ReStore-in-Action: Adaptive Multi-Tier Storage Management via Intelligent Data Migration Policy

Shadman Saqib Eusuf
University of Massachusetts Boston
S.Eusuf001@umb.edu

Tianru Zhang
Uppsala University
tianru.zhang@it.uu.se

Teona Bagashvili
Boston University
teona@bu.edu

Manos Athanassoulis
Boston University
mathan@bu.edu

Tarikul Islam Papon
University of Massachusetts Boston
t.papon@umb.edu

ABSTRACT

Modern data systems employ a diverse range of storage devices, from high-performance, expensive SSDs to capacity-oriented SSDs and cost-effective, high-capacity HDDs. To leverage these varying characteristics, storage devices are often organized into Multi-Tiered Storage (MTS) systems. In MTS, data placement must be dynamic to align with evolving workload patterns through a migration policy that continuously upgrades or downgrades pages between tiers. However, managing these movements is challenging, as traditional rule-based methods often lack the flexibility to handle complex workloads and fail to account for hardware-specific properties, such as SSD read-write asymmetry and internal parallelism. To address these limitations, we recently presented ReStore, a lightweight, page-level Reinforcement Learning-based migration policy that adaptively optimizes data placement by considering both workload and storage device characteristics. Experimental results using industry-grade benchmarks and real-life traces show that ReStore achieves up to 6× lower runtime and 48× fewer migrations compared to state-of-the-art baselines. In this demonstration, we present a web simulation of an MTS where we integrate ReStore along with five other baseline policies. The conference participants can configure storage tier properties, generate custom workloads, and observe ReStore's real-time behavior (data placement and migration between tiers) alongside the baseline approaches. The demonstration is available at: https://disc-projects.bu.edu/ReStore/research.html.

PVLDB Reference Format:
Shadman Saqib Eusuf, Tianru Zhang, Teona Bagashvili, Manos Athanassoulis, and Tarikul Islam Papon. ReStore-in-Action: Adaptive Multi-Tier Storage Management via Intelligent Data Migration Policy. PVLDB, 19(12): 4702-4705, 2026.
doi:10.14778/3827998.3828101

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/BU-DiSC/disc-projects.bu.edu/tree/master/ReStore. The demo is available at https://disc-projects.bu.edu/ReStore/research.html.

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828101

1 INTRODUCTION

Multi-Tiered Storage in Big Data Era. The volume of data produced by modern applications continues to grow rapidly, fueled by social media, digital services, sensors, and the Internet of Things. This growth stresses storage systems that must provide low access latency while keeping costs manageable. As a result, storage stacks increasingly adopt *multi-tiered storage* (MTS) design that combine heterogeneous devices with different latency, throughput, and cost characteristics. Typical deployments place faster, smaller devices (e.g., NVMe SSDs or Optane-class storage) at upper tiers, and larger, slower devices (e.g., SATA SSDs or HDDs) at lower tiers [2]. The effectiveness of such systems depends on how well they *place* and *migrate* data across tiers over time.

Related Work and Their Limitations. Prior tiered storage systems mostly manage data at file/object granularity. Early systems like AutoRAID [10] and subsequent solutions rely on policy rules or temperature metadata to guide migration, which do not translate well to *page-level* workloads common in DBMSs and KV stores. Page-level management has been extensively studied in caching and buffer management, including recency-based (LRU [7]), frequency-based (LFU [1]), and hybrid policies (ARC [6], LRFU [5], EXD [3]). These methods struggle under workload drift and ignore device-specific SSD properties: (i) *read/write asymmetry*, where writes can be substantially slower than reads [9], and (ii) *concurrency*, where performance depends on exploiting internal parallelism [8].

Learning-Based Migration Policies. ML approaches such as logistic regression and XGBoost [4] have been applied to data migration and caching in tiered storage. Such methods improve adaptability but incur heavy training overhead and rely on retraining to remain current. Reinforcement learning (RL) instead learns continuously from the environment, naturally adapting to unpredictable workload drifts. RL has been explored for migration decisions [11–13], yet existing deep RL designs are too heavy for fine-grained, page-level decisions and do not explicitly model SSD properties.

