamless integration because the application layer was protected from provider-specific credentials or physical paths.

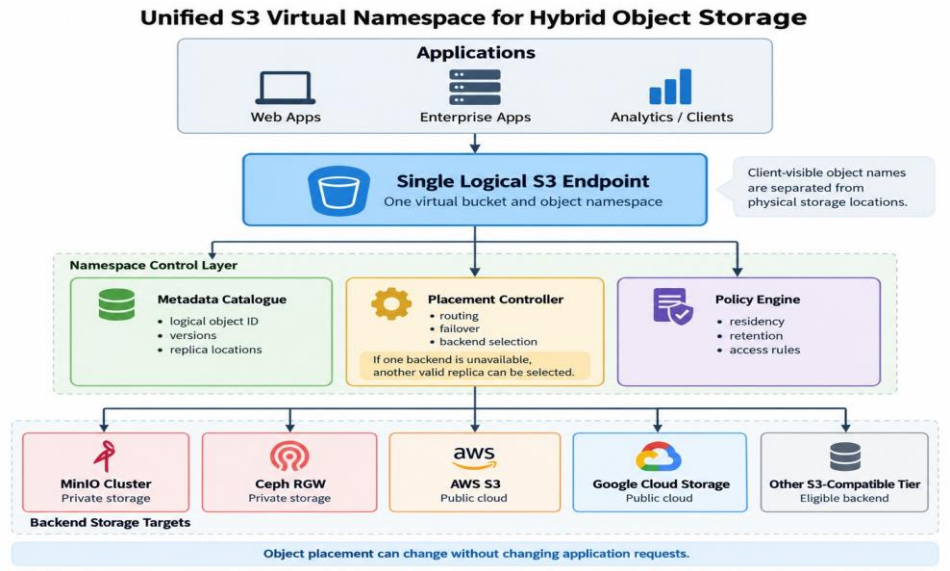


Figure 5: Unified S3-namespace architecture, which simplifies the seamless integration of applications, policy controls, metadata services, and hybrid backend storage.

Figure 5 shows the end-to-end architecture of a unified virtual namespace deployed back-to-back in the S3 system to serve applications and access a wide range of private and public object storage. The namespace control layer interfaces with the metadata catalog, placement controller, and policy engine, and forwards requests to MinIO, Ceph RGW, AWS S3, Google Cloud Storage, or any other available backend. This design ensures that object names known to clients differ from physical locations so that objects can move, replicated objects can fail over, backend independence is achieved, and policy-based routing is possible without modifying the request in the application.

4.2 Metadata, Compliance and Policy Enforcement Layer

The study maintained information on recent access statistics, residency class, retention policy expiration time, policy integrity, policy signature status, object versions, replica locations on disk, provider region, and policy label. The study ran pre-checks for policy conformance before advising on policy placement, replication, deletion, and migration. A policy decision point based on OPA and Rego can be defined for cases that do not include a governance rule in the storage code and trigger an evaluation of object and request attributes (Brambilla, Falconi, Fortina, Plebani, & M. Vitali, 2025).

Signed policy labels establish metadata integrity and authenticity only under secure key custody; they do not attest to physical data location or backend behavior. A deployment specification must define HMAC key creation and rotation in an access-controlled KMS/HSM, key IDs and expiration, canonical serialization, replay protection, audit trails and rejection of unknown/expired keys. The deployment must periodically reconcile backend-region configuration, bucket policies, replication endpoints and provider inventory. Missing or altered policy labels must cause new placements and routing decisions to fail closed. A policy revision triggers revalidation of existing replica eligibility and all in-flight migrations before a destination can be committed; protected replicas remain intact until retention restrictions and legal holds permit deletion. Where no policy-permitted replica is available, routing is denied rather than redirected outside an approved region. These controls reflect design needs, not the results determined in the reported study.

Data architectures also include data security, data scalability, data control vouchers, cybersecurity controls, real-time data decision-making, and the use of different types of data and data sources in distributed data environments (Bonthu, Malik, & M. R. Dhanagari, 2025). If the placement was requested and either granted or denied, the system created an audit event that included the object creation time, the placement decision time, the requested placement destination object, the requested residency state, and the migration action performed. This provided records of occurrences for post-compliance review and incident analysis.

