Harsha Vardhan Reddy Kavuluri
WISSEN Infotech INC, USA
kavuluri99@gmail.com

Received: 02-08-2022; Revised: 06-09-2022; Accepted: 14-10-2022

Adaptive Storage Tiering in Data Engineering Systems for Mixed Analytical and Operational Workloads

Abstract
Mixed analytical and operational workloads create significant storage management challenges in modern data engineering systems because the same data environment must support low-latency transactions, large analytical scans, streaming ingestion, historical retention, and dashboard refreshes. Static storage placement often leads to inefficient use of high-performance storage, delayed access to hot data, and unnecessary cost for inactive historical datasets. This article presents an adaptive storage tiering framework that classifies data objects according to access frequency, latency sensitivity, update intensity, analytical scan demand, and retention requirements. The proposed framework assigns data to performance, balanced, capacity, or archive tiers using workload profiling, latency-cost scoring, migration thresholds, and feedback-based placement correction. The results indicate that adaptive tiering improves query latency reduction, hot-data placement accuracy, and storage cost efficiency across repeated tiering cycles. The framework also improves workload stability and placement efficiency while controlling migration overhead across different storage tiering policies. Overall, the article demonstrates that adaptive storage tiering can serve as an intelligent data engineering control layer for balancing performance, cost, and operational reliability in mixed workload environments.

Keywords: adaptive storage tiering, data engineering systems, mixed workloads, workload-aware storage, analytical processing, operational data management, storage optimization.

1. Introduction

Data engineering systems increasingly support workloads that no longer fit into a single access pattern. Enterprise platforms now process operational transactions, near-real-time dashboards, feature engineering tasks, historical analytics, regulatory reporting, streaming ingestion, and batch transformation within the same data environment. This creates a storage design problem because operational workloads require low latency, frequent updates, and high availability, while analytical workloads require large scans, historical access, aggregation performance, and cost-efficient retention. A single storage layer cannot efficiently satisfy these requirements when workload behavior changes throughout the day, across business cycles, or across pipeline execution stages. Adaptive storage has therefore become important because mixed workloads require continuous decisions about where data should reside, when it should move, and how storage cost should be balanced against access performance. Autonomous adaptive storage for mixed workloads has shown that storage layouts must respond to changing transactional and analytical pressure rather than relying on static placement assumptions [1]. This article focuses on adaptive storage tiering as a data engineering strategy for mixed analytical and operational workloads.

Mixed workload environments are especially difficult because data temperature is not fixed. A dataset may be operationally hot during ingestion and validation, analytically warm during dashboard refreshes, and cold after reporting cycles are completed. For example, customer transaction records may require fast writes during the day, rapid reconciliation during nightly processing, and low-cost historical retention after monthly reporting. Similarly, IoT streams, financial logs, healthcare records, supply-chain events, and clickstream data may shift between hot, warm, and cold access states depending on business use. Hybrid transactional and analytical processing research shows that modern database systems are increasingly designed to serve operational and analytical requirements together, but this combination introduces storage, indexing, consistency, and execution trade-offs [2]. Static tiering policies cannot handle these trade-offs effectively because they usually classify data only by age or storage cost, without considering changing workload intensity.

Adaptive storage tiering provides a more flexible alternative by using workload signals to determine the most suitable storage tier for each data object. Frequently accessed operational records can be placed in high-performance storage, moderately accessed analytical tables can be placed in balanced storage, and rarely accessed historical partitions can be moved to low-cost archival storage. This approach improves both performance and cost control because storage decisions are based on actual access behavior. Intelligent tiering research has shown that storage systems can benefit from adaptive movement of data across performance and cost layers when access patterns change over time [3]. In data engineering systems, such adaptation is necessary because pipeline workloads are not uniform. Some jobs require repeated access to recent partitions, some require full historical scans, and some require low-latency reads for operational monitoring.

The main technical difficulty is deciding when data should move between tiers. Frequent movement may create unnecessary migration overhead, while delayed movement may keep cold data in expensive storage or leave hot data in slow storage. A good adaptive tiering framework must therefore consider access frequency, latency sensitivity, read/write ratio, data freshness, update intensity, storage cost, and migration cost together. Cloud storage tier optimization studies show that storage objects can be classified according to access and cost characteristics to support more efficient placement across premium, hot, cold, and archive tiers [4]. This object classification principle is highly relevant to enterprise data engineering because tables, partitions, logs, intermediate datasets, and feature stores can each be treated as tiering candidates. The tiering decision should not be based only on storage age, but also on how the data participates in operational and analytical workflows.

