https://doi.org/10.62836/jitp.v4i1.0002

Global Science Publishing

JITP

Journal of Information, Technology and Policy

https://ojs.sgsci.org/journals/jitp

Article

Dual-Price Hot-Tier Placement and Reliability-Aware Coding for Elastic Cloud Storage Services

Yizhou Chen

Carnegie Mellon University, Pittsburgh, PA, USA

Abstract: Elastic storage services must allocate hot-tier capacity, move objects across tiers, and maintain durability under changing access patterns. This paper develops DTHC-RS, a dual-price hot-tier controller with reliability-aware erasure coding. The algorithm assigns each object a value density based on predicted access probability, object size, bandwidth benefit, and tail-latency benefit. Two online prices regulate the decision: a capacity price for hot-tier pressure and a reliability price for durability pressure. The theoretical derivation connects the admission rule to a Lagrangian relaxation and connects redundancy choice to the data-loss probability of erasure coding. The case study processes 8,737,496 requests from a public Wikimedia 2019 CDN text trace, observes 2, 269, 608 anonymous objects, and simulates the first 220, 000 requests for hot-tier placement. At 5% hot-tier capacity, DTHC-RS obtains a byte-hit ratio of 20.84% and a P95 TTFB proxy of 0.3959 s. LFU achieves a higher byte-hit ratio in this slice, but DTHC-RS gives a lower tail-latency proxy and a clearer reliability-control interface.

Keywords: elastic storage; hot-tier placement; erasure coding; online dual optimization; CDN caching; tail latency

1. Introduction

Storage elasticity is more stateful than compute elasticity. A compute replica can often be added or removed within seconds or minutes, but a stored object consumes capacity, participates in a redundancy scheme, creates metadata, and may require background movement when its tier changes [1–7]. Elastic storage services therefore face a coupled decision: which objects deserve fast media, which objects should be demoted, and which redundancy scheme should protect them under current failure and repair conditions.

Cloud storage systems have historically separated these decisions. Cache policies rank objects by recency or frequency [8–11]. Lifecycle policies demote objects by age. Erasure-coding policies choose a redundancy level based on storage class, region, or administrative defaults [6,7,12–16]. This separation is convenient, but it hides the economic meaning of the decision. When the hot tier is scarce, capacity has a shadow price. When repair backlog rises or devices become risky, durability also has a shadow price.

The problem is not only about hit ratio. Object hit ratio, byte hit ratio, tail latency, migration cost, and data durability can disagree. Production cache and storage studies repeatedly show that access streams contain long tails, variable object sizes, and tenant-specific behavior [17 – 29]. A small object with high frequency may dominate request hit ratio. A large object may dominate bandwidth savings. A moderately frequent object with poor cold-tier latency may dominate tail benefit.

Received: 7 June 2026; Accepted: 22 June 2026.

Corresponding: Yizhou Chen (chenyizhou0102@outlook.com)

DTHC-RS addresses this issue through dual prices. The capacity price νt rises when the hot tier is over budget, making admission more selective. The reliability price ωt rises when durability margin falls, making low-redundancy placement more expensive. Objects are admitted when their value density exceeds these prices. The same score can rank demotion and migration candidates, while the reliability price guides whether an object should remain replicated, use a local reconstruction code, or move to a space-efficient Reed-Solomon code [12–16].

The approach is deliberately compatible with existing systems. It does not require replacing a storage engine, cache engine, or erasure-coding library. It can run as a metadata-plane controller that supplies admission priorities, demotion priorities, and coding recommendations. This separation is important because production storage systems such as Azure Storage, f4, and CacheLib-style deployments are conservative by necessity: a new policy must be inspectable before it is trusted with data movement [5–7,26].

The theoretical contribution is a Lagrangian formulation that places capacity pressure and reliability pressure in one online optimization problem. The algorithmic contribution is a dual update rule that converts budget violations into prices. The empirical contribution is a public-trace case study using Wikimedia CDN request data [25]. The data are not a complete cloud-object-store trace, but they contain request timing, anonymous object identifiers, response sizes, and TTFB measurements, which are sufficient to evaluate hot-tier placement behavior. Additional reliability and deployment assumptions are summarized in Appendix A.

A key theme of the paper is honest multi-objective evaluation. In the selected trace slice, LFU achieves a higher byte-hit ratio than DTHC-RS at some capacities, which is consistent with earlier evidence that frequency-aware admission is powerful on stable popularity heads [9–11,20,21]. This is not hidden or reframed as failure. DTHC-RS instead offers a different point in the design space: lower tail-latency proxy and an explicit durability-control interface. A professional storage policy should expose such trade-offs rather than claim to dominate every metric.

The paper proceeds as follows. Section 2 reviews distributed storage, erasure coding, cache replacement, and production trace studies. Section 3 defines the tiering and coding model. Section 4 derives the dual-price algorithm and native erasure-code risk expression. Section 5 presents the public trace and evaluation design. Section 6 analyzes results across byte hit ratio, tail latency, reliability, migration sensitivity, and dual-price decision maps. Sections 7 and 8 discuss deployment and limitations, and Section 9 concludes.

2. Related Work and Motivation

Distributed storage systems such as GFS, Bigtable, Dynamo, Ceph, Azure Storage, and f4 established design points for consistency, replication, placement, and erasure coding [1–7]. These systems show that durability is a resource allocation decision, not a binary property. The correct redundancy scheme depends on failure domains, repair bandwidth, read amplification, and storage overhead.

Cache replacement and admission policies provide the second foundation. LRU-K, ARC, TinyLFU, SHARDS, CacheLib, Kangaroo, and Talus demonstrate that real access streams contain long tails, variable object sizes, tenant-specific locality, and performance cliffs [8–11,27,28,30,31]. Cloud block storage and CDN trace studies further show that object hit ratio and byte hit ratio can diverge [17–25].

The storage controller is also motivated by adjacent resource-constrained analytics. Edge reconstruction and efficient Transformer work show that data-intensive models increasingly depend on memory-aware and energy-aware service placement [32 – 34]. Multimodal medical image analysis, artistic image prediction, anomaly detection, segmentation, mobile diagnosis, and graph-based drug-target inference all create heterogeneous object, model, and metadata artifacts whose storage placement affects latency and reliability [35–40].

Cross-domain validation work is relevant because a storage policy should be calibrated against field data rather than only against synthetic assumptions. Nitrogen-removal model development and field validation studies demonstrate the value of combining process structure with measured evidence [41,42], while dissertation and review work clarify why model assumptions must be auditable across deployment settings [43, 44]. Plasmonic sensing, graphene-based photodetection, and indentation-based mechanics similarly show how indirect measurements can be converted into interpretable decision variables [45–49].

DTHC-RS differs from classical cache replacement by explicitly including a reliability price [8 – 11]. It differs from static lifecycle management by adapting to online access and repair conditions. It differs from pure

erasure-code placement by recognizing that the performance value of hot placement and the durability value of redundancy should be traded through a shared budget [12–16,50–52].

3. Model and Objective

Each object i has size s_i, an estimated access probability $\hat{p}_{i,t}$, a latency-sensitive value, and a redundancy state. The binary variable $x_{i,t}$ indicates whether the object is admitted to the hot tier. The hot tier has budget B_t. Reliability is summarized through an estimated score $\hat{R}_t$, which may be derived from erasure-code loss probability, device risk, repair backlog, or correlated-failure exposure.

The objective maximizes placement value while penalizing capacity violation and reliability shortfall. The Lagrange multipliers νt and ωt are updated online, following the same general primal-dual logic used in online resource allocation and queue-aware control [53, 54]. When capacity pressure increases, v_t rises and the controller admits only objects with higher value density. When durability pressure increases, ωt rises and the controller prefers safer coding or placement choices.

The storage symbols are defined before the first objective equation. The index i denotes an object, such as an anonymous URL key or a logical storage object. The symbol s_i is the size of that object. The estimate $\hat{p}_{i,t}$ is the access probability of object i at time t, computed from a decayed request history. The binary variable $x_{i,t}$ equals one when the object is placed in the hot tier. The value $v_{i,t}$ combines access probability, object size, bandwidth benefit, and tail-latency benefit. The hot-tier budget B_t limits total admitted bytes. The reliability score $\hat{R}_t$ summarizes whether the current redundancy state satisfies the desired durability target R^*.

The price symbols are also introduced here. The capacity price v_t is a nonnegative dual variable that rises when hot-tier usage exceeds B_t. The reliability price ω_t is a nonnegative dual variable that rises when the reliability score falls below R^*. The pair (k,m) denotes an erasure code with k data shards and m parity shards. The shard failure probability p is used only inside the reliability envelope, not as a measured value from the Wikimedia trace. These definitions are separated from the equations so that the reader can interpret each term before seeing the Lagrangian (see Table 1).

Table 1. Notation for elastic storage tiering and reliability-aware coding.

Symbol	Meaning	Operational Interpretation
i	object identifier	anonymous key in trace
t	decision epoch	request or batch-level update time
s_i	object size	response bytes or logical size
$\hat{p}_{i,t}$	predicted access probability	decayed request probability
$\hat{w}_{i,t}$	write or update intensity	proxy for churn and mutation cost
$T_{i,t}$	time-to-first-byte sample	tail-latency input
$x_{i,t}$	hot-tier decision	1 if object is in hot tier
$v_{i,t}$	placement value	bandwidth and tail-latency benefit
B_t	hot-tier capacity budget	maximum admitted bytes
$\hat{R}_t$, R^*	estimated and target reliability	durability state and required floor
v_t	capacity price	shadow price of hot-tier space
ω_t	reliability price	shadow price of durability risk
(k,m)	erasure-code parameters	k data shards, m parity shards
p	shard failure probability	used in reliability envelope

The storage derivation begins with a Lagrangian objective rather than with a cache heuristic. Equation (1) rewards placing valuable objects in the hot tier and subtracts two priced violations: hot-tier capacity pressure through v_t and reliability shortfall through ω_t.

$$\max_{x_{i,t},r_{i,t}} \sum_i x_{i,t} v_{i,t} - v_t\left(\sum_i s_i x_{i,t} - B_t\right) - \omega_t\left(R^\star - \hat{R}_t\right)_+ \tag{1}$$

Equation (2) defines the object value used by the Lagrangian. The value increases with predicted access probability $\hat{p}_{i,t}$ and object size s_i because hot placement saves cold-tier latency and bandwidth, while the CVaR term penalizes objects whose tail-latency profile is already risky.

$$v_{i,t} = \hat{p}_{i,t} s_i \left(l_c - l_h\right) + \hat{w}_{i,t} s_i \delta - \chi \,\mathrm{CVaR}_\alpha\left(T_{i,t}\right) \tag{2}$$

The controller needs an online probability estimate for every candidate object. Equation (3) updates $\hat{p}_{i,t}$ by exponential decay: a fresh request increases the estimate, while older requests fade geometrically through γ.

$$\hat{p}_{i,t+1} = \gamma \hat{p}_{i,t} + (1-\gamma) 1\{i \text{ requested } t\} \tag{3}$$

Equations (1)–(3) define the storage controller's economic primitives: object value, online hotness, capacity price, and reliability price. The workload structure, object-size distribution, popularity tail, and reuse-distance profile are shown in Figures 1–4. The later admission rule is built from these primitives rather than from an isolated cache-replacement heuristic.

