International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| Volume 4, Issue 1, 165-174, 2023
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V4I1P118

Original Article

AdaptCacheAI: Adaptive Hybrid Caching with Machine-Learned Eviction for Dynamic Cloud Workloads

DevenderRao Takkalapally[1], Mahender Rao Takkellapally[2]
[1]Performance Architect at Virtusa Corporation, USA.
[2]Senior Manager at Cognizant, USA.

Abstract - *Dynamic cloud workloads bring challenges such as high variability, multitenant interference, and heterogeneous access patterns, which make it necessary to have effective cache eviction to retain low latency and cost efficiency. Conventional eviction algorithms like LRU, LFU, ARC, and W-TinyLFU, which are based on static heuristics, give good results under stable conditions but their performance decreases significantly when request patterns change rapidly or differ across applications. AdaptCacheAI, to solve these problems, features an adaptive hybrid caching framework that combines machine learning–driven eviction with a multi-tier cache architecture going from memory to SSD layers. A predictive eviction model that, in real time, calculates the probability of reuse of an object is at the center, supported by a workload classification engine that is capable of recognizing patterns like temporal hotspots, bursty traffic, and tenant-specific behaviors. The continuous feedback loop that accounts for signals such as latency, eviction regret, and tier pressure refurbishes the decisions and thus allows AdaptCacheAI to modify policies on a dynamic basis. Tests on diversified traces confirm the substantial improvements that traditional algorithms can make through the employment of AdaptCacheAI, such as increased hit rates, decreased tail latencies, and lower SSD write amplification, all resulting in tangible performance gains and cost savings. AdaptCacheAI serves as a demonstration of the feasibility of employing machine learning for intelligent cache management and also it paves the way for the potential of future enhancements such as reinforcement learning–based optimization and cross-layer coordination in cloud systems.*

Keywords - *Hybrid Caching, Cloud Computing, Machine Learning, Eviction Policy, Adaptive Systems, Dynamic Workloads, Cache Optimization, Predictive Modeling, Multi-Tier Storage.*

1. Introduction

1.1. Challenges in Modern Cloud Caching

Elasticity, multi-tenancy, and the inherent unpredictability of distributed applications are the key features that characterize cloud computing platforms. Data access patterns volume and variety continue to increase as companies deploy microservices, serverless functions, and globally distributed systems. On the one hand, elasticity enables resources to scale quickly; however, on the other hand, it also increases traffic variability, thus causing sudden spikes of request rates that break caching behavior in a steady state. Sharing the same cache infrastructure, multi-tenant environments, thus, different applications with totally different workloads, lead to interference, noisy neighbors, and frequent changes in access distributions. Unpredictability is becoming the standard instead of the exception, hence the static caching strategies are more and more insufficient.

Contemporary workloads are bursty, have domain shifts and seasonal patterns; they can switch from read-intensive to write-heavy phases within a few minutes, or be suddenly boosted by an external factor such as a product launch or a media mention. At the same time, cold-start situations serverless architectures or newly deployed services disrupt cache warm-up and lower performance even further during peak moments. On top of it all, there are differences in object features: the items vary greatly in size, TTLs, expiration behavior, and access frequencies. Lots of cloud services, for example, AWS ElastiCache, Redis, Memcached, and CDN edge caches, put vendor-specific limitations on eviction strategies, memory layout, and storage tiers, thus making cross-platform consistency a challenge.

Traditional eviction algorithms LRU, LFU, ARC, W-TinyLFU, and their derivatives have been developed taking into account the assumptions of stationary or slowly changing access patterns. These heuristics perform well when the past is a good predictor of the near future; however, under rapid workload transitions or heterogeneous tenants, they frequently evict highly valuable objects and keep stale or low-utility data. Besides this, the trade-off between memory cost and performance has balancing challenges: on the one hand, DRAM used for high-performance is costly, while on the other hand, SSD-based tiers, though less expensive, introduce latency and write amplification concerns. As enterprises strive to get the best performance for a limited budget, these issues point out the need for smart cache systems that can adjust dynamically.