4.3 Adaptive Multi-Objective Placement Controller

The adaptive placement controller took into account factors such as the frequency the objects are accessed, the recency of access, the size of the objects, the cost of cloud space, and the egress rate card, the origin of objects, the available regions, the observed latency, existing replicas, and the mandatory residency along with the retention requirements. The study conducted a preliminary eligibility assessment. Before the cost-ranking process, the study excluded destinations that required sovereignty and ownership, as well as destinations that violated the latency requirement.

The study filled the remaining targets using selection criteria such as estimated storage cost, estimated data-transfer cost on the storage network, recent request behavior, and estimated migration overhead. The controller enabled initial object placement, migration through re-tiering as workloads dropped, replica removal as retention or caching periods ended, and migration in response to changes in workload or price. This differed from

traditional lifecycle rules, where rules were set by age and never changed (Pang, Alazzawe, Ray, Kant, & J. Swift, 2023). It also was not about minimizing costs, since they had to be accommodated, meaning a high-cost backend was never really an option.

The reference implementation architecture consists of a versioned metadata catalog, an OPA/Rego-compatible policy evaluator, a registry of backend capabilities and prices, per-object access counters, and a migration work queue. For each placement event, the controller verifies the signed policy and key version, constructs the eligible destination set, estimates the common-horizon cost for each candidate, applies a deterministic tie-break and the migration hysteresis rules, and then records the decision and relevant policy version in an audit event. A worker stages and verifies a destination copy and commits a version-checked metadata pointer update before scheduling permitted cleanup. The design is sufficiently specified as a reference algorithm, but source code, configuration files and historical execution artifacts for the reported experiments were not supplied.

4.4 Decoupled Data Path, Migration and Availability

The architecture was designed to split the design between a data path and a control path. The control service did the matching and placed each object, and the real data was placed via a reduced-impact routing or proxying layer. This separation restricted its ability to be a performance bottleneck when transferring large amounts of data during policy evaluation (Zhu, et al., 2025). Seven progressive phases can be observed during migration. First, the controller evaluated the policy and picked an appropriate destination. It then began an asynchronous copy and, at the same time, computed the object's checksum and integrity.

After the destination object has been copied and a full-object checksum and version have been verified, the authoritative metadata pointer can be atomically changed by a compare-and-swap update, and subsequent reads will be against the source down to that commit. This ordering is designed for availability, but zero-downtime is not actually tested. PUT overwrite or Delete must be concurrent in a serial or versioned manner, such that no obsolete version becomes current, tombstones must be retained, and object generations must be verified before performing switchover. Complete or terminate multipart uploads while following consistent ownership rules; convert provider-specific ETags for objects, assuming that ETags from different providers are not identical (As Amazon states, the multipart ETag is not a full-object MD5 hash: https://docs.aws.amazon.com/AmazonS3/latest/API/API_Object.html).

When endpoints change, presigned URLs need to be proxied/renewed based on backend hostnames that they are referencing. Based on the retention holds and minimum requirements for replicas, the old versions are kept. The superfluous source replica is deleted upon exemption and not automatically kept until long-lived policy expiration, as the cost model will account for the overlap period when present. When an outage or failover occurs, the controller checks residency requirements and will refuse to pick route endpoints that are not in approved regions. The fault-injection measurements not provided in this study are all required to perform downtime, concurrent-operation correctness, and failover behavior.