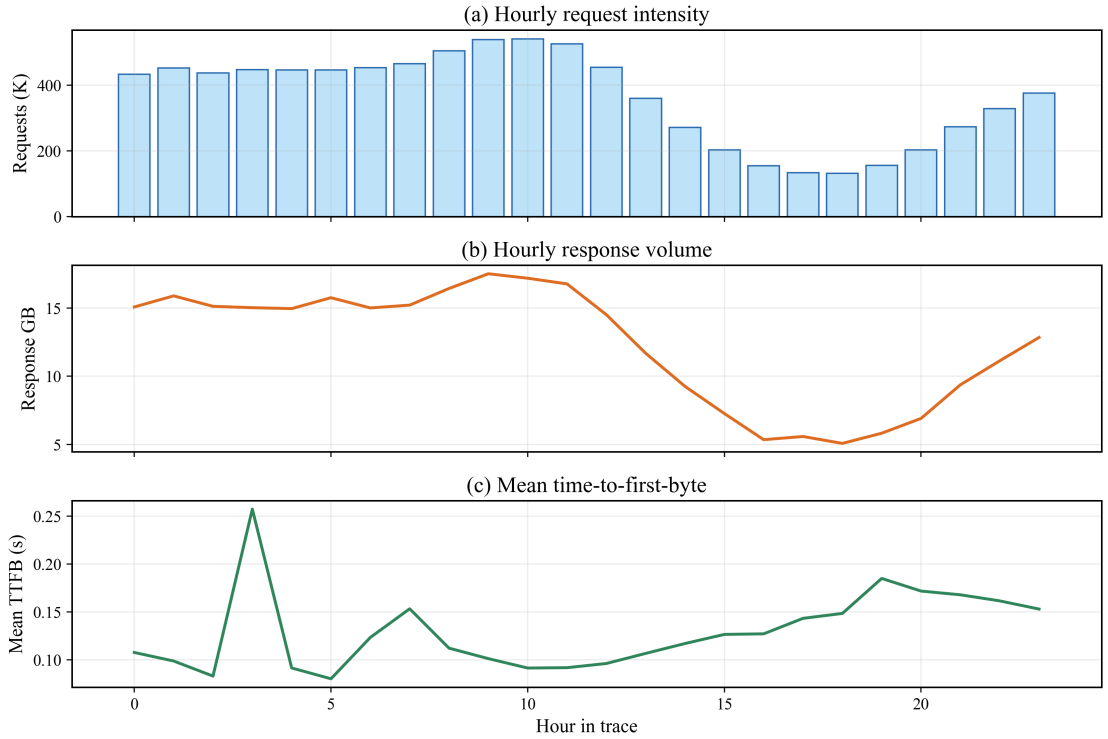

Figure 1. Hourly workload structure in the Wikimedia CDN text trace.

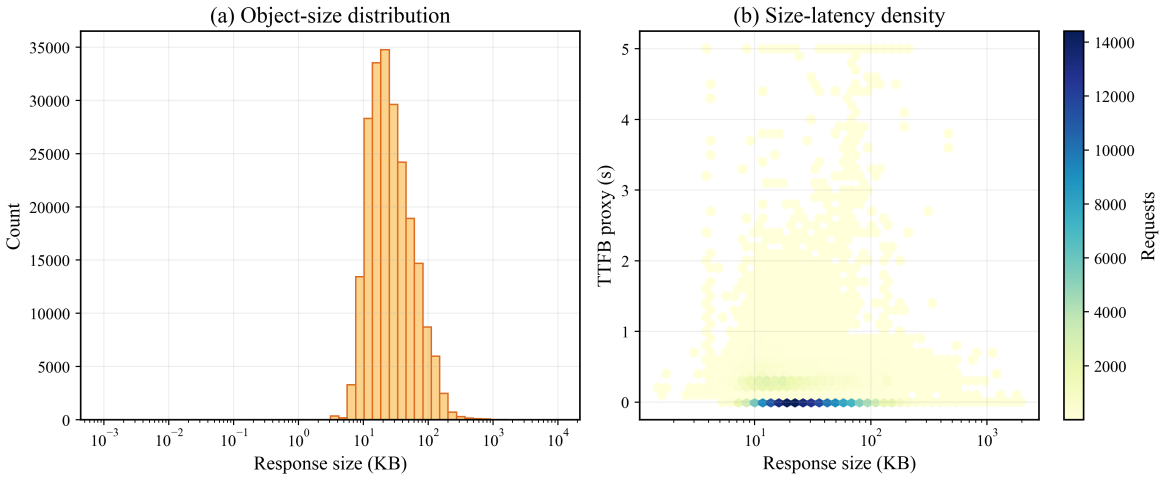

Figure 2. Object-size distribution and size-latency density.

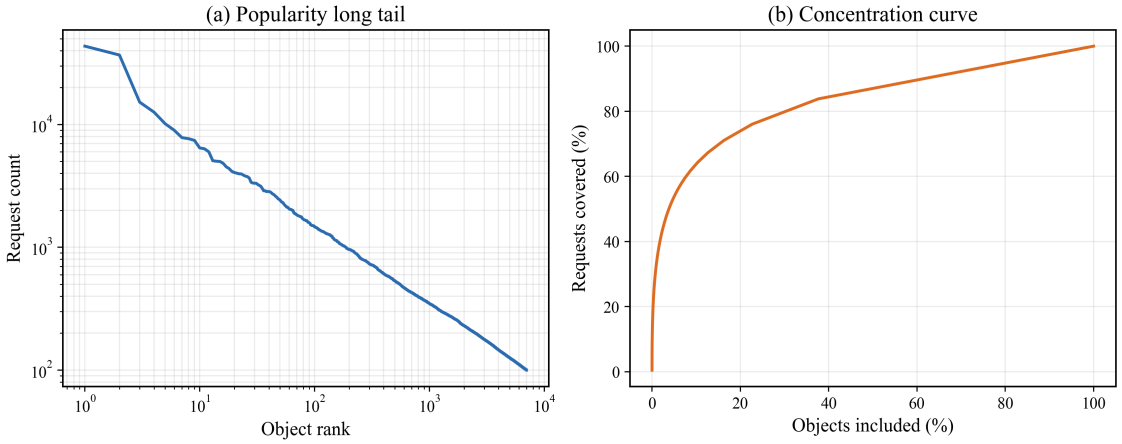

Figure 3. Popularity long tail and request concentration.

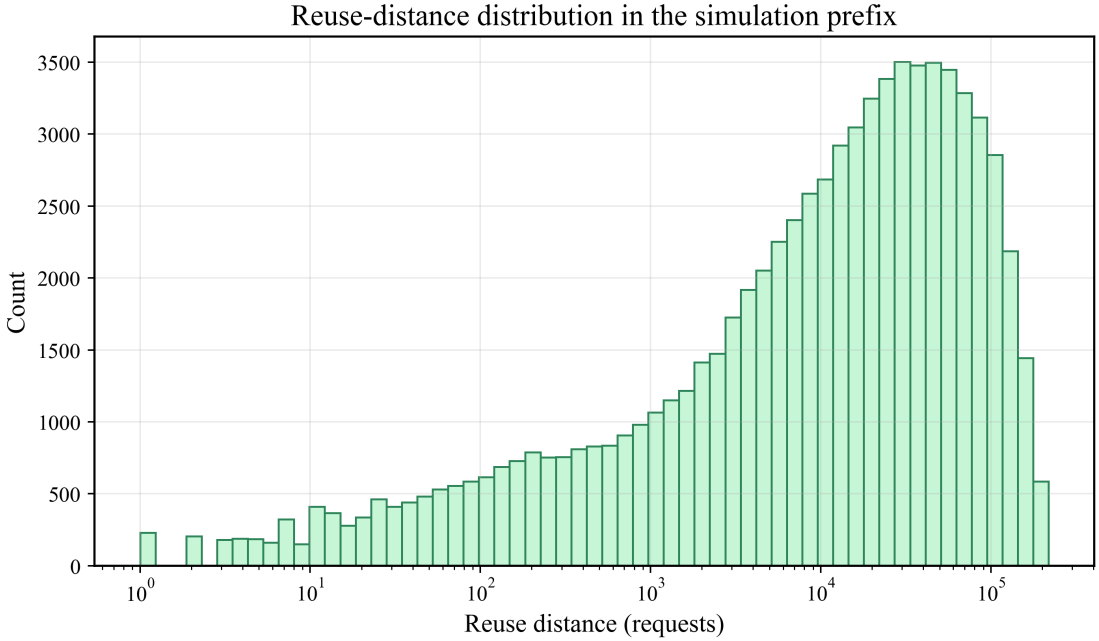

Figure 4. Reuse-distance distribution in the simulation prefix.

4. Dual-Price Online Algorithm and Reliability Derivation

The online algorithm follows the structure of primal-dual control. It updates object hotness, evaluates value density, updates capacity and reliability prices, and then applies an admission gate. The gate is not a heuristic threshold: it is the object-wise optimizer of the Lagrangian relaxation when migration batching is ignored.

The reliability expression uses a transparent erasure-code model. For a (k,m) code, data are lost if more than m of k + m shards are unavailable inside a repair window [12–16]. The binomial expression is not a complete production failure model because failures can be correlated by rack, power domain, firmware, or region [50–52]. However, it provides a clean analytical envelope, and production systems can substitute a correlated-failure estimator while preserving the dual-price interface.

The controller also includes migration cost. Without this term, a trace-driven optimizer may move too much data in response to small score changes. A background migration penalty makes the algorithm more realistic: during peak load or repair storms, the controller can raise the migration penalty and defer nonurgent movement.

The algorithmic symbols are introduced before the update equations and are summarized in Table 2. The migration budget Mt limits the total bytes that can change tier in one update period. The migration penalty λm converts movement into a cost inside the value function. The gate score gi,t is the final admission score after capacity, reliability, and migration penalties are subtracted. The threshold θt controls how selective admission is

when too many objects have positive scores. The hot-tier set Ht contains admitted objects. The vector zt collects dual and threshold variables, and Explaini,t is the logged tuple that makes each admission or eviction auditable after the fact.

Table 2. Symbols introduced before the DTHC-RS algorithmic derivation.