1.2. Problem Statement

Even after research on caching algorithms for decades, there is still a big difference between the capability of eviction policies to adjust to non-stationary and very dynamic cloud environments. Present methods depend on predetermined rules, heuristics made by hand, or thresholds that are manually adjusted and that cannot react quickly to changes in workloads. They do not have the context of the patterns of the workload, the behavior of the tenant, or the pressure of the resource in the different tiers. Therefore, most of the time, they decide to evict the wrong items resulting in a decrease in hit rates and an increase in latency, which is aggravated, in particular, during peak or anomalous situations.

ML-based caching research has shown promise; however, it is still constrained in practical scenarios. Some approaches depend heavily on offline training, large neural models, or resource-intensive feature extraction pipelines that are incompatible with low-latency cloud systems due to their overhead. Furthermore, some models do not support online learning and therefore cannot adjust to new workloads without retraining. Moreover, the majority of ML caching frameworks assume single-tier caches and do not consider the hybrid memory-SSD architectures that are typically used in cloud infrastructures. The lack of frameworks that facilitate predictive eviction, online learning, workload classification, and multi-tier coordination constitutes a significant gap in distributed systems of today.

Hence, the necessity of a single intelligent caching system capable of learning optimal eviction strategies on-the-fly, continuously adapting to variable workload situations, and being functionally efficient across hybrid cache hierarchies is very apparent. Such a device would integrate the stability of conventional policies with the inferential capability of machine learning without the imposition of exorbitant computational or operational costs.

1.3. Motivation

Adaptive,machine-learned caching is a response to several technological trends. Edge computing and microservices, for example, have deeply fragmented data access patterns, causing workloads to be smaller in scale but more variable. It is not easy to imagine that a single microservice can change very fast its local or frequency characteristic, and hence the behavior of cache coordinating across a hundred services becomes like an algorithm-level problem. At the same time, the cost of high-end memory resources, especially DRAM, is still going up, which means that organizations have to make cache work more efficiently to save money in the cloud. New layers that use SSDs are slowing down the price trend without real help from intelligent milestones. Labor Generator Disks are increasing write amplification and therefore frequency.

The idea of a self-governing cloud component that requires no hand tuning and automatically adapts to the workload is becoming more popular. Machine learning is an excellent way of making caches smarter by enabling them to predict the workload, forecast the requests, and decide on call evict scoring most efficiently. The cloud caches, which are using lightweight online learning models, can compute object reuse probability, recognize the new access pattern and change policies immediately. This results in the potential of achieving very low cache misses, tail latency as well as resource usage optimization across tiers.

In fact, an ML-backed caching system performs two major functions: first, it makes the system highly efficient; second, it brings down the operational costs substantially a very important factor, which is at the core of large-scale cloud deployments. Hence, the development of AdaptCacheAI, an adaptive caching framework, capable of tackling these modern challenges, is driven by this motivation.

2. Literature Review

2.1. Traditional Cache Eviction Policies

Classical eviction algorithms such as Least Recently Used (LRU), Least Frequently Used (LFU), First-In–First-Out (FIFO), Most Recently Used (MRU), and Random represent the basic methods for cache replacement. LRU selects items accessed recently, thus assuming temporal locality, whereas LFU chooses high-frequency items assuming their long-term popularity. FIFO evicts items in the order of insertion without taking access patterns into account, and MRU removes the most recently accessed object, which is only applicable to a few stack-like workloads. Random eviction is simple and quite robust; however, it does not have any locality-awareness. To overcome the shortcomings of the basic policies, the researchers came up with multi-criteria versions: LRU-K takes into account K most recent references in order to recognize long-term reuse; 2Q distinguishes recently seen versus frequently accessed items; CLOCK imitates LRU by means of a circular buffer and reference bits; and ARC (Adaptive Replacement Cache) adjusts the balance between recency and frequency by the adaptive tuning of two lists. These systems, although they perform well under stable access distributions, their efficiency is lowered under the workloads that are dynamic and have abrupt shifts, multi-tenant interference, or unpredictable request bursts. Static heuristics are not able to foresee the changing behavior which results in excessive churn and reduced hit rates.