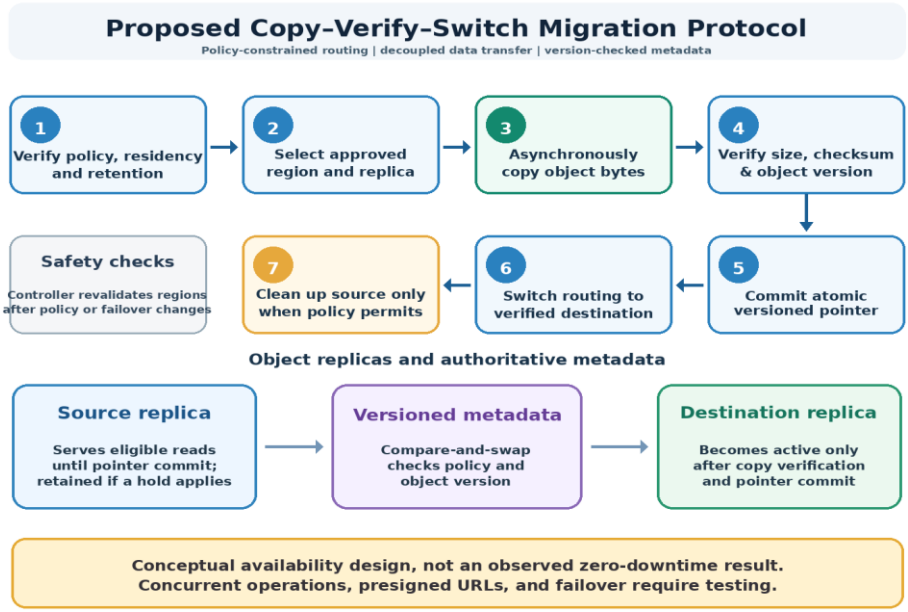


Figure 6: Proposed copy–verify–metadata-switch sequence. Availability and Interruption testing is a method for measuring zero downtime, and none is available for this evaluation.

Figure 6 is the list of steps that must be taken care of in the control and data planes: create a copy of the destination, verify the number of bytes copied and the correctness of the copied version, use the atomic pointer, delete the version of source replicas if allowed by the replication and retention policies. This does not imply zero downtime, concurrency, presigned URLs, or failover/single service interruption; these must be tested separately.

5. Results and Discussion

5.1 Egress and Operational Cost Reduction

The displayed modeled totals yield static-tiering relative reductions of 38.7% (Case 1), 43.8% (Case 2), 44.6% (Case 3), 47.2% (Case 4), and 44.2% (Case 5), computed as $100 \times (C_{static} - C_{policy})/C_{static}$. The maximum modeled-cost reduction across the five cases was 47.2%, while the unweighted mean of the five percentage reductions was 43.7%. The comparison considered only modeled charges and excluded the fixed infrastructure costs of the private storage cluster. The archived breakdown does not allow for the calculation of amounts of tariff inputs supplied to regions, nor lengths of replication, so the calculation of quantitative total differences in avoided cross-boundary egress is not possible.

Static tiering is the sole auditable comparison based on the available paired totals. The cost-only and nearest-region series in the earlier cost visualization lack sufficiently specified algorithms and per-component accounting and are therefore not used for an optimizer-superiority inference. No like-for-like replay of SkyStore or Macaron is recorded and such comparisons remain outside the evidential scope of this dataset. The benefit of policy prefiltering is an explicit feasibility constraint, not proof that constrained optimization dominates an unconstrained minimum.

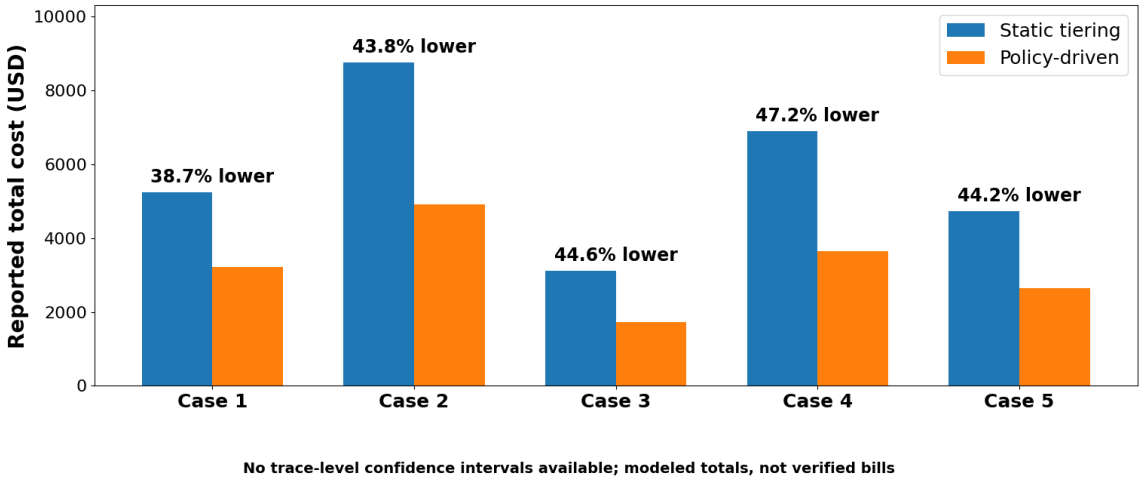


Figure 7: Reported static-tiering and policy-driven modeled totals for five modeled workload cases associated with the sampled IBM COS traces (USD). Savings are calculated from displayed paired totals. Uncertainty bars are omitted because trace-level replicates and confidence intervals are unavailable.