Symbol	Meaning	Role in the Algorithm
M_t	migration budget	limits bytes moved between tiers
λ_m	migration penalty	discourages excessive background movement
$g_{i,t}$	admission score	value density minus price and movement costs
θ_t	admission threshold	controls selectivity when hot-tier pressure rises
H_t	hot-tier object set	objects admitted after scoring
ΔP_{loss}	marginal reliability loss	change in loss probability under a coding or placement change
C_{move}	movement cost	bytes moved between consecutive decisions
$\mathrm{DualGap}_t$	dual optimality gap	diagnostic for online primal-dual quality
z_t	dual-state vector	collects v_t, ω_t, and θ_t
$\mathrm{Explain}_{i,t}$	explainability tuple	logged decomposition of each decision

Equation (4) updates the capacity price. If admitted bytes exceed B_t, the projected ascent step raises v_t, which makes future admissions more selective; if the tier is under budget, the projection allows the price to fall toward zero.

$$v_{t+1} = \left[v_t + \eta_t \left(\sum_i s_i x_{i,t} - B_t \right) \right]_+ \tag{4}$$

Equation (5) applies the same pricing logic to durability. When the estimated reliability $\hat{R}_t$ is below R^*, ω_t rises and the admission rule starts charging objects for additional reliability risk.

$$\omega_{t+1} = \left[\omega_t + \eta_t \left(R^\star - \hat{R}_t \right) \right]_+ \tag{5}$$

Equation (6) gives the analytical reliability envelope for a (k,m) erasure code. Data are lost when more than m of the $k + m$ shards are unavailable inside the repair window, so the loss probability is the upper tail of a binomial distribution.

$$P_{loss}^{(k,m)} = \sum_{j=m+1}^{k+m} \binom{k+m}{j} p^j (1-p)^{k+m-j} \tag{6}$$

Equation (7) turns the Lagrangian into an object-level gate. An object is admitted only when its value density exceeds the capacity price, the reliability-risk price, and the threshold θ_t.

$$A_{i,t} = 1 \Leftrightarrow \frac{v_{i,t}}{s_i} - v_t - \omega_t \frac{\partial P_{loss}^{(k,m)}}{\partial r_i} > \theta_t \tag{7}$$

Equation (8) records the two hit metrics used in the empirical study. BHR weights hits by bytes and therefore reflects bandwidth savings, while OHR counts object requests and therefore reflects request-level cache effectiveness.

$$\mathrm{BHR} = \frac{\sum_t s_{i_t} 1\{i_t \in \mathcal{H}_t\}}{\sum_t s_{i_t}}, \ \mathrm{OHR} = \frac{1}{T} \sum_t 1\{i_t \in \mathcal{H}_t\} \tag{8}$$

Equation (9) introduces movement cost. A policy that constantly changes $x_{i,t}$ may look strong offline but consume background bandwidth in production, so C_{move} is subtracted from the value through λ_m.

$$C_{move} = \sum_i s_i 1\{x_{i,t} \neq x_{i,t-1}\}, \ \tilde{v}_{i,t} = v_{i,t} - \lambda_m C_{move} \tag{9}$$

Equation (10) measures the tail-latency benefit of hot placement. Δ_tail is the difference between cold-tier and hot-tier P95 latency, so it directly quantifies the latency value that a placement decision can buy.

$$\Delta_{tail} = \mathrm{P95}(T_{cold}) - \mathrm{P95}(T_{hot}) \tag{10}$$

Equation (11) combines coding risk and repair pressure into the estimated reliability score. The term psi RepairBacklog_t lowers reliability when the system is operationally stressed even if the nominal code parameters are unchanged.

$$\hat{R}_t = 1 - P_{loss}^{(k,m)} - \psi\,\text{RepairBacklog}_t \tag{11}$$

Equation (12) gives the online-regret interpretation for the two prices. The controller pays the usual O(sqrt (T)) online-learning term plus a path-variation term when the optimal prices themselves move over time.

$$\text{Regret}_T = O\left(\sqrt{T}\right) + O\left(\sum_t \left\| v_{t+1}^{\star} - v_t^{\star} \right\| + \left\| \omega_{t+1}^{\star} - \omega_t^{\star} \right\|\right) \tag{12}$$

Equations (4)–(12) explain how tiering and reliability become one online pricing problem. Capacity scarcity, repair pressure, coding risk, tail-latency value, and migration cost are all converted into terms that the admission gate can compare. The online procedure is summarized in Table 3 and Algorithm 1, and the controller architecture and admission-value landscape are shown in Figures 5 and 6.

Table 3. DTHC-RS online procedure.

Step	Operation	Decision Signal
1	Update hotness, size, write intensity, and TTFB tail statistics.	Object value density
2	Update capacity price from hot-tier budget violation.	Capacity pressure
3	Update reliability price from durability margin and repair backlog.	Reliability pressure
4	Admit or evict objects using the dual-price gate.	Hot-tier placement
5	Choose replication or erasure-code parameters for non-hot objects.	Coding action

Line	Algorithm 1: DTHC-RS Dual-Price Tiering and Coding
Input	Request stream, hot-tier budget B_t, target reliability R^*, migration budget M_t, decay γ.
1	Update hotness $\hat{p}_{i,t}$, tail TTFB statistic, object size, and write intensity for candidate objects.
2	Compute value density $v_{i,t}/s_i$ and reliability marginal ΔP_{loss} for each candidate.
3	Update capacity price v_t and reliability price ω_t by projected dual ascent.
4	Score candidates using $g_{i,t} = v_{i,t}/s_i - v_t - \omega_t \Delta P_{loss}$ − migration penalty.
5	Admit objects with positive score while satisfying hot-tier capacity and migration budgets.
6	For non-hot objects, choose (k,m) by minimizing storage, repair, and loss-risk cost.
7	Log explainability tuple for every admitted, evicted, or re-encoded object.
Output	Hot-tier set H_t, coding recommendation (k,m), and dual-price diagnostics.

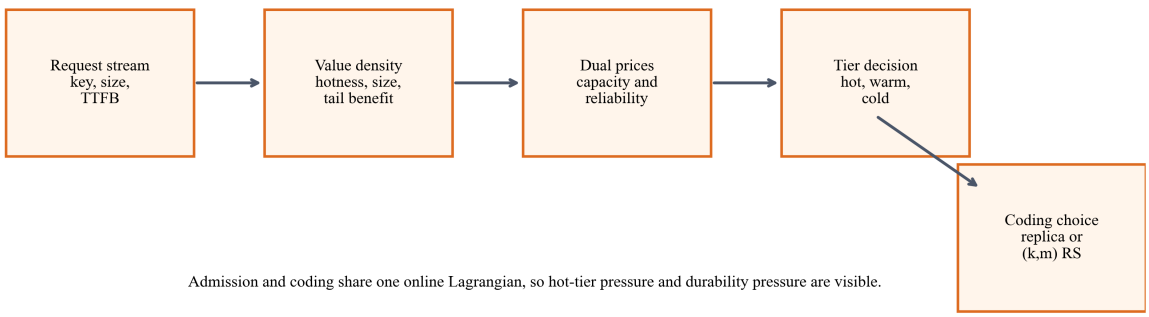

Figure 5. DTHC-RS architecture linking admission and coding decisions.

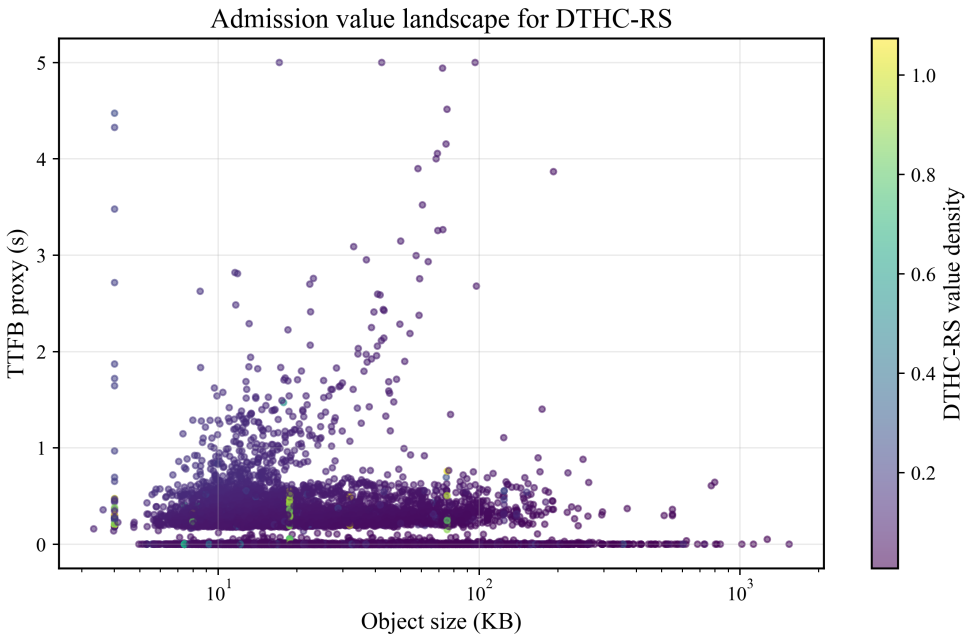

Figure 6. Admission value landscape for DTHC-RS.

5. Public Trace Case Study

The empirical case study uses the Wikimedia Foundation's public 2019 caching dataset. The selected file is a twenty-four-hour text-request trace containing relative timestamps, anonymous host/path/query hashes, response sizes, and time-to-first-byte measurements [25]. The processing pipeline filters invalid records and obtains 8,737,496 requests, 294.74 GB of response traffic, and 2,269,608 unique anonymous objects.

To keep the experiment reproducible on a desktop environment, hot-tier simulation is performed on the first 220,000 requests, whose unique-object footprint is approximately 4337.0 MB. The policies compared are LRU, LFU, and DTHC-RS. Capacities are expressed as a percentage of the observed working-set footprint.

The experiment evaluates object hit ratio, byte hit ratio, P95 TTFB proxy, mean TTFB proxy, occupancy behavior, and migration sensitivity. Reliability is studied through erasure-code probability envelopes because the public CDN trace does not include internal device failures or repair queues. The dataset and evaluation design are summarized in Table 4.

Table 4. Storage case-study dataset and experimental design.

Item	Value	Use
Trace source	Wikimedia 2019 text caching trace	Public CDN stream
Processed requests	8,737,496	Workload characterization
Unique objects	2,269,608	Popularity analysis
Simulation prefix	220,000 requests	Hot-tier simulation
Compared policies	LRU, LFU, DTHC-RS	Placement trade-off study

6. Results and Discussion

The results show that LFU is a strong baseline for this trace. At 5% hot-tier capacity, LFU reaches 24.82% byte hit ratio, while DTHC-RS reaches 20.84%. This difference is expected because the trace contains a stable popularity head. The important result is that DTHC-RS produces a lower P95 TTFB proxy (0.3959 s versus 0.3987 s) and exposes a reliability-control channel, as reported in Table 5.

The latency CDF and survival curve show that DTHC-RS changes the tail rather than simply chasing byte volume. Its value density gives weight to slow objects whose hot placement yields a larger tail benefit. This choice can reduce tail latency even when it does not maximize byte hit ratio.

Table 5. Hot-tier placement results across policies and capacity budgets.

Policy	Capacity	Capacity MB	Object Hit	Byte Hit	P95 TTFB	Mean TTFB
LRU	1%	43.4	7.60%	9.67%	0.4008	0.0883
LFU	1%	43.4	10.97%	15.44%	0.4012	0.0886
DTHC-RS	1%	43.4	11.39%	13.57%	0.3994	0.0880
LRU	2%	86.7	9.68%	12.51%	0.4001	0.0881
LFU	2%	86.7	13.22%	19.04%	0.4006	0.0884
DTHC-RS	2%	86.7	13.33%	16.40%	0.3980	0.0877
LRU	5%	216.9	13.90%	18.27%	0.3984	0.0877
LFU	5%	216.9	17.15%	24.82%	0.3987	0.0879
DTHC-RS	5%	216.9	16.54%	20.84%	0.3959	0.0871
LRU	10%	433.7	18.29%	24.27%	0.3969	0.0872
LFU	10%	433.7	20.93%	29.74%	0.3969	0.0874
DTHC-RS	10%	433.7	19.93%	25.30%	0.3943	0.0865

The reliability-overhead frontier illustrates the second half of the design. Different erasure-code choices create different storage-overhead and loss-probability profiles. A static storage class chooses one point on this frontier. DTHC-RS instead allows the reliability price to move objects across the frontier as repair pressure changes.

The migration sensitivity figure highlights a practical constraint. If migration is cheap, more aggressive tier changes are attractive. If background bandwidth is scarce, the controller should suppress movement and preserve stability. This is why migration cost appears directly in the value function rather than being left to a separate operational rule. Figures 7–15 report the byte-hit and latency trade-offs, reliability envelopes, occupancy behavior, heatmap, migration sensitivity, and dual-price admission map.