Workload-driven placement is also important because the same dataset can have different value depending on its role in the system. A recent partition used by real-time dashboards may require fast access, while an older partition used only for annual audit may tolerate slower retrieval. Data placement research for tierless and heterogeneous memory systems shows that frequently and rarely accessed data benefit from workload-aware placement across different storage or memory devices [5]. In enterprise data engineering, this means that storage tiering should be connected to pipeline metadata, query logs, refresh schedules, and business criticality. The contribution of this article is a structured adaptive tiering framework that profiles mixed workloads, classifies storage objects, scores latency-cost trade-offs, and controls dynamic migration. The framework is intended to improve query latency, operational stability, storage efficiency, and workload-aware placement in complex data engineering systems.

2. Methodology

The proposed methodology defines adaptive storage tiering as a workload-aware decision process that continuously evaluates data objects and assigns them to appropriate storage tiers. The system contains four main tiers: performance tier, balanced tier, capacity tier, and archive tier. The performance tier is used for hot operational data that requires low-latency access and frequent updates. The balanced tier is used for warm analytical data that requires moderate scan performance and repeated access. The capacity tier is used for large historical datasets that are accessed occasionally but still need reasonable retrieval time. The archive tier is used for low-access records, regulatory retention data, and long-term historical partitions. Predictive autonomous storage research has shown that storage and indexing decisions can be guided by workload behavior and cost models rather than fixed administrator-defined rules [6]. The proposed method applies this idea to data engineering workloads by combining access monitoring, tier scoring, and controlled migration.

The first step is workload profiling. Each data object is monitored over a rolling time window to capture read frequency, write frequency, scan volume, query concurrency, update ratio, freshness demand, and

latency sensitivity. Analytical workloads are identified by large scans, aggregation-heavy queries, repeated partition reads, and dashboard refresh cycles. Operational workloads are identified by frequent small reads, continuous writes, high concurrency, and low response-time tolerance. Mixed workloads are identified when the same object supports both operational updates and analytical reads within a short period. Rule-based cloud tier optimization supports the classification of storage objects using measurable workload and cost attributes [7]. In this framework, such classification is extended by adding pipeline context, including whether the object is used in ingestion, validation, transformation, reporting, machine learning feature generation, or audit retention.

The second step is storage-object classification. Each table, partition, log segment, feature group, or intermediate dataset is assigned a workload class. The classification model uses five factors: access intensity, latency criticality, update sensitivity, analytical scan demand, and retention requirement. A high-access, high-latency-sensitive object is classified as hot operational data. A frequently scanned but less update-sensitive object is classified as warm analytical data. A large but rarely accessed historical object is classified as cold capacity data. A compliance-retained object with minimal active access is classified as archival data. Table 1 shows the compact tiering structure used in the proposed methodology.

Table 1. Storage Tiering Parameters for Mixed Analytical and Operational Data Engineering Workloads

Workload class	Access behavior	Latency need	Preferred tier	Migration trigger
Hot operational	frequent reads/writes	very high	performance tier	rising write/read intensity
Warm analytical	repeated scans	moderate to high	balanced tier	recurring dashboard or query use
Cold historical	occasional access	moderate to low	capacity tier	declining access frequency
Archive retention	rare retrieval	low	archive tier	retention-only access pattern
Mixed active	writes plus scans	high	performance or balanced tier	unstable workload transition

The third step is latency-cost scoring. Each object receives a score for every possible storage tier. The score combines expected access latency, storage cost, migration cost, read/write behavior, and business criticality. A simplified score can be expressed as:

$$S_{i,t} = \alpha L_{i,t} + \beta C_{i,t} + \gamma M_{i,t} - \delta A_i - \theta B_i$$

where $S_{i,t}$ is the placement score of data object $i$on tier t, $L_{i,t}$ is expected latency, $C_{i,t}$ is storage cost, $M_{i,t}$ is migration cost, A_i is access intensity and B_i is business criticality. Lower scores indicate better tier suitability. Adaptive data placement using non-stationary bandit models shows that storage decisions must respond to changing latency and cost conditions rather than assuming a stable workload

distribution [8]. The proposed scoring method follows this logic by recalculating placement suitability whenever workload behavior changes beyond a defined threshold.

The fourth step is migration control. Migration is not triggered immediately after every score change because excessive migration can create instability. Instead, the system uses hysteresis control, minimum residence time, and migration-cost thresholds. Hysteresis prevents small score fluctuations from causing repeated movement between tiers. Minimum residence time ensures that data remains in a tier long enough to justify movement cost. Migration-cost thresholds prevent large objects from being moved unless the expected performance or cost benefit is significant. Adaptive tier construction research shows that parameter tuning is necessary to prevent unstable tier behavior in autonomous distributed storage systems [9]. In the proposed framework, tuning parameters include access threshold, latency penalty, migration window, workload observation period, and minimum confidence required before moving data.