Our Approach. ReStore [14] is a lightweight RL-based policy for page-level data migration in multi-tiered storage. ReStore models data placement in tiered storage as a Markov Decision Process, with state variables capturing *workload dynamics* (access recency and frequency) and *device properties* (asymmetry and concurrency). It uses a cooperative multi-agent design with one agent per tier, enabling modularity when tiers are added or removed. The policy learns online using a compact value-function approximation and makes page-level migration decisions with low overhead.

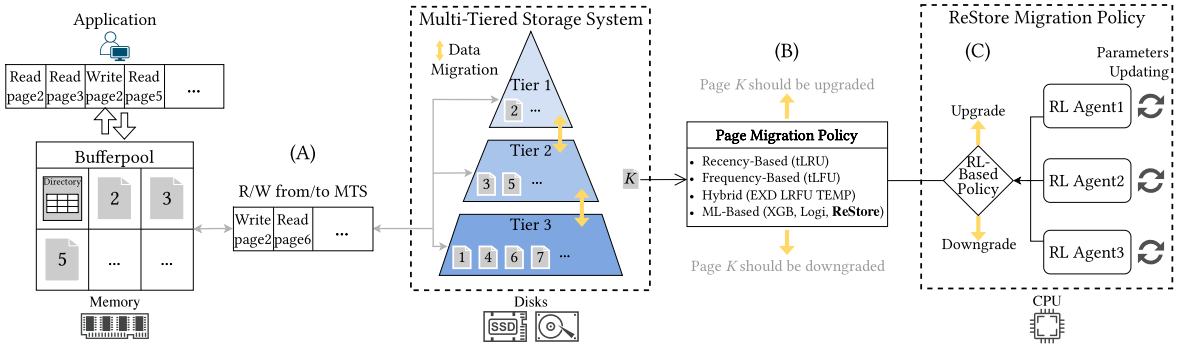


Figure 1: Workflow of ReStore. (A) Application-level I/O traces pass through a buffer pool into the Multi-Tiered Storage System (MTS). (B) A *migration policy* governs data placement by deciding whether each page should be upgraded or downgraded. (C) ReStore automates these decisions via reinforcement learning, optimizing placement across tiers for dynamic workloads.

Demonstration. This demo allows the audience to interact with the ReStore migration policy through a web-based simulation. Users can customize the workload (number of pages and operations, skewness, etc.), configure SSD properties (concurrency, asymmetry), vary tier capacities, and select the page migration algorithm (LRU, LFU, EXD, LRFU, TEMP, ReStore). The simulation provides real-time visualization of the tier states while showing page migrations. Conference participants can visualize performance metrics, enabling side-by-side comparisons across policies under different settings.

2 RESTORE FOR MULTI-TIERED STORAGE

Methods Overview. A Multi-Tiered Storage (MTS) system consists of storage tiers ordered by performance characteristics (e.g., latency and throughput) and capacity, where each tier represents a specific storage device such as SSDs or HDDs. As shown in Fig. 1(A), application-level I/O traces are initially served by a main memory bufferpool and subsequently transmitted to the MTS as a stream of read/write requests. Each request targets a specific page, which resides in exactly one tier at one time. While requests are served from the page's current tier, the system employs a *migration policy* (Fig. 1(B)) to dynamically move pages to faster or slower tiers to optimize future access performance. Since real-world workloads exhibit significant temporal locality and frequent distribution drifts, static thresholds or fixed policies often result in suboptimal placement. The objective is to minimize end-to-end response time while preventing I/O overhead from excessive data movement. Our method, ReStore [14], leverages reinforcement learning (RL) to learn a migration policy that adaptively balances the performance gains of placing *hot* pages in higher tiers against the migration costs in terms of write endurance and bandwidth consumption.