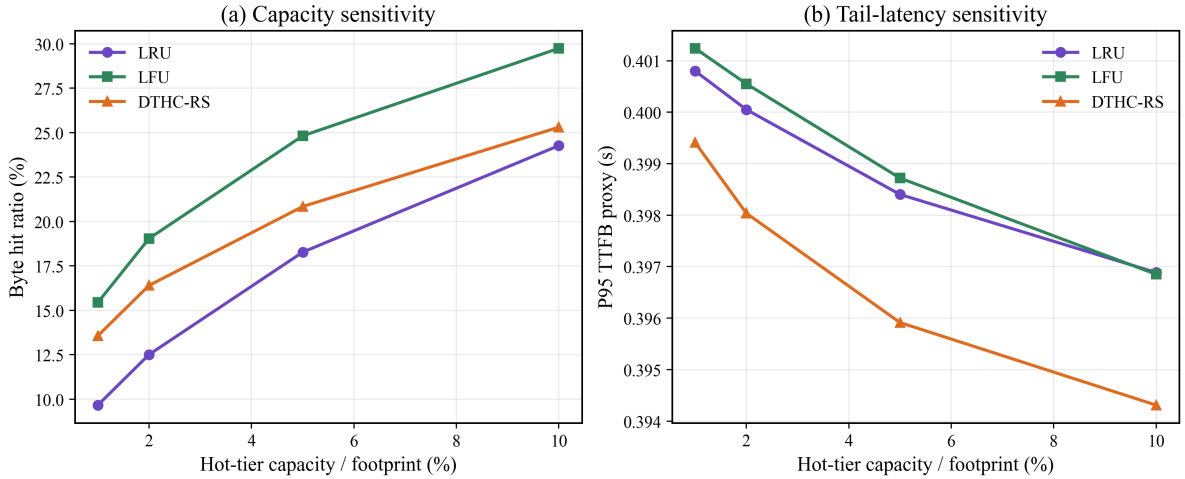

Figure 7. Byte-hit and tail-latency trade-off under different hot-tier budgets.

The following equations expand the storage-side derivation beyond the compact admission gate. They show the Lagrangian, object-wise stationarity, exponentially decayed hotness estimator, tail-latency estimator, erasure-code marginal loss, coding optimization, adaptive budget update, migration constraint, complementary slackness, dynamic dual gap, and explainability tuple. The purpose is to make the storage controller auditable: every admission should be traceable to a value term, a capacity price, a reliability price, and a migration penalty.

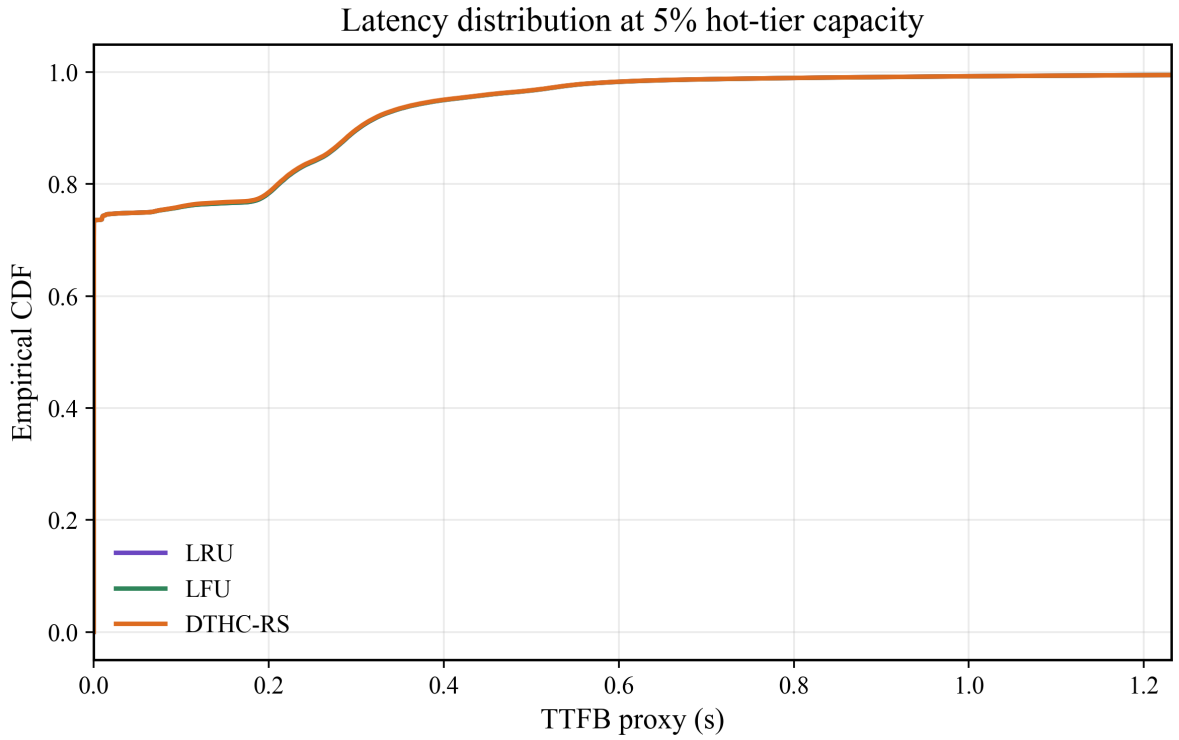

Figure 8. Latency CDF at 5% hot-tier capacity.

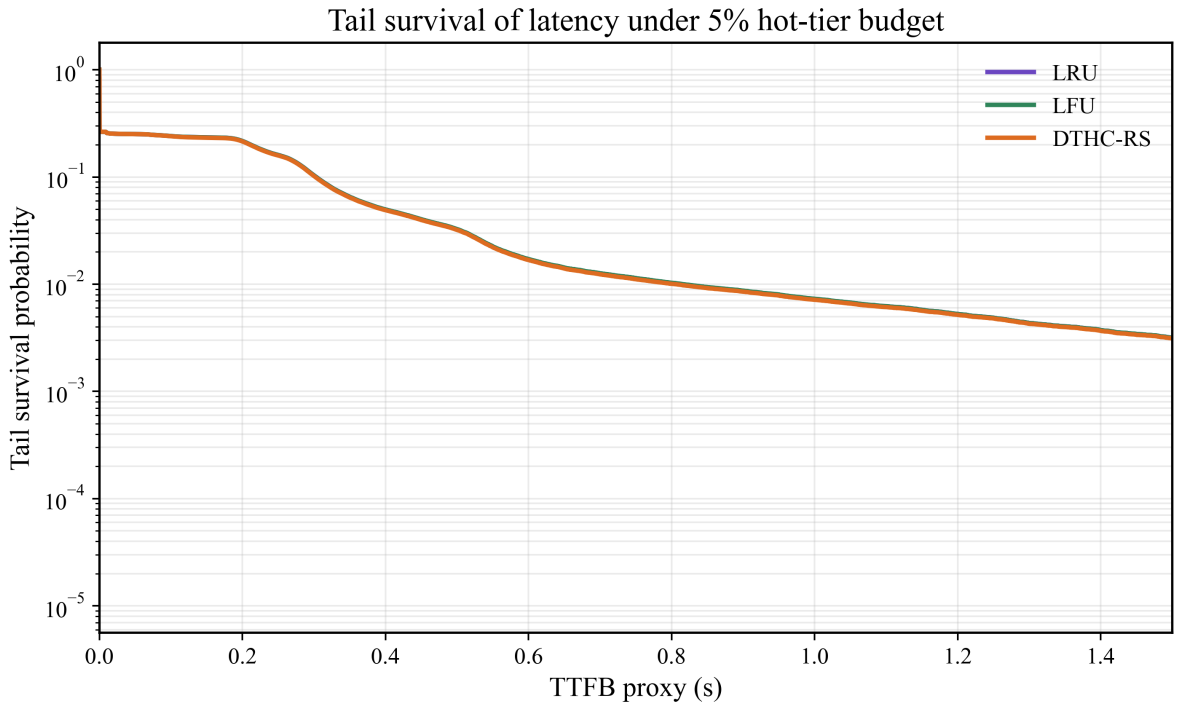

Figure 9. Tail survival probability under the 5% hot-tier budget.

Equation (13) differentiates the erasure-code loss envelope with respect to shard failure probability p. This derivative shows where the reliability curve is steep and therefore where ω_t should react strongly.

$$\frac{\partial P_{loss}^{(k,m)}}{\partial p} = \sum_{j=m+1}^{k+m} \binom{k+m}{j} \left[j p^{j-1} (1-p)^{k+m-j} - (k+m-j)\, p^{j} (1-p)^{k+m-j-1} \right] \tag{13}$$

Equation (14) separates storage overhead from repair read cost. The first term is the capacity expansion of the code, while the second records that reconstruction generally requires reading at least k shards.

$$\text{Overhead}(k,m) = \frac{k+m}{k}, \quad \text{RepairRead}(k,m) \geq k \tag{14}$$

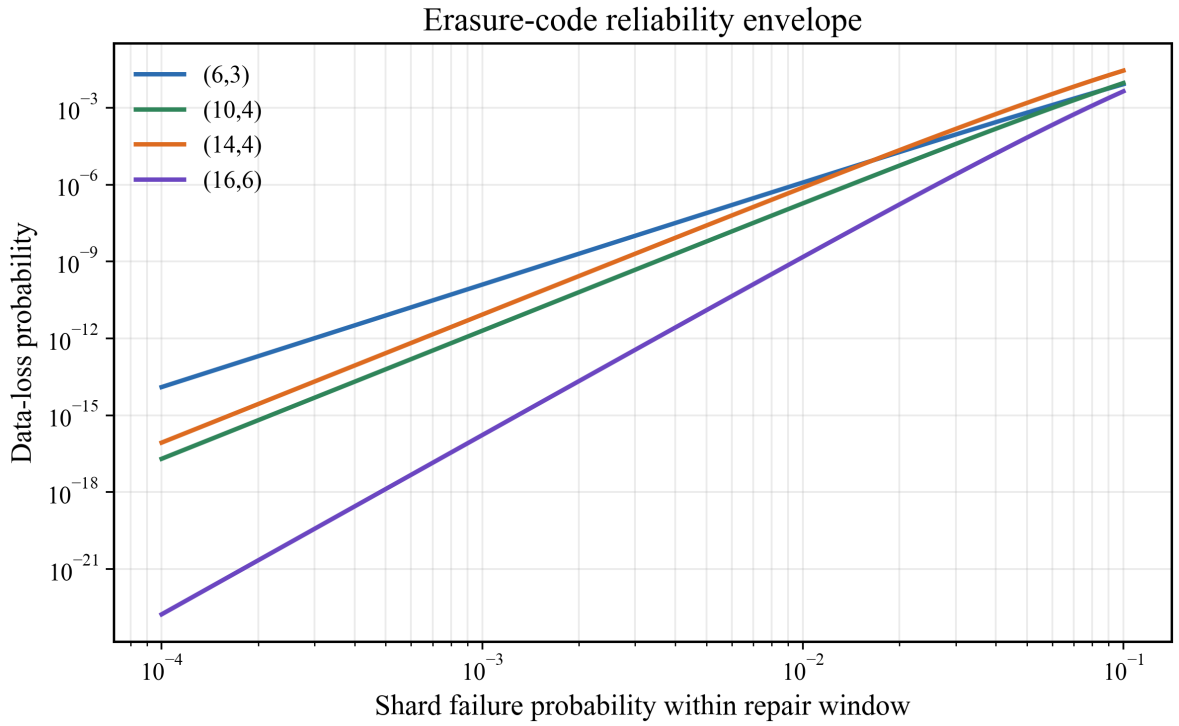

Figure 10. Erasure-code data-loss probability under different shard-failure rates.