Figure 7 labels the five available paired modeled-cost observations Case 1 through Case 5. The corresponding trace identities are not verified by the surviving plotting records. Static modeled totals are $5,240, $8,760, $3,120, $6,890 and $4,730; the paired policy-driven totals are $3,210, $4,920, $1,730, $3,640 and $2,640. These values reproduce the available reported totals, rather than measured cloud invoices. No case-to-trace assignment is inferred from these aggregate totals. The missing replicate measurements preclude confidence intervals or error bars.

5.2 Compliance and Residency Enforcement

This abstract eligibility ensures safety in placement, since the placements that would be eligible but filtered according to the residency requirements, by controller's determination, are not allowed to select a prohibited destination. No records of any comparison placement policy's attempted placements of prohibited placements, physically misplaced replicas, retention failures, or completed unauthorized transfers exist. The result of the evaluation thus does not lead to a 100% operational rate of compliance. Some potentially relevant fault classes include when labels are changed or are missing, policy changes during the middle of a run, replicas, or legal holds during migration or failover to an off-allowed region when the system is "on the plane".

5.3 Adaptive Placement and Migration Availability Design

Proposed copy–verify–switch protocol maintains reads at a source with a valid version until a destination-copy integrity check and atomic pointer flip. It alone does not mean zero downtime. There are no operation counts available for failed operations, retries, or stale or interrupted operations, nor is there any probe timeline. The protocol should include support for concurrent PUTs, DELETEs, and overwrites, multipart state, backend-dependent ETags, presigned URLs, and failover. Replicas of source data that are retained or under a legal hold need to be kept secure until deletion is allowed. Temporary staging and copy-related storage costs during migration are captured under the migration-cost component, but source replicas retained after switchover to meet retention requirements or legal holds are captured under the retained-replica cost component.

After lawful deletion of a source replica, subsequent occupancy charges do not apply unless subject to any specified minimum-storage-duration occupancy charges. Repeat GET operations in read-heavy scenarios were more likely to trigger a replica-on-reads decision, as cross-boundary GET traffic can make it cost-effective to keep an additional replica (Park, Qiu, Ganger, & Amvrosiadis, 2024). Colder objects were candidates for re-tiering or eventual removal from their policy, if allowed. Additionally, policy checks prevented migrations whose intended destination conflicted with residency and retention objectives.

5.4 Integrated Discussion against the Literature

The cost comparison revealed that the cost of the policy-driven approach was less for all five cases studied, and that the greatest cost of calculated savings from the two approaches was 47.2% below cost for static tiering. The cost comparisons are descriptive results and will not be offered as proof of statistical significance in cost savings. Not necessarily evidence of superiority over an unrestricted cost optimum or 5 trace level of significance. There is an architectural eligibility component in policy prefiltering, but actual placement evidence, baseline violation counts, and adversarial testing are needed to ensure residency and retention compliance in the real world. Failure logs and the original benchmark need to be available to determine any defensible throughput, tail latency, or zero-downtime conclusion.

The economic viability of these experiments, as shown in the SkyStore experiments, has shaped the current access control paradigm by introducing private and public virtualization, constraints on access to virtual resources, cryptographically verified residency metadata, and a separation of data movement from the control plane. Several studies on enterprise clouds indicate that efficient scalability, analytics usefulness, and orchestration automation are important factors in complex clouds (Rangu, Chadha, & S. Samala, 2025). The proposed framework goes beyond traditional storage optimization by combining cost efficiency, regulatory compliance, system performance, and data migration management in a hybrid-cloud storage architecture.

5.5 Future Research Recommendation

Future work includes analyzing and adapting the automatic pricing model; deploying in carbon-aware cloud regions and clouds; supporting outage-based replication; deploying backend clouds in sovereign clouds; deploying cloud in a zone; and processing sensitive metadata. Experiments at larger scales on several petabytes are also required to measure controller scalability by object count and region size, and to process large volumes of metadata management workloads (Baron, et al., 2026). In addition, these would provide control-plane guidance on how capacity can support policy enforcement in the context of growing storage size, low-latency routing, and cost-effective placement in ever larger and more pervasive storage environments.