2.1 RL Formulation

We model page migration as a Markov Decision Process (MDP) $\langle S, A, P, R, \gamma \rangle$, and each storage tier is associated with a dedicated agent (e.g., RL Agent 1–3 in Fig. 1(C)) that cooperates toward a global performance goal. The key components are:

- **State** S**:** captures workload dynamics and device properties.
- **Action** A**:** for a requested page, maintain it in the current tier or promote it to a faster neighboring tier.
- **Reward** R**:** negative of the estimated response time, so that maximizing the reward corresponds to minimizing latency.

This formulation enables online adaptation: as workloads shift, the policy updates its value estimates from observed reward feedback.

State Variables. We use two compact state variables per tier to keep the policy lightweight while retaining key signals:

s_1: **Tier temperature:** the average temperature of pages in the tier, summarizing both access *recency* and *frequency*.

s_2: **Tier pressure:** An estimate of queued service time based on pending requests and device-specific concurrency limits.

These variables encode workload intensity and device capability, allowing the policy to decide when migration is worth its cost.

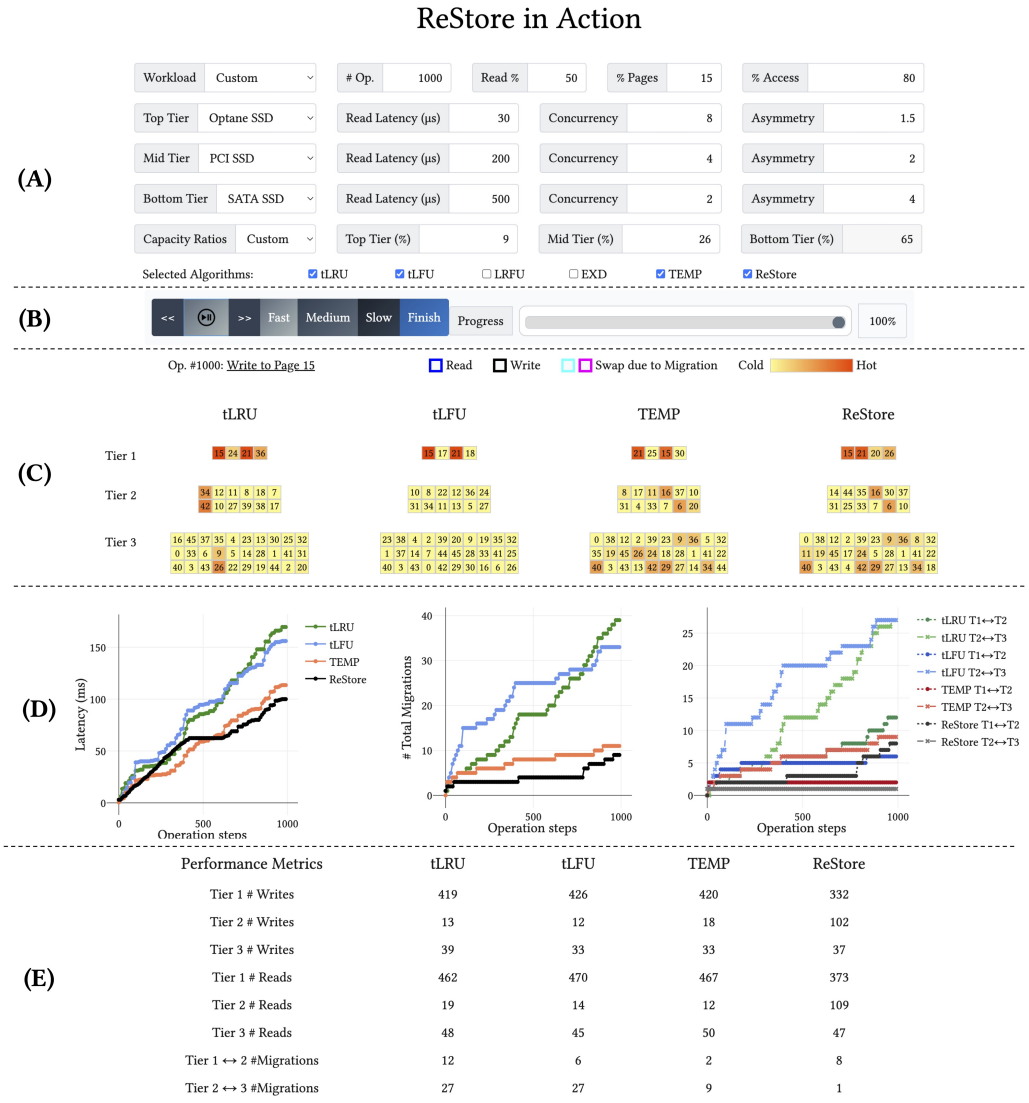
Temperature Model. To combine recency and frequency at page granularity, each page maintains a temperature value that increases upon access and decays over time. This compact statistic tracks both short-term bursts and long-term popularity, providing a stable signal under workload drift. In the RL context, the state variable s_1 aggregates these per-page temperatures to summarize the overall 'hotness' distribution of a tier [14]. Beyond its use in ReStore, this model also powers our TEMP baseline, which makes migration decisions purely based on the page temperatures.