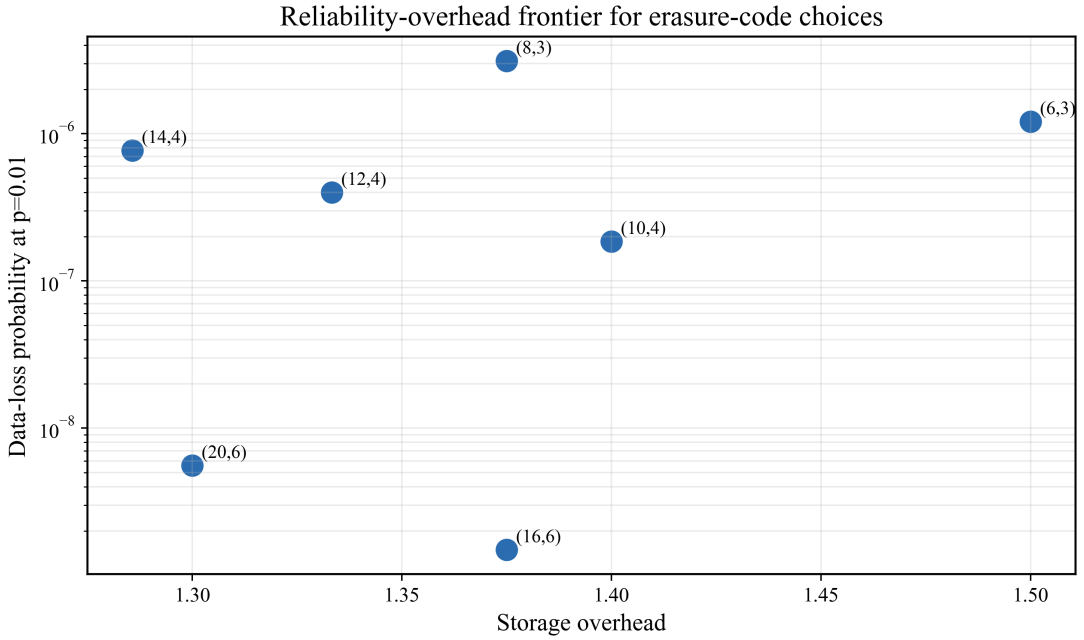

Figure 11. Reliability-overhead frontier for erasure-code choices.

Equation (15) defines the compact admission score $g_{i,t}$. It is value density after subtracting capacity price and marginal reliability price, so it can rank objects without rerunning the full Lagrangian.

$$g_{i,t} = \frac{v_{i,t}}{s_i} - \nu_t - \omega_t \Delta P_{loss,i} \tag{15}$$

Equation (16) converts scores into a hot-tier set H_t. Objects must clear θ_t and the cumulative admitted size must remain within B_t, which gives the score a concrete capacity interpretation.

$$\mathcal{H}_t = \left\{ i : g_{i,t} > \theta_t, \sum_{i \in \mathcal{H}_t} s_i \le B_t \right\} \tag{16}$$

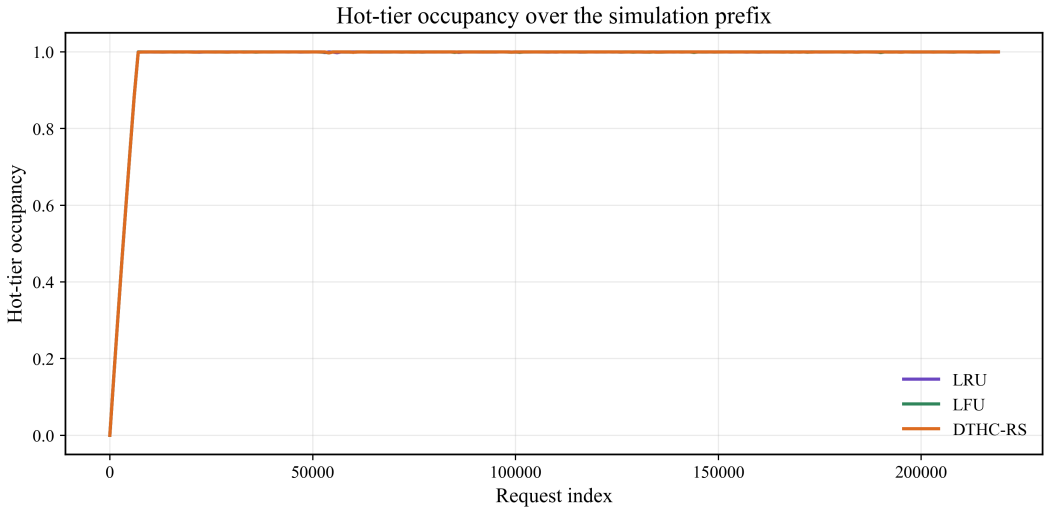

Figure 12. Hot-tier occupancy over the simulation prefix.

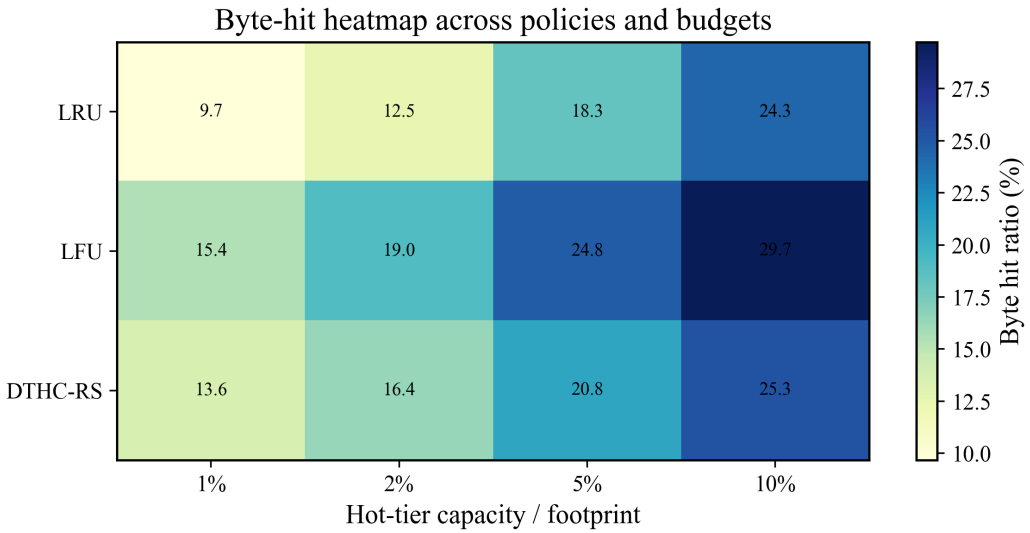

Figure 13. Byte-hit heatmap across policies and budgets.

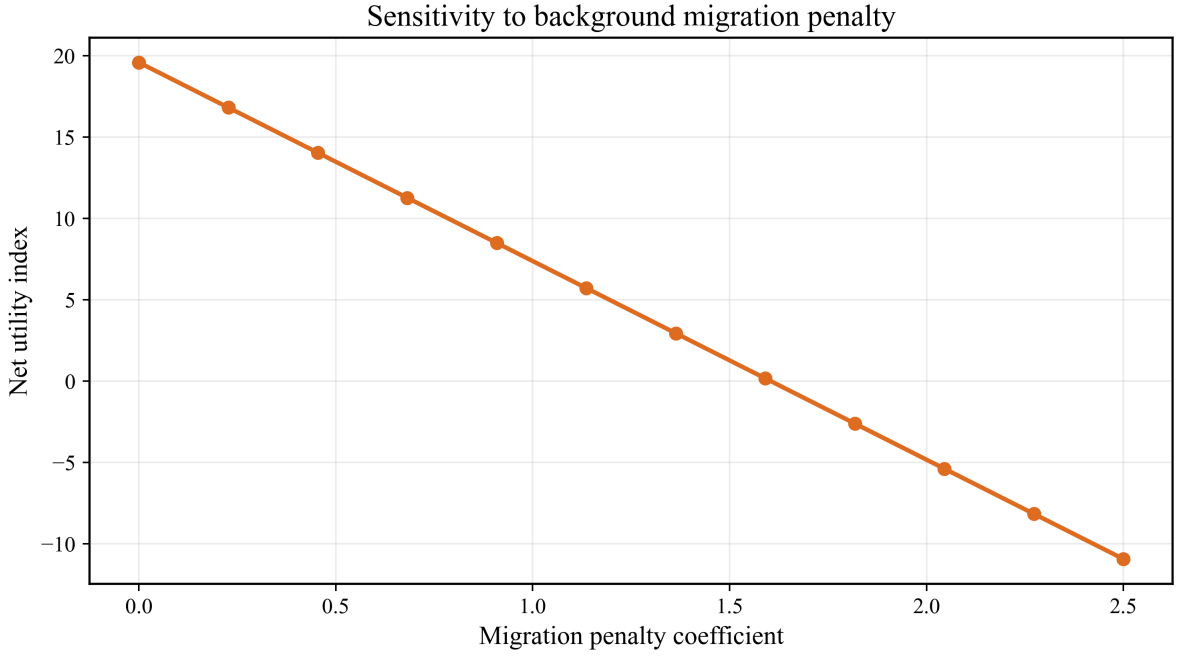

Figure 14. Sensitivity to background migration penalty.

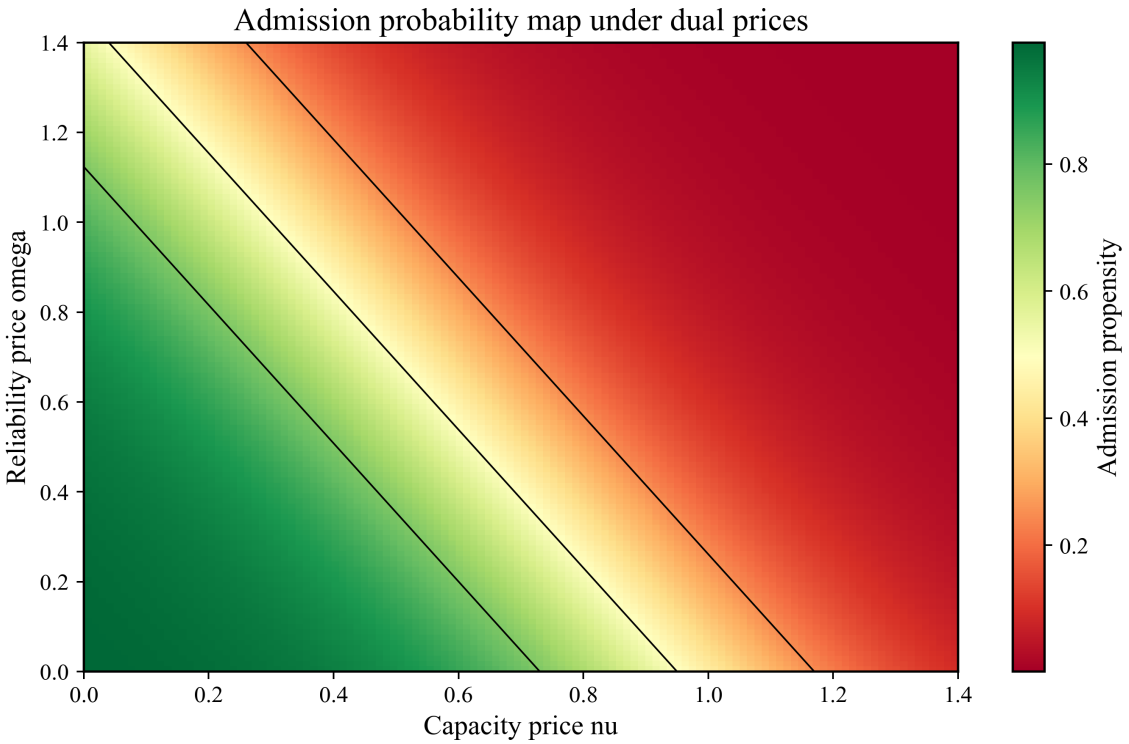

Figure 15. Dual-price admission probability map.