New machine learning technologies, such as forecasting and reinforcement learning, can make more efficient placement decisions by leveraging information on access patterns, workload size, and provider prices to predict future access patterns (Xing & Y. Wang, 2025), However, that does not mean predictive optimization can replace necessary governance. Until components can move to different locations, hard constraints such as residency, retention, and sovereignty should not be treated as optional optimizations. The extended incursion should therefore incorporate adaptive intelligence to improve storage efficiency and address storage-environment concerns in hybrid-cloud situations without upstream storage regulatory assurance, while remaining auditable and operationally secure.

6. Conclusions

The virtualization design plan is to have a single namespace that is controlled by a policy for where the object should be placed, whether in a public or private object store, to

support an S3-compatible interface. A maximum range of modeled cost reduction in the five modeled cost cases is between 39 percent and 47 percent, with a mean of 43.7 percent, and is subject to unverifiable cost inputs, traceability to charting, and missing replay information, based on the original static and modeled totals and cost reduction. This finding is with respect to charges reported in the model, not the savings in private infrastructure. Only 5 pairs of data are at TRACE level; therefore, the exact 2-sided signed rank test cannot be used to reject at the 5 % level of significance. Evidence does not detail results in compliance measured line-rate, sub-3% tail-latency, and zero downtime or independently verified compliance of 100%. Such performance and compliance results are therefore targets for future empirical testing that requires repeatable performance measurements, independent compliance measures, and controlled failure-test procedures.

The contribution will be an integrated approach to policy eligibility and adaptive location selection, and decoupled routing. Although the use of HMAC-SHA-256 can add some verification to the authentication and integrity of the policy labels, physical residency, provider inventory, and geographic configuration evidence must be included with it. Common objective baselines, rules for assigning the origin of a request, rules for transferring records by trace, per-trace bytes rule, concurrent operation/failure testing rules, and recording of public controller produce/code must be reproducible for historical region-specific rules.

This is still the case for a hybrid-cloud configuration, in which they need to terminate physically beyond the logical S3 identifier of the destination of the back-end. Different means for testing the potential cost and governance benefits are being explored, such as complete transfer accounting, retain replica charges, observable placement audit, and explicit consistency semantics. Though the Copy verify switch safeguards you from downtime, it cannot be guaranteed from the available report.

The object storage the study was limited to, however, met only S3 standard, as did the geographical areas of timely public cloud, as well as the actual testbed used in the investigation. It did not attempt to propose that, using server virtualization, we need never traverse the boundaries of workload and storage, or even those of jurisdiction. Adapt and process new rate cards automatically, outages and replication support, sovereign and local-zone support, and petabyte-size deployment will be supported in the future.

Algorithms used to learn or predict placement may have been limited, in which there are no possibilities of placement due to other factors such as sovereign status, retention, or place of residence. The research showed that it's not necessary to offer a range of provider tiers of an object storage service, as we can manage it as a controlled service. This gave flexibility when it comes to autonomy of the storage systems while incorporating compliance, performance, and cost concerns. Cost, signals, and workload evidence were other bricks that compliance is a pre-condition to optimization.

References:

Azumah, K. K., Maciel, P. R., Sørensen, L. T., & S. Kosta. (2023). “Modeling and simulating a process mining-influenced load-balancer for the hybrid cloud,”. *IEEE Transactions on Cloud Computing*, vol. 11, no. 2, pp. 1999–2010, Apr.–Jun. https://ieeexplore.ieee.org/abstract/document/9781255/.

Baron, B., Bousquet, A., Metens, E., Pimpale, S., Puz, N., Sauveur, M. d., et al. (2026). “ACOS: Apple’s geo-distributed object store at exabyte scale,” in. *Proc. 24th USENIX*

Conf. File and Storage Technologies (FAST '26), Santa Clara, CA, USA, pp. 53–66. https://www.usenix.org/system/files/fast26-baron-updated.pdf.

Bonthu, C., Malik, G., & M. R. Dhanagari. (2025). "AI-driven secure and scalable data architecture with multi-domain MDM and cybersecurity for real-time fintech decision-making," in. *Proc. 2025 Tenth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*, pp. 1–7. IEEE. https://ieeexplore.ieee.org/document/11374416.

Brambilla, M., Falconi, M., Fortina, V. M., Plebani, P., & M. Vitali. (2025). "Automating policy-as-code generation pipeline for data products: An OpenAPI-driven Rego generator,". *in Service-Oriented Computing: 23rd International Conference, ICSOC 2025.*, https://re.public.polimi.it/retrieve/6c5801d5-c25d-4965-8dbd-9ed966b705ec/ICSOC25_Policy_Short.pdf.