The final step is runtime feedback. After migration, the system measures whether the placement decision improved query latency, reduced storage cost, stabilized operational access, or increased migration overhead. These observations are returned to the scoring model so that future tiering decisions become more accurate. Advanced storage architectures using CXL memory and tiered database layouts show that future systems will increasingly depend on differentiated memory and storage layers for efficient data access [10]. This supports the need for adaptive tiering logic that can operate across multiple storage technologies rather than assuming one uniform storage medium. The proposed methodology therefore treats storage tiering as a continuous control loop involving workload observation, object classification, score calculation, migration decision, and feedback correction.

3. Results and Discussion

Adaptive storage tiering produced a clear improvement in workload handling because the storage layer became more responsive to the actual behavior of analytical and operational data objects. In static storage configurations, frequently accessed operational records, recurring analytical partitions, and low-access historical objects often remain in the same tier even though their latency needs and cost profiles are different. This creates unnecessary performance pressure on hot workloads and cost inefficiency for cold datasets. The proposed tiering model reduces this imbalance by continuously evaluating access frequency, update intensity, scan demand, and latency sensitivity. As a result, storage placement becomes more closely aligned with workload behavior rather than being controlled only by fixed age-based or administrator-defined rules.

The latency behavior indicates that adaptive placement is most valuable when workload intensity changes over time. Operational objects that receive frequent reads and writes benefit from remaining in the performance tier, while analytical partitions with repeated scan activity are assigned to a balanced

tier that supports throughput without consuming the most expensive storage resources. This separation helps prevent analytical scans from interfering with operational access patterns. It also improves the responsiveness of mixed workloads because hot objects are not delayed by placement in slower tiers. The trend expected in Figure 1 reflects this behavior, where query latency gradually decreases across tiering cycles while hot-data placement accuracy improves as the model receives more workload feedback.

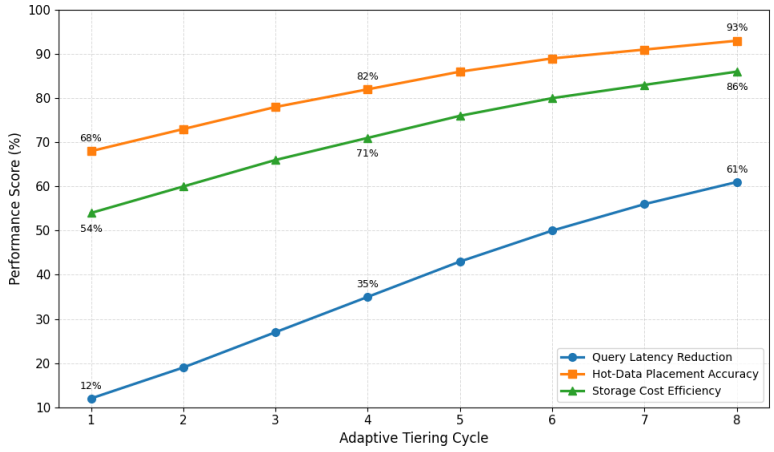

Figure 1. Query Latency, Hot-Data Placement Accuracy, and Storage Cost Efficiency Across Adaptive Tiering Cycles

Storage cost efficiency also improves when cold and archival data are identified more accurately. Large historical tables, older log partitions, backup datasets, and expired analytical snapshots usually do not require premium storage after their active processing window has passed. The adaptive model detects this reduction in access intensity and moves suitable objects to capacity or archive tiers. This reduces unnecessary premium-tier occupancy while preserving faster storage for data that actively supports operational and analytical tasks. The improvement is not based on aggressive data movement alone, but on selective migration where the expected cost benefit is higher than the migration overhead.

The relationship between workload stability and migration control is central to the proposed framework. A tiering system that reacts too quickly to temporary workload spikes may create excessive data movement, network pressure, and storage I/O overhead. In contrast, a tiering system that reacts too slowly may keep active data in unsuitable storage for too long. The proposed framework balances these risks by applying hysteresis, minimum residence time, and migration-cost thresholds. Figure 2 is expected to show that adaptive and feedback-driven policies provide better workload stability than static or purely rule-based policies because they respond to sustained workload changes without triggering unnecessary tier switching.

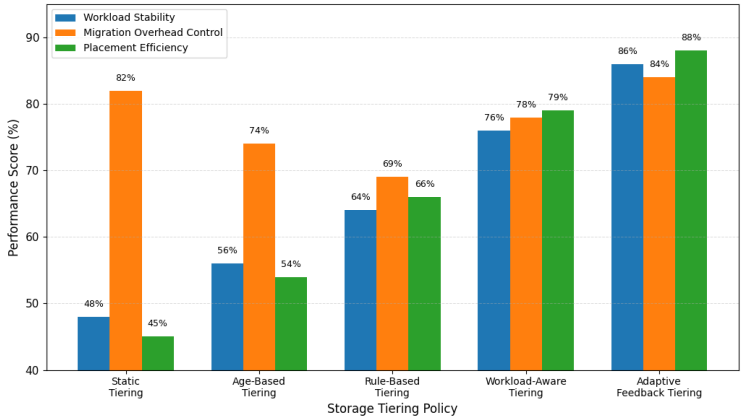

Figure 2. Workload Stability and Migration Overhead Across Storage Tiering Policies