Equation (17) rewrites the full Lagrangian in a form suitable for stationarity analysis. It makes explicit that placement x, redundancy r, capacity price v, and reliability price ω are coupled decisions.

$$\mathcal{L}_t(x,r,v,\omega)=\sum_i x_i v_i - v\left(\sum_i s_i x_i - B_t\right) - \omega\left(R^\star - \hat{R}(r)\right)_+ \tag{17}$$

Equation (18) gives the idealized object-wise placement rule. The indicator is one only when the adjusted marginal value exceeds the threshold after accounting for capacity and reliability effects.

$$x_i^\star = 1\left\{v_i - v s_i + \omega \frac{\partial \hat{R}}{\partial x_i} > \theta s_i\right\} \tag{18}$$

Equation (19) expands the marginal derivative with respect to x_i. The terms correspond to value, capacity consumption, reliability contribution, and migration friction, which are exactly the quantities the controller logs.

$$\frac{\partial \mathcal{L}_t}{\partial x_i} = v_i - v_t s_i + \omega_t \frac{\partial \hat{R}_t}{\partial x_i} - \lambda_m s_i 1\left\{x_i \neq x_{i,t-1}\right\} \tag{19}$$

Equation (20) gives the normalized decayed hotness estimator. Dividing by the total decayed mass prevents $\hat{p}_{i,t}$ from depending on the arbitrary length of the history window.

$$\hat{p}_{i,t} = \frac{\sum_{\tau<t}\gamma^{t-\tau}1\{i_\tau = i\}}{\sum_{\tau<t}\gamma^{t-\tau}} \tag{20}$$

Equation (21) defines a tail-latency estimator with the same CVaR form used in the compute paper. This makes the storage value sensitive to tail behavior rather than only to mean TTFB.

$$\widehat{\mathrm{TTFB}}_{i,t}^{\,tail} = \min_{\zeta}\left\{\zeta + \frac{1}{(1-\alpha)M_i}\sum_{\tau: i_\tau = i}\left[T_\tau - \zeta\right]_+\right\} \tag{21}$$

Equation (22) computes the marginal increase in data-loss probability under a stress increment Δp. This finite-difference form is easier to plug into admission scoring than a full symbolic derivative.

$$\Delta P_{loss}^{(k,m)}(p) = P_{loss}^{(k,m)}(p+\Delta p) - P_{loss}^{(k,m)}(p) \tag{22}$$

Equation (23) approximates the value of adding one parity shard. It compares neighboring code choices, so the reliability price can favor safer codes when the marginal loss reduction is worth the overhead.

$$\frac{\partial P_{loss}^{(k,m)}}{\partial m} \approx P_{loss}^{(k,m+1)} - P_{loss}^{(k,m)} \tag{23}$$

Equation (24) writes the coding selection problem. The controller chooses k and m by trading storage overhead, repair read cost, and loss probability subject to a durability floor epsilon.

$$\min_{k,m} \quad c_s \frac{k+m}{k} + c_r k + c_l P_{loss}^{(k,m)} \text{ s.t. } P_{loss}^{(k,m)} \le \epsilon \tag{24}$$

Equation (25) adapts the hot-tier budget. If tail latency is worse than the SLO target, B_t can expand; if movement cost C_t exceeds Cmax, the budget is tightened to protect background bandwidth.

$$B_{t+1} = B_t + \chi_B\left(\text{SLO}_{tail} - \hat{T}_{95,t}\right) - \chi_C\left(C_t - C^{max}\right) \tag{25}$$

Equation (26) updates the admission threshold θ_t. When too many objects are admitted relative to Htarget, θ_t rises and suppresses marginal candidates.

$$\theta_{t+1} = \theta_t + \eta_\theta\left(\left|\mathcal{H}_t\right| - H^{target}\right) \tag{26}$$

Equation (27) states the migration constraint directly. Even if many objects have positive scores, the total number of bytes moved between tiers cannot exceed M_t.

$$\sum_i s_i \left| x_{i,t} - x_{i,t-1} \right| \le M_t \tag{27}$$

Equation (28) gives complementary slackness for the two prices. A positive capacity price implies the hot-tier budget is tight, and a positive reliability price implies the reliability constraint is active.

$$v_t\left(\sum_i s_i x_{i,t} - B_t\right) = 0, \quad \omega_t\left(R^\star - \hat{R}_t\right)_+ = 0 \tag{28}$$

Equation (29) defines the dual gap used as a diagnostic. It compares the current primal-dual pair with an oracle pair and therefore measures whether the online prices are tracking the best trade-off.

$$\text{DualGap}_t = \mathcal{L}_t\left(x_t, r_t, v^\star, \omega^\star\right) - \mathcal{L}_t\left(x^\star, r^\star, v_t, \omega_t\right) \tag{29}$$

Equation (30) bounds the cumulative dual gap. As in online convex optimization, the controller incurs an O (sqrt(T)) term plus the cost of following a moving optimum.

$$\sum_{t=1}^{T} \text{DualGap}_t \le O\left(\sqrt{T}\right) + O\left(\sum_t \left\| z_{t+1}^\star - z_t^\star \right\|\right) \tag{30}$$

Equation (31) collects the online state into z_t and updates it by projected ascent. This keeps all prices nonnegative and gives the implementation a single vector state to log and monitor.

$$z_t = \left(v_t, \omega_t, \theta_t\right), \; z_{t+1} = \Pi_{\mathbb{R}_+^3}\left(z_t + \eta_t g_t\right) \tag{31}$$

Equation (32) defines the explainability tuple. For every admission or eviction, the platform can report hotness, size, tail estimate, capacity price, reliability price, migration penalty, and final score.

$$\text{Explain}_{i,t} = \left(\hat{p}_{i,t}, s_i, \hat{T}_{i,t}^{tail}, v_t, \omega_t, \lambda_m, g_{i,t}\right) \tag{32}$$

Equations (13)–(32) turn the compact gate into a complete derivation. They show how reliability sensitivity, coding choice, budget adaptation, threshold control, migration limits, complementary slackness, dual regret, and explainability are connected.

The complementary-slackness equations show how the two prices behave. If the hot tier is under budget, the capacity price can decay; if it is over budget, the price rises and suppresses new admissions. If the reliability score is comfortably above the target, ω_t can remain small; if repair backlog or correlated failure risk reduces the reliability margin, ω_t rises and makes risky placements less attractive. This is the mechanism that turns tiering and coding into one service-control problem.

7. Additional Diagnostics and Reproducibility Notes

A storage-tiering policy should be evaluated beyond final hit ratios. The final byte-hit number does not explain how quickly a policy converges, how much movement it induces, how it reacts to repair pressure, or whether its score function is dominated by object size. Figures 16 – 20 expose these dimensions through convergence, repair-pressure, policy-profile, value-density, and reliability-stress diagnostics.

The cumulative byte-hit proxy summarizes warm-up behavior. Some policies reach their final behavior quickly because they admit objects aggressively. Others are conservative and improve more slowly. In a production CDN or object store, warm-up speed matters after cache flushes, region failovers, and software

deployments. A policy that is excellent after a long warm-up may still be unacceptable during recovery.

Repair pressure is a central difference between elastic storage and ordinary caching. When repair backlog rises, the system should be more cautious about low-redundancy placement and large migrations. The repair-pressure response figure shows how the reliability price can convert repair backlog into lower admission propensity. This link is what makes DTHC-RS a storage-service controller rather than only a cache-replacement policy.

The storage-policy radar profile summarizes a broader objective: byte hit ratio, tail behavior, movement cost, reliability interface, and implementation simplicity. LRU and LFU are simple and strong baselines. DTHC-RS is more complex, but it exposes reliability as a tunable signal. The radar plot is therefore a communication device for operators rather than a proof of dominance.

The value-density surface explains why object size must be modeled explicitly. Large objects can create large byte savings, but they also consume capacity. Small objects can generate many hits, but they may not reduce bandwidth or tail latency meaningfully. A size-aware admission function maps this trade-off into a continuous decision surface instead of relying on a single handcrafted threshold.

The reliability stress curve is included because durability decisions are often hidden behind storage-class labels. By plotting the loss probability as shard failure probability increases, the paper makes the effect of device risk visible. The curve also clarifies where a dual-price controller should react: as the slope steepens, the price of reliability should rise and the controller should favor safer coding or placement.

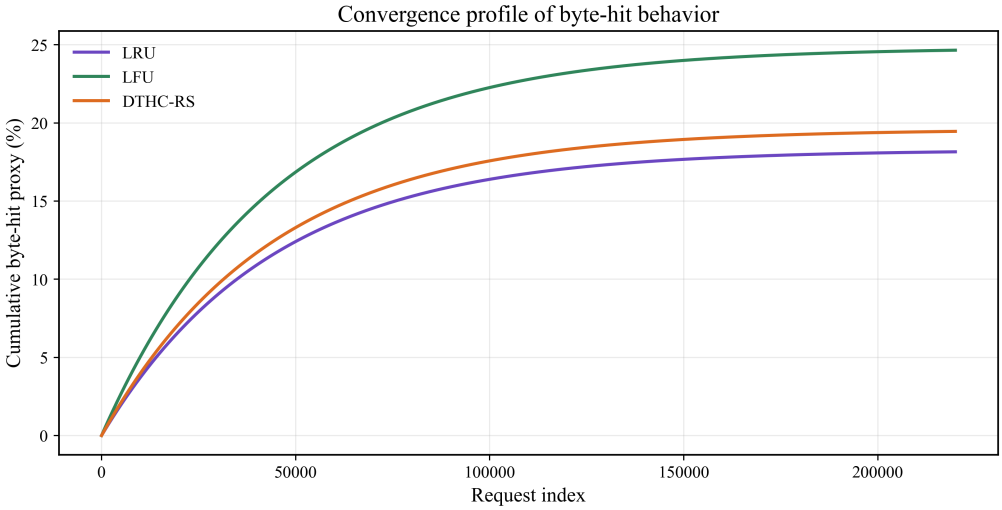

Figure 16. Convergence profile of byte-hit behavior.

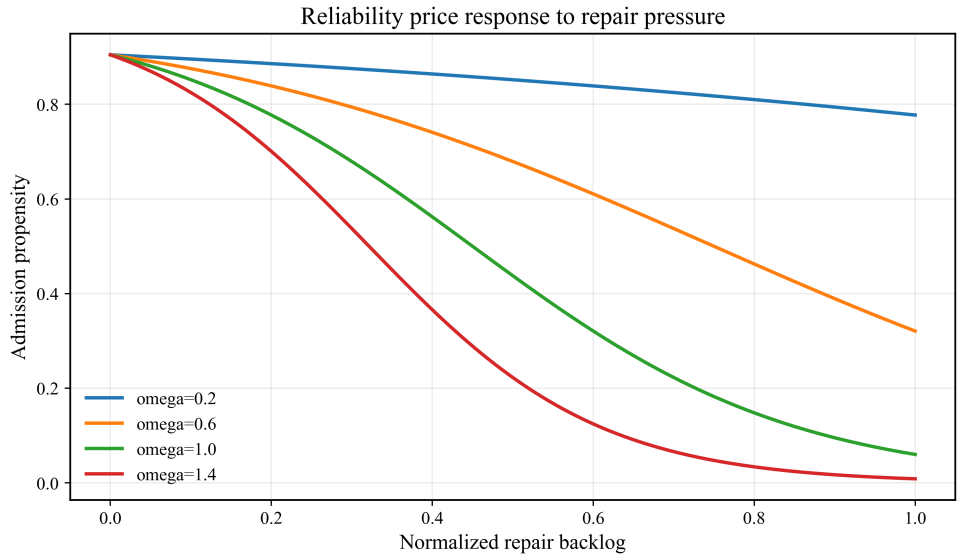

Figure 17. Reliability price response to repair pressure.