Carrizales-Espinoza, D., Sanchez-Gallegos, D. D., Gonzalez-Compean, J. L., & J. Carretero. (2024). "StructMesh: A storage framework for serverless computing continuum,". *Future Generation Computer Systems*, vol. 159, pp. 353–369, Oct. 2024. https://www.sciencedirect.com/science/article/pii/S0167739X24002401 .

Dockara, T. R., & M. Malhotra. (2026). "TrustDS: Policy-compiled governance and verifiable evidence for cross-cloud marketplace analytics under explicit security assumptions,". *Scientific Reports*, vol. 16, Art. no. 17958, https://www.nature.com/articles/s41598-026-48740-8_reference.pdf.

E. Ofer. (2021). "Evaluating Cache Performance Using Cloud Storage Traces,". *Storage Developer Conference EMEA, Storage Networking Industry Association (SNIA), June*, https://www.snia.org/educational-library/evaluating-cache-performance-using-cloud-storage-traces-2021.

Eytan, O., Harnik, D., Ofer, E., Friedman, R., & R. Kat. (2020). "It's Time to Revisit LRU vs. FIFO," in . *Proc. 12th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage '20)*, https://www.usenix.org/conference/hotstorage20/presentation/eytan.

Flinn, J., Dou, X., Aggarwal, A., Boyko, A., Richard, F., Sun, E., et al. (2022). "Owl: Scale and flexibility in distribution of hot content," in. *Proc. 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pp. 1–15. https://www.usenix.org/system/files/osdi22-flinn.pdf.

Gefen, I., Zuck, A., Bransky, D., Hershcovitch, M., Harnik, D., & D. Tsafrir. (2025). "Why paying for storage beats free networking in cloud bursting," in. *Proc. 17th ACM Workshop on Hot Topics in Storage and File Systems, Jul*, pp. 31–37. https://dl.acm.org/doi/pdf/10.1145/3736548.3737826.

Goggin, Z., Lekshmanan, A., Valverde, R., & E. Bocchi. (2025). "Ceph at CERN in the multi-datacentre era," . *EPJ Web of Conferences, ,* vol. 337, Art. no. 01331. https://www.epj-conferences.org/articles/epjconf/pdf/2025/22/epjconf_chep2025_01331.pdf.

Hardt, A., Nahhas, A., Müller, H., & K. Turowski. (2025). "Deep reinforcement learning for selecting the optimal hybrid cloud placement combination of standard enterprise IT applications," in. *Proc. International Conference on Enterprise Information Systems*

(ICEIS), vol. 1, 2025, pp. 434–443. https://www.scitepress.org/Papers/2025/132108/132108.pdf.

J. Demšar. (2006). "Statistical Comparisons of Classifiers over Multiple Data Sets," . *Journal of Machine Learning Research*, vol. 7, pp. 1–30. https://www.jmlr.org/beta/papers/v7/demsar06a.html.

Juliussen, B. A., Kozyri, E., Johansen, D., & J. P. Rui. (2023). "The third country problem under the GDPR: Enhancing protection of data transfers with technology,". *International Data Privacy Law*, vol. 13, no. 3, pp. 225–243, Aug. 2023. https://doi.org/10.1093/idpl/ipad013.

Ke, Q., Lv, Y., Zhang, Z., Shen, Z., Yu, Y., Chen, H., et al. (2026). "CrossFS: Improving cross-domain file system performance with CRDT-based metadata synchronization,". *ACM Transactions on Storage,*, vol. 22, no. 2, pp. 1–25, https://dl.acm.org/doi/pdf/10.1145/3777470.

Khan, A. Q., Matskin, M., Prodan, R., Bussler, C., Roman, D., & A. Soylu. (2024). "Cloud storage cost: A taxonomy and survey," . *World Wide Web*, vol. 27, no. 4, Art. no. 36, 2024. https://link.springer.com/content/pdf/10.1007/s11280-024-01273-4.pdf.

Khan, A. Q.; Matskin, M.; Prodan, R.; Bussler, C.; Roman, D.; A. Soylu. (2024). "Cloud storage tier optimization through storage object classification,". *Computing,*, vol. 106, pp. 3389–3418. https://link.springer.com/content/pdf/10.1007/s00607-024-01281-2.pdf.