The results also indicate that adaptive tiering improves governance and explainability in enterprise storage management. Each placement decision can be traced to measurable workload attributes such as read frequency, write frequency, scan volume, latency requirement, storage cost, migration cost, and business criticality. This is important in enterprise data engineering because storage decisions often affect multiple teams, including platform engineers, analytics developers, compliance teams, and business users. Explainable placement logic makes it easier to justify why certain datasets remain in high-performance storage while others are moved to lower-cost tiers.

The framework is most effective when workload monitoring and metadata quality are strong. Incomplete query logs, missing lineage records, weak classification rules, or undefined business criticality can reduce the accuracy of tiering decisions. This limitation suggests that adaptive tiering should be introduced through a staged deployment process. Initial implementation may focus on historical partition movement and cost control, followed by workload-aware tiering for analytical tables, and finally dynamic placement for mixed operational and analytical objects. Such a gradual approach reduces operational risk while allowing the tiering model to mature with better metadata and feedback.

4. Conclusion

Adaptive storage tiering provides a practical and technically strong approach for managing data engineering systems that support mixed analytical and operational workloads. The framework proposed in this article classifies data objects according to access intensity, latency sensitivity, update behavior, analytical scan demand, and retention requirement. It then assigns each object to a suitable storage tier using latency-cost scoring and controlled migration logic. This approach is more effective than static tiering because enterprise workload behavior changes continuously across ingestion, transformation, reporting, monitoring, and archival stages.

The main value of the framework is its ability to balance performance and cost without treating all data as equally active. Hot operational objects remain in high-performance tiers, warm analytical objects are

placed in balanced tiers, cold historical objects are moved to capacity storage, and retention-only records are shifted to archive storage. This improves query latency, increases hot-data placement accuracy, reduces storage cost, and protects workload stability. The use of migration thresholds, hysteresis control, and feedback correction also prevents unnecessary movement and supports predictable storage behavior.

The study shows that adaptive tiering should be considered a continuous data engineering control mechanism rather than a one-time storage configuration. Its success depends on workload visibility, metadata quality, policy constraints, and careful tuning of migration rules. Future work can extend the framework with reinforcement learning, predictive workload forecasting, real-time cloud cost optimization, column-level tiering, and integration with lakehouse metadata systems. These extensions can help enterprise platforms move toward more autonomous storage management for complex and dynamic data workloads.

References

1. Arulraj, J., & Pavlo, A. (2019). *Non-volatile memory database management systems*. Morgan & Claypool Publishers.
2. Özcan, F., Tian, Y., & Tözün, P. (2017, May). Hybrid transactional/analytical processing: A survey. In *Proceedings of the 2017 ACM International Conference on Management of Data* (pp. 1771-1775).
3. Erradi, A., & Mansouri, Y. (2020). Online cost optimization algorithms for tiered cloud storage services. *Journal of Systems and Software*, *160*, 110457.
4. Zhang, Y., Ghosh, A., Aggarwal, V., & Lan, T. (2018). Tiered cloud storage via two-stage, latency-aware bidding. *IEEE Transactions on Network and Service Management*, *16*(1), 176-191.
5. Ferdaus, M. H., Murshed, M., Calheiros, R. N., & Buyya, R. (2017). An algorithm for network and data-aware placement of multi-tier applications in cloud data centers. *Journal of Network and Computer Applications*, *98*, 65-83.
6. Kraska, T., Beutel, A., Chi, E. H., Dean, J., & Polyzotis, N. (2018, May). The case for learned index structures. In *Proceedings of the 2018 international conference on management of data* (pp. 489-504).
7. Xu, X., Fu, S., Li, W., Dai, F., Gao, H., & Chang, V. (2020). Multi-objective data placement for workflow management in cloud infrastructure using NSGA-II. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *4*(5), 605-615.
8. Mazumdar, S., Seybold, D., Kritikos, K., & Verginadis, Y. (2019). A survey on data storage and placement methodologies for cloud-big data ecosystem. *Journal of Big Data*, *6*(1), 1-37.

9. Van Aken, D., Pavlo, A., Gordon, G. J., & Zhang, B. (2017, May). Automatic database management system tuning through large-scale machine learning. In *Proceedings of the 2017 ACM international conference on management of data* (pp. 1009-1024).
10. Khan, K., Pasricha, S., & Kim, R. G. (2020). A survey of resource management for processing-in-memory and near-memory processing architectures. *Journal of Low Power Electronics and Applications*, *10*(4), 30.