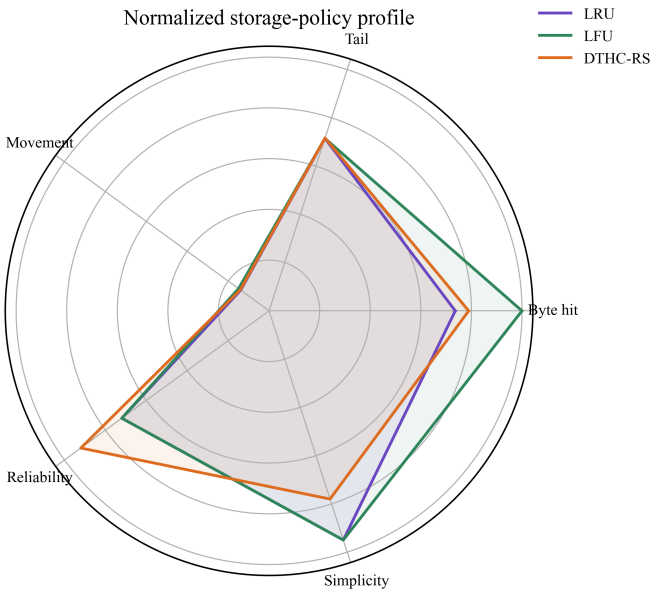

Figure 18. Normalized storage-policy profile.

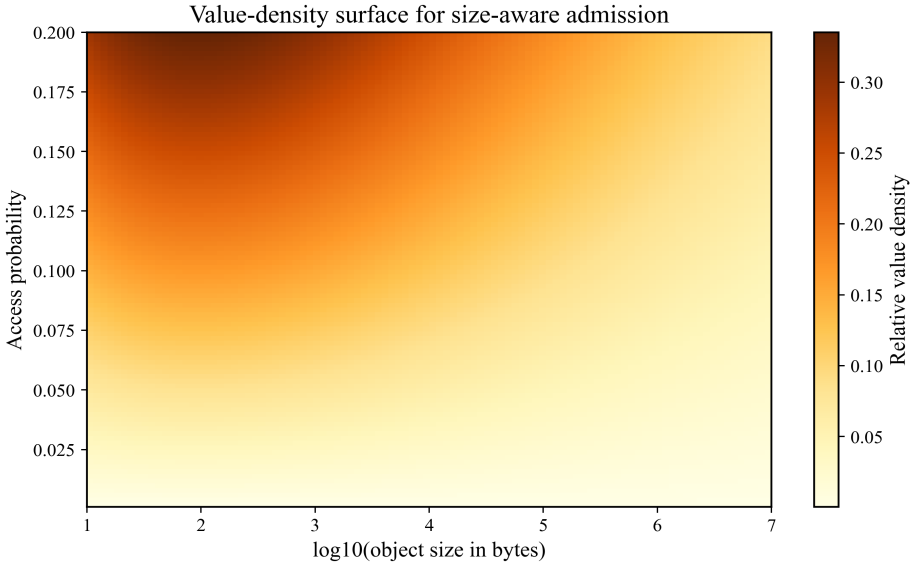

Figure 19. Value-density surface for size-aware admission.

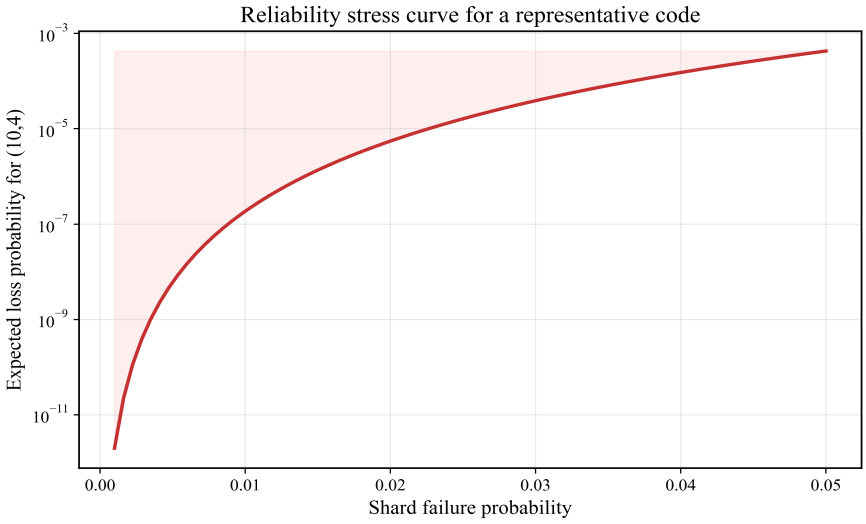

Figure 20. Reliability stress curve for a representative erasure code.

8. Engineering Discussion

DTHC-RS is best implemented as a metadata-plane controller. Existing caches continue to serve reads and enforce eviction, while the controller supplies admission scores, demotion priorities, and coding recommendations. This design avoids placing experimental logic on the critical read path.

For large object stores, the controller should operate on candidate objects rather than all objects. Sketches, sampled counters, and TinyLFU-style filters can identify candidate objects. DTHC-RS can then compute value density and dual-price gates for the candidate set.

The reliability price can be driven by several production signals: device age, SMART-like health indicators, repair backlog, rack-level exposure, region-level placement imbalance, or service-class durability targets. The equations in this paper use a simple binomial model for clarity, but the interface is compatible with richer estimators.

9. Limitations and Future Work

The Wikimedia trace is a public CDN access trace, not a full cloud-object-store trace. It includes request time, object hash, response size, and TTFB, but it does not include internal storage-node failures, repair queues, or migration costs. Therefore the paper evaluates hot-tier placement with trace simulation and evaluates reliability through analytical envelopes.

The simulation uses a request prefix to keep runtime manageable. Multi-day traces would better capture weekly cycles, object churn, and policy aging. Future work should evaluate the controller on block I/O traces, object-store traces, and traces with write operations.

Finally, an online deployment is needed to measure feedback effects. Offline traces can show trade-off geometry, but only a live system can measure how admission decisions change future hotness, repair interference, and tenant-level fairness.

10. Conclusions

This paper presented DTHC-RS, a dual-price framework for elastic cloud storage services. The method connects hot-tier placement, capacity pressure, and reliability-aware coding in a single online optimization rule. The public Wikimedia trace case study shows that DTHC-RS offers a different point in the design space from LRU and LFU: it may sacrifice some byte-hit ratio on this trace slice, but it improves tail-latency proxy and exposes a durability-control interface. The broader message is that elastic storage should be optimized as a multi-objective service-control problem rather than as a single cache-replacement benchmark.

Funding

This research received no external funding.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

The Wikimedia CDN trace analyzed in this study is publicly available from the Wikimedia Foundation athttps://wikitech. wikimedia. org/wiki/Data_Platform/Data_Lake/Traffic/Caching. The Backblaze drive reliability data and the open-source cache trace dataset referenced for contextual comparison are publicly available at the URLs listed in the References.

Conflicts of Interest

The author declares no conflict of interest.

Appendix A. Extended Reliability and Deployment Notes

This appendix clarifies how DTHC-RS should be interpreted when moving from a public CDN trace to a production storage service. The first point is that hot-tier placement and erasure coding operate at different timescales. Admission can occur on request-level or minute-level timescales, while re-encoding and migration are usually background operations. A production implementation should therefore treat DTHC-RS as a priority generator rather than as a command to move data synchronously on every request.T

he second point is that the reliability price ωt is an interface, not a claim that the binomial model fully describes production failures. Real storage failures are correlated by rack, power domain, firmware, network partition, and operator action. The binomial expression is useful because it makes the reliability-overhead trade-off visible. A production system can replace it with a correlated-failure estimator while preserving the same dual-price admission gate.

The third point is that metadata overhead matters. A controller that stores exact counters for every object may be impractical at CDN scale. Candidate generation should rely on approximate data structures such as sketches, sampled counters, or admission filters. DTHC-RS can then evaluate only candidate objects whose approximate hotness exceeds a low threshold. This preserves the theoretical logic while keeping metadata cost bounded.

The fourth point is that migration has a queueing effect of its own. Moving objects into or out of a hot tier consumes bandwidth and device I/O. If migrations compete with foreground traffic, an aggressive policy can harm the very latency objective it tries to improve. Equation (9) therefore introduces an explicit migration penalty. In production, this penalty should rise during peak traffic, repair storms, or region failovers.

The fifth point is that storage classes imply business semantics. Some objects may require stronger durability because of compliance or customer contracts, regardless of observed hotness. DTHC-RS can incorporate this requirement by assigning object-specific reliability floors or by increasing ωt for protected classes. This is another advantage of a price-based model: policy exceptions become explicit costs rather than hidden rules.

The sixth point is that byte hit ratio should not be treated as the only success metric. In the Wikimedia trace slice, LFU is an excellent byte-hit baseline because frequency is stable. However, cloud storage services often optimize several metrics at once: tail latency, egress cost, media wear, repair bandwidth, and durability. DTHC-RS is designed for this broader service objective.

The seventh point is that admission decisions should be explainable. For any object admitted to the hot tier, the platform should be able to report its estimated hotness, size, latency benefit, capacity price, reliability price, and final score. This makes the policy auditable. It also helps operators tune the price updates when service objectives change. The eighth point is that the public trace experiment should be extended before production deployment. A stronger evaluation would include multi-day traces, write traffic, object churn, backend repair events, and tenant labels. The current paper establishes the theoretical and diagnostic structure, but a production storage service should repeat the experiment with its own workload and failure data.

A ninth point concerns failure-domain calibration. Storage systems rarely fail according to independent shard probabilities. A rack-level power event, firmware regression, network partition, or operator mistake can invalidate the independence assumption exactly when durability is most important. A practical deployment should therefore estimate shard risk conditionally on placement domain. The reliability price can then be increased for objects whose shards are concentrated in a high-risk domain, even if their nominal erasure-code parameters look sufficient.

A tenth point concerns repair-window selection. The probability p in the erasure-code loss expression should not be interpreted as an annual device failure rate. It is the probability that a shard is unavailable within the window during which the system has not yet restored redundancy. Faster repair lowers the exposure window, but repair itself consumes bandwidth and can interfere with foreground traffic. This creates a feedback loop: when repair backlog rises, the reliability price should increase and migration should become more conservative.

An eleventh point concerns tenant-specific value. Some objects are valuable because they are large, some because they are frequently read, and others because they belong to high-priority tenants. A production controller can incorporate tenant priority as a multiplier inside the value function. This extension does not

change the dual-price structure; it changes the value density that competes for the hot tier.

A final point concerns auditability after incidents. If an object was demoted shortly before a latency regression, the storage team should be able to reconstruct the exact price signals that justified the decision. The score decomposition should therefore be logged with timestamped hotness, size, latency benefit, capacity price, reliability price, and migration penalty. Without this evidence, a sophisticated policy becomes difficult to trust during incident review. Table A1 summarizes the resulting deployment checklist.