Liu, S., Mo, X., Hershcovitch, M., Zhang, H., Cheng, A., Girmonsky, G., et al. (2025). "SkyStore: Cost-optimized object storage across regions and clouds," . *arXiv preprint arXiv*, 2502.20818, https://arxiv.org/pdf/2502.20818.

N. M. K. Koneru. (2025). "AI-driven optimization of CI/CD pipelines for multi-cloud infrastructure deployment," in. *Proc. 2025 Tenth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), Nov*, pp. 1–6. IEEE. https://ieeexplore.ieee.org/abstract/document/11374441/.

Nogueira, L., & A. Resende. (2026). "Reusing policy-as-code across CI/CD and Kubernetes admission control: An empirical assessment of governance consistency," . *Computers* , vol. 15, no. 7, Art. no. 453. https://www.mdpi.com/2073-431X/15/7/453.

Pallewatta, S., & M. A. Babar. (2024). "Towards secure management of edge-cloud IoT microservices using policy as code,". *in European Conference on Software Architecture, Cham, Switzerland: Springer Nature Switzerland, Sep.*, pp. 270–287. https://arxiv.org/pdf/2406.18813.

Pang, L., Alazzawe, A., Ray, M., Kant, K., & J. Swift. (2023). "Adaptive intelligent tiering for modern storage systems," . *Performance Evaluation*, vol. 160, Art. no. 102332. https://www.sciencedirect.com/science/article/am/pii/S0166531623000020.

Park, H., Qiu, Z., Ganger, G. R., & Amvrosiadis, G. (2024). "Reducing cross-cloud/region costs with the auto-configuring Macaron cache," in. *Proc. ACM SIGOPS 30th Symposium on Operating Systems Principles*, pp. 347–368. https://dl.acm.org/doi/pdf/10.1145/3694715.3695972.

Poat, M. D., Lauret, J., & T. Rao. (2024). "Federated access from DOE labs to distributed storage in the EIC era of computing," in. *EPJ Web of Conferences*, vol. 295, p. 04021,

May 2024. https://www.epj-conferences.org/articles/epjconf/pdf/2024/05/epjconf_chep2024_04021.pdf.

Rangu, S., Chadha, K. S., & S. Samala. (2025). "AI-powered cloud architecture, agile analytics, and enterprise DevOps-enabled framework for healthcare and insurance ecosystem optimisation," in. *Proc. 2025 Tenth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*, pp. 1–7. IEEE. https://ieeexplore.ieee.org/document/11374414.

Ristov, S., Brandacher, S., Hautz, M., Felderer, M., & R. Breu. (2024). "CODE: Code once, deploy everywhere serverless functions in federated FaaS,". *Future Generation Computer Systems*, vol. 160, pp. 442–456, 2024. https://www.sciencedirect.com/science/article/pii/S0167739X24003170.

Sadavarte, M. V., Reddy, A. K., Moghe, D., Kannan, F. S., Moorjani, S., Kamat, C. D., et al. (2026). U.S. Patent . 12,596,682. https://patents.google.com/patent/US12596682B2/en.

Sardana, J., Hariharan, R., & R. S. Pudi. (2025). "AI-driven secure and resilient healthcare data platforms with intelligent search and personalized communication,". *in Proc. 2025 Tenth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), Nov.* , pp. 1–6. IEEE. https://ieeexplore.ieee.org/document/11374370.

Venkatesh, R. T., Chandrashekar, D. K., Rao, P. B., Sridhar, R., & S. Rajanna. (2023). "Systematic approaches to data placement, replication and migration in heterogeneous edge-cloud computing systems: A comprehensive literature review,". *Ingenierie des Systemes*, vol. 28, no. 3, p. 751, 2023. https://openurl.ebsco.com/contentitem/gcd:167351562?sid=ebsco%3Aplink%3Acrawler-gcd&id=ebsco%3Agcd%3A167351562&crl=c&jrnl=16331311.

Xing, S., & Y. Wang. (2025). "Proactive data placement in heterogeneous storage systems via predictive multi-objective reinforcement learning," . *IEEE Access*, vol. 13, pp. 117986–117998, https://ieeexplore.ieee.org/iel8/6287639/6514899/11072103.pdf.

Zhu, Y., Dhakal, A., Bruel, P., Rattihalli, G., Xiao, Y., Lombardi, J., et al. (2025). "An RDMA-first object storage system with SmartNIC offload," in. *Proc. SC'25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1650–1657. https://dl.acm.org/doi/pdf/10.1145/3731599.3767522.