2.2. Modern Adaptive Caching Mechanisms

Some of these advanced adaptive algorithms in recent studies have to a great extent succeeded in solving the problems of traditional policies, at least in theory. One of the algorithms, W-TinyLFU (Windowed-TinyLFU), combines frequency sketches

with a small window for recency thus it shows strong performance under skewed workloads. Nevertheless, its frequency decay model can be out of sync with sudden pattern shifts. Hyperbolic caching implements a mathematical prioritization function based on size and frequency, thereby improving byte hit rates, but it needs manual tuning. LeCaR (Learning Cache Replacement) utilizes a bandit-inspired learning framework to mix LRU and LFU weights, thus gradually changing based on regret feedback. Even though LeCaR is able to adapt to changing patterns, it converges slowly under volatile workloads and also, it does not have multi-tier awareness. SIEVE streamlines eviction by concentrating on temporal filtering, thus it attains high efficiency for single-tier caches but provides limited adaptability to wider workload diversity. In general, contemporary adaptive policies have gained flexibility but they are still limited by rule-based heuristics or narrow adaptation mechanisms which are not enough for very dynamic cloud environments.

2.3. Machine-Learning-Based Cache Systems

Machine-learning-based caching methods investigate both reinforcement learning and supervised prediction to enhance eviction precision. Systems based on reinforcement learning consider the process of sequentially evicting actions, thus they optimize long-term hit rates by means of reward signals. In fact, RL-based methods are theoretically very strong but they can be quite heavy from a computational point of view and require a lot of exploration, which is not that safe in latency-sensitive environments. Supervised predictive techniques that try to estimate future reuse probability of a data point by using historical access features, however, training done offline can limit generalization when workloads drift. Furthermore, there are also challenges in model overhead, feature extraction complexity, and deployment feasibility. ML models usually bring latency and memory overhead, and many of them do not have online learning capabilities that are required for real-time adaptation. Moreover, the majority of them also tend to focus on single-tier caches thus ignoring the architectural realities of hybrid memory–SSD systems that are widely used in modern cloud infrastructures.

2.4. Hybrid and Multi-Tier Caching Architectures

Hybrid caching architectures use DRAM to provide very fast access while SSDs are used to give larger and cost-efficient storage. Facebook's CacheLib is an example of such an approach that mixes software-based memory and SSD caching to deliver both good performance and low cost. The introduction of SSD tiers brings several issues call write amplification, endurance, and variable read latencies that make eviction strategies have new kinds of problems. CDN providers have caches with a hierarchy beyond single-cluster systems where edge, regional, and core layers work together to get the maximum global hit rates. The existence of these multi-level systems necessitates a thorough understanding of data locality, consistency policies, and asynchronous content propagation. The data locality in serverless and microservices-based architectures is even more sensitive because of short-lived containers, unpredictable cold starts, and distributed execution across regions which makes cache coordination across tiers a more difficult problem.

2.5. Research Gap Identified

Despite considerable improvements in replacement algorithms, adaptive caching mechanisms, and ML-based predictors, a literature survey reveals no comprehensive solution that combines hybrid caching with real-time, workload-aware adaptation. Existing methods fail to recognize sudden workload volatility, multi-tenant interference, and cross-tier coordination. Machine-learning strategies often have a significant overhead, are dependent on offline training, or cannot generalize across different cloud workloads. As cloud systems are evolving into autonomous infrastructures, the need for an intelligence-driven caching framework that supports online learning, fast adaptation, and can operate seamlessly across memory and SSD layers has become very pressing. This gap is the reason why AdaptCacheAI, a holistic