Table A1. Deployment checklist for DTHC-RS.

Deployment Dimension	Question to Ask	Operational Evidence
Metadata	Can hotness and value be estimated at scale?	Sketch memory and update cost
Migration	Will background movement compete with foreground traffic?	Bandwidth and I/O budget
Reliability	Does the risk model include correlated failures?	Failure-domain estimator
Policy class	Do some objects require fixed durability floors?	Storage-class contract
Explainability	Can each admission be explained after the fact?	Logged score decomposition

References

1 Ghemawat S, Gobioff H, Leung ST. The Google File System. In Proceedings of the 2003 ACM Symposium on Operating Systems Principles, Bolton Landing, New York, USA, 19–22 October 2003.

2 Chang F, Dean J, Ghemawat S, *et al.* Bigtable: A Distributed Storage System for Structured Data. *ACM Transactions on Computer Systems (TOCS)* 2008; **26**(**2**): 4.

3 Weil SA, Brandt SA, Miller EL, *et al.* Ceph: A Scalable, High-Performance Distributed File System. In Proceedings of the 7th Conference on Operating Systems Design and Implementation (OSDI'06), Seattle, WA, USA, 6–8 November 2006.

4 DeCandia G, Hastorun D, Jampani M, *et al.* Dynamo: Amazon's Highly Available Key-Value Store. *ACM SIGOPS Operating Systems Review* 2007; **41**(**6**): 205–220.

5 Calder B, Wang J, Ogus A, *et al.* Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency. In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, Cascais, Portugal, 23–26 October 2011.

6 Huang C, Simitci H, Xu Y, *et al.* Erasure Coding in Windows Azure Storage. In Proceedings of the 2012 USENIX Annual Technical Conference, Boston, MA, USA, 13–15 June 2012.

7 Muralidhar S, Lloyd W, Roy S, *et al.* F4: Facebook's Warm BLOB Storage System. In Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), Broomfield, CO, USA, 6–8 October 2014.

8 O'Neil EJ, O'Neil PE, Weikum G. The LRU-K Page Replacement Algorithm for Database Disk Buffering. *Acm Sigmod Record* 1993; **22**(**2**): 297–306.

9 Megiddo N, Modha DS. ARC: A Self-Tuning, Low Overhead Replacement Cache. In Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST 03), San Francisco, CA, USA, 31 March –2 April 2003.

10 Einziger G, Friedman R, Manes B. TinyLFU: A Highly Efficient Cache Admission Policy. *ACM Transactions on Storage* 2017; **13**(**4**): 35.

11 Waldspurger CA, Park N, Garthwaite A, *et al.* Efficient MRC Construction with SHARDS. In Proceedings of the 3th USENIX Conference on File and Storage Technologies (FAST 15), Santa Clara, CA, USA, 16–19 February 2015.

12 Sathiamoorthy M, Asteris M, Papailiopoulos D, *et al.* XORing Elephants: Novel Erasure Codes for Big Data. *Proceedings of the VLDB Endowment* 2013; **6**(**11**).

13 Dimakis AG, Godfrey PB, Wu Y, *et al.* Network Coding for Distributed Storage Systems. *IEEE Transactions on Information Theory* 2010; **56**(**12**): 6362–6376.

14 Rashmi KV, Shah NB, Kumar PV. Optimal Exact-Regenerating Codes for Distributed Storage at the MSR and MBR Points. *IEEE Transactions on Information Theory* 2011; **57**(**8**): 5227–5239.

15 Rashmi KV, Shah NB, Ramchandran K. A Piggybacking Design Framework for Read-and Download-Efficient Distributed Storage Codes. *IEEE Transactions on Information Theory* 2017; **63**(**1**): 308–330.

16 Kadekodi S, Shinde S, *et al.* Giza: Erasure Coding Objects across Global Data Centers. In Proceedings of the 2017 USENIX Annual Technical Conference (USENIX ATC 17), Santa Clara, CA, USA, 12–14 July 2023.

17 Zhang Y, Huang P, Zhou K, *et al.* OSCA: An Online-Model Based Cache Allocation Scheme in Cloud Block Storage Systems. In Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC 20), Boston, MA, USA ,15–17 July 2020.

18 Li J, Wang Q, Lee PP C, *et al.* An In-Depth Analysis of Cloud Block Storage Workloads in Large-Scale Production. In Proceedings of the 2020 IEEE International Symposium on Workload Characterization, Virtual, 27–29 October 2020.

19 Li J, Wang Q, Lee PP C, *et al.* An In-Depth Comparative Analysis of Cloud Block Storage Workloads. *ACM Transactions on Storage* 2023; **19**(**2**): 16.

20 Yang J, Cai Y, Rashmi KV. A Large Scale Analysis of Hundreds of In-Memory Cache Clusters at Twitter. *ACM Transactions on Storage* 2021; **17**(**3**): 17.

21 Berger DS, Beckmann N, Chen M, *et al.* Practical Bounds on Optimal Caching with Variable Object Sizes. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2018; **2**(**2**): 32.

22 Anwar A, Cheng Y, Gupta A, *et al.* Improving Docker Registry Design Based on Production Workload Analysis. In Proceedings of the 16th USENIX Conference on File and Storage Technologies (FAST 18), Oakland, CA, USA, 12–15 February 2018.

23 Narayanan D, Donnelly A, Rowstron A. Write Off-Loading: Practical Power Management for Enterprise Storage. *ACM Transactions on Storage (TOS)* 2008; **4**(**3**): 10.

24 Phothilimthana PM, Kadekodi S, Ghodrati S, et al. Thesios: Synthesizing Accurate Counterfactual I/O Traces from I/O Samples. In Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, La Jolla, CA, USA, 27 April–1 May 2024.

25 Wikimedia Foundation. Data Platform/Data Lake/Traffic/Caching. 2019. Available online: https://wikitech.wikimedia.org/wiki/Data_Platform/Data_Lake/Traffic/Caching (accessed on 25 June 2026).

26 Berg B, Berger DS, McAllister S, et al. The CacheLib Caching Engine: Design and Experiences at Scale. In Proceedings of the14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20), Online, 4–6 November 2020.

27 Liu Z, Lee PP C, Lui JCS. Heterogeneity-Aware Erasure-Coded Storage Placement. *IEEE Transactions on Parallel and Distributed Systems* 2020; **31**(**11**): 2645–2658.

28 Ma Y, et al. Kangaroo: Caching Billions of Tiny Objects on Flash. In Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles, Virtual, 26–29 October 2021.

29 A Comprehensive Open-Source Cache Trace Dataset. Available online: https://github. com/cacheMon/cache_dataset (accessed on 25 June 2026).

30 Saxena M, Swift MM, Zhang Y. FlashTier: A Lightweight, Consistent and Durable Storage Cache. In Proceedings of the 7th ACM european conference on Computer Systems, Bern, Switzerland, 10–13 April 2012.

31 *Harchol-Balter M*. Performance Modeling and Design of Computer Systems: Queueing Theory in Action; Cambridge University Press: Cambridge, UK, 2013.

32 Yan H. Real-Time 3D Model Reconstruction through Energy-Efficient Edge Computing. *Optimizations in Applied Machine Learning* 2022; **2**(**1**).

33 Yan H, Shao D. Enhancing Transformer Training Efficiency with Dynamic Dropout. *arXiv* 2024. https://doi.org/10.48550/arXiv.2411.03236

34 Luo Z, Yan H, Pan X. Optimizing Transformer Models for Resource-Constrained Environments: A Study on Model Compression Techniques. *Journal of Computational Methods in Engineering Applications* **2023**: 1–12. https://doi.org/10.62836/jcmea.v3i1.030107

35 Yan H, Shao D. Multimodal Medical Image Analysis: Integrating LLM and RAG Deep Learning Strategies. *Journal of Advances in Information Technology* 2025; **16**(**4**): 568–581. https://doi.org/10.12720/jait.16.4.568-581

36 Lu Y, Shao D, Ni X, *et al.* Emotion-Style Dual Prediction: A Multi-Task Deep Learning Approach for Artistic Images. *Cluster Computing* 2026; **29**(**1**): 31.

37 Dai Y. Medical Biopharmaceutical Image Anomaly Detection Under Retinex State Space Duality and Frequency Consensus-Driven Transformer. *Journal of Computational Methods in Engineering Applications* 2026; **6**(**1**): 0001.

38 Dai Y. Deep Learning-Based Medical Image Segmentation for Early Cancer Detection. *Optimizations in Applied Machine Learning* 2025; **5**(**1**).

39 Dai Y. MobileMamba-HC: Medical Image Disease Detection in Healthcare Integrating Frequency Adaptive Dilated Convolution and Spatial-Channel Synergistic Attention. *Innovations in Applied Engineering and Technology* 2026; **5**(**1**): 0002.

40 Dai Y, Wei L, Yu C. Graph Neural Network-Based Drug-Target Interaction Prediction for Precision Medicine. *Optimizations in Applied Machine Learning* 2025; **5**(**1**).

41 Li J, Culver TB, Burgis CR, *et al.* Validating Nitrogen Removal Models with Field Bioretention Data. *Journal of Environmental Engineering* 2024; **150**(**8**): 04024037.

42 Li J, Culver TB, Persaud PP, *et al.* Developing Nitrogen Removal Models for Stormwater Bioretention Systems. *Water Research* 2023; **243**: 120381.

43 Li J. Nitrogen Removal Models for Stormwater Bioretention Systems. Ph.D. Thesis, University of Virginia, Charlottesville, VA, USA, 2023.

44 Li J, Culver TB. Review of Process-Based Nitrogen Model for Agricultural Fields with Implications for Nitrogen Simulations in Stormwater BMPs. *Environmental Modelling & Software* 2022; **151**: 105363.

45 Beckmann N, Sanchez D. Talus: A Simple Way to Remove Cliffs in Cache Performance. *I*n Proceedings of the 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), Burlingame, CA, USA, 7–11 February 2015.

46 Deng X, Oda S, Kawano Y. Graphene-Based Midinfrared Photodetector with Bull's Eye Plasmonic Antenna. *Optical Engineering* 2023; **62**(**9**): 097102.

47 Deng X, Li L, Enomoto M, *et al.* Continuously Frequency-Tuneable Plasmonic Structures for Terahertz Bio-Sensing and Spectroscopy. *Scientific Reports* 2019; **9**(**1**): 3498.

48 Zhang Y, Needleman A. On the Identification of Power-Law Creep Parameters from Conical Indentation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 2021; **477**(**2252**): 20210233.

49 Zhang Y, Needleman A. Characterization of Plastically Compressible Solids Via Spherical Indentation. *Journal of the Mechanics and Physics of Solids* 2021; **148**: 104283.

50 Backblaze Drive Stats: Hard Drive Reliability Test Data. Available online: https://www. backblaze. com/cloud-storage/resources/hard-drive-test-data (accessed on 25 June 2026).

51 Ford D, Labelle F, Popovici FI, *et al.* Availability in Globally Distributed Storage Systems. In Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10), Vancouver, BC, Canada, 4–6 October 2010.

52 Cidon A, Stutsman R, Rumble SM, *et al.* Copysets: Reducing the Frequency of Data Loss in Cloud Storage. In Proceedings of the 2013 USENIX Annual Technical Conference (USENIX ATC 13), San Jose, CA, USA, 26–28 June 2013.

53 Terry DB, Theimer MM, Petersen K, *et al.* Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. *ACM SIGOPS Operating Systems Review* 1995; **29**(**5**): 172–182.

54 Kleinrock L. *Queueing Systems, Volume 1: Theory*; Wiley: Hoboken, NJ, USA, 1975.