2.2 Value Function and Migration Policy

To map continuous states to the action space efficiently, ReStore employs a lightweight Fuzzy Rule-Based (FRB) value-function approximation. With only 4 parameters and 4 tunable hyperparameters, our FRB design enables per-request decisions without large computational overhead. Parameters are updated online via temporal-difference learning simultaneously with the workload executing. This avoids heavy pre-training while remaining adaptive. The migration policy is then based on the value function $v(s)$: A page K is upgraded to the upper tier $i+1$ from tier i if the condition

$$v^i_{\text{up}}(s') + v^{i+1}_{\text{up}}(s') \geq v^i_{\text{not}}(s) + v^{i+1}_{\text{not}}(s) \tag{1}$$

satisfies, where v_{up} and s' represent the estimated value and state following a promotion, and v_{not} and s correspond to them before upgrade. This logic ensures migrations only occur when the predicted latency benefit outweighs the cost of data movement.



Performance Metrics	tLRU	tLFU	TEMP	ReStore
Tier 1 # Writes	419	426	420	332
Tier 2 # Writes	13	12	18	102
Tier 3 # Writes	39	33	33	37
Tier 1 # Reads	462	470	467	373
Tier 2 # Reads	19	14	12	109
Tier 3 # Reads	48	45	50	47
Tier 1 ↔ 2 #Migrations	12	6	2	8
Tier 2 ↔ 3 #Migrations	27	27	9	1

Figure 2: The demonstration user interface allows the participant to customize (1) the input workload, both via presets and custom workload parameters (# of operations, % of reads, workload skew), (2) the devices at three tiers via presets or by providing the device asymmetry and read latency. Once the setup is configured, the user can execute this workload step-by-step or at varying speeds to observe the migrations and various performance metrics to compare ReStore' against the baselines.

3 THE DEMONSTRATION UI

Overview. Figure 2 shows the ReStore web interface which is organized into five functional modules: (A) input panel, (B) control panel, (C) animation panel, (D) figure panel, (E) performance panel. The interface allows users to configure environmental parameters, visualize real-time data movement, and evaluate the efficacy of ReStore against traditional baselines.

Workload and System Configuration. Through the input panel (A), users can select from pre-configured workload profiles (e.g., read-heavy, write-heavy, skewed, or uniform) or define a custom trace by specifying the number of operations, read/write ratios, and workload skewness (i.e. what fraction of accesses target what fraction of pages). Furthermore, participants can parameterize the underlying hardware, adjusting SSD read/write latencies, concurrency, and tier capacity. The demonstration supports a comparative analysis of six migration policies: tLRU, tLFU, LRFU, EXD, TEMP, and ReStore. As summarized in Table 1, tLRU, tLFU, LRFU, and EXD adapt classical replacement algorithms to a tiered context by adding specific promotion criteria, whereas TEMP relies exclusively on our proposed page temperature model described in Section 2.1.

Interactive Simulation and Visualization. Upon configuration, the control panel (B) enables users to start, pause, or adjust the simulation speed. A dedicated progress bar allows for temporal navigation, including fast-forwarding or back-tracking through the

Table 1: Upgrade/downgrade criteria of different policies.

Policy	Upgrade Criteria	Downgrade
tLFU	Frequency of K > the LFU page in the upper tier	the LFU page
tLRU	Recency of K > the LRU page in the upper tier	the LRU page
LRFU	CRF value of K > lowest CRF value in the upper tier	the lowest CRF value page
EXD	EXD score of K > lowest EXD score in the upper tier	the lowest EXD score page
TEMP	Temperature of K > avg temp in the upper tier	the lowest temperature page
ReStore	RL condition (Eq. 1)	the lowest temperature page

trace execution. The Animation Panel (C) provides a live visualization of the multi-tiered storage state. While the framework supports six distinct migration policies, the dashboard is designed to display a side-by-side comparison of any four selected policies to maintain visual clarity. Individual page accesses are colored blue for reads and black for writes. Promotions are indicated by a cyan box around a page as it moves to a faster tier. Downgrades triggered by capacity constraints in the upper tier are marked in magenta. Further, each page's hotness is shown as a color gradient. By hovering over the page, participants can also see the value of the hotness metric (e.g., recency round for tLRU, temperature for Temp).

Performance Monitoring. The figure panel (D) and performance panel (E) provide real-time visualization of system statistics, including per-tier I/O counts and migration frequencies. The figure panel (D) generates three plots to monitor the accumulated system latency and total number of migrations as the workload progresses. This allows users to quantify improvements in system latency and reductions in migration overhead achieved by ReStore.

4 DEMONSTRATION SCENARIOS

This section outlines three guided scenarios that allow attendees to interactively explore page migration dynamics within an MTS.

Scenario 1: Visualizing Adaptive Migrations in MTS. In this scenario, users can see the operation of an MTS and perform a comparative analysis of ReStore against traditional baselines, including tLRU, tLFU, and the temperature-only TEMP. Through the animation panel, participants can observe the real-time movement of pages as each policy manages promotions and downgrades. Unlike static recency- or frequency-based methods that often trigger excessive, suboptimal migrations, users can visualize how ReStore optimizes data placement by learning the specific cost-benefit tradeoff of each potential move. This demonstrates how ReStore decisions result in significantly lower workload latency by placing the right data in the right tier at the right time. By monitoring the figure and performance panel, the users can compare ReStore against other baselines in terms of latency, migration count and per-tier I/O stats.

Scenario 2: Hardware Heterogeneity and Tier Constraints. Participants can modify MTS parameters to observe how ReStore adapts to different hardware profiles, including changing devices, adjusting read/write latencies, device concurrency, and tier capacities. For example, by narrowing the performance gap between Tier 2 and Tier 3, users can see that RL agents automatically reduce migration frequency. This illustrates the ability of the policy to preserve the system bandwidth when the marginal performance gain of a promotion no longer justifies the I/O cost of the migration.

Scenario 3: Workload Drift and Custom Traffic. This scenario highlights the online-updating capabilities of ReStore. Users can experiment with dynamic workloads that shift from read-heavy to write-heavy mid-simulation, or with "drifting" workloads in which the skew level and the percentage of "hot" pages fluctuate over time. While traditional policies like tLFU or tLRU often lag behind these shifts due to their reliance on simple, historic features, participants can monitor the performance panel as ReStore agents minimize end-to-end latency under changing access patterns.

5 CONCLUSION

In this paper, we demonstrate ReStore, a lightweight, page-level reinforcement learning-based migration policy for multi-tiered storage systems. ReStore optimizes data placement by considering both workload patterns and storage device-specific properties. ReStore continuously adapts to workload shifts without retraining overhead. The live demonstration allows attendees to directly experience these benefits by configuring storage tier properties, generating custom workloads, and observing ReStore's real-time migration decisions alongside five popular baseline policies.

ACKNOWLEDGMENTS

This work was funded by Kjell and Märta Beijer Foundation, the US National Science Foundation under Grants No. IIS-2144547 and CCF-2403012, a Facebook Faculty Research Award, and a Meta Gift.

REFERENCES

[1] Martin F Arlitt, Ludmila Cherkasova, John Dilley, Rich Friedrich, and Tai Jin. 2000. Evaluating content management techniques for Web proxy caches. *SIGMETRICS Performance Evaluation Review* 27, 4 (2000).

[2] Edward I Cohen, Gary M King, and James T Brady. 1989. Storage Hierarchies. *IBM Systems Journal* 28, 1 (1989).

[3] Avrilia Floratou, Nimrod Megiddo, Navneet Potti, Fatma Özcan, Uday Kale, and Jan Schmitz-Hermes. 2016. Adaptive Caching in Big SQL using the HDFS Cache. In *Proceedings of the ACM Symposium on Cloud Computing*.

[4] Herodotos Herodotou and Elena Kakoulli. 2019. Automating Distributed Tiered Storage Management in Cluster Computing. *Proceedings of the VLDB Endowment* 13, 1 (2019).

[5] Donghee Lee, Jongmoo Choi, Jong-Hun Kim, Sam H Noh, Sang Lyul Min, Yookun Cho, and Chong-Sang Kim. 2001. LRFU: A Spectrum of Policies that Subsumes the Least Recently Used and Least Frequently Used Policies. *IEEE Trans. Comput.* 50, 12 (2001).

[6] Nimrod Megiddo and Dharmendra S. Modha. 2003. ARC: A self-tuning, low overhead replacement cache. In *Proceedings of the USENIX Conference on File and Storage Technologies*.

[7] Elizabeth J. O'Neil, Patrick E. O'Neil, and Gerhard Weikum. 1993. The LRU-K Page Replacement Algorithm For Database Disk Buffering. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*.

[8] Tarikul Islam Papon. 2024. Enhancing Data Systems Performance by Exploiting SSD Concurrency & Asymmetry. In *Proceedings of the IEEE International Conference on Data Engineering*.

[9] Tarikul Islam Papon and Manos Athanassoulis. 2021. A Parametric I/O Model for Modern Storage Devices. In *Proceedings of the International Workshop on Data Management on New Hardware*.

[10] John Wilkes, Richard A Golding, Carl Staelin, and Tim Sullivan. 1996. The HP AutoRAID Hierarchical Storage System. *ACM Transactions on Computer Systems* 14, 1 (1996).

[11] Tianru Zhang. 2024. Autonomous Hierarchical Storage Management via Reinforcement Learning. In *Proceedings of the VLDB Endowment (PhD Workshop)*.

[12] Tianru Zhang, Ankit Gupta, María Rodríguez, Ola Spjuth, Andreas Hellander, and Salman Toor. 2024. Data management of scientific applications in a reinforcement learning-based hierarchical storage system. *Expert Systems with Applications* 237, Part B (2024).

[13] Tianru Zhang, Andreas Hellander, and Salman Toor. 2023. Efficient Hierarchical Storage Management Empowered by Reinforcement Learning. *IEEE Transactions on Knowledge and Data Engineering* 35, 6 (2023).

[14] Tianru Zhang, Tarikul Islam Papon, Teona Bagashvili, Salman Toor, and Manos Athanassoulis. 2026. ReStore: A Reinforcement Learning Approach for Data Migration in Multi-Tiered Storage. *Proceedings of the ACM on Management of Data (PACMMOD)* 4, 3, Article 227 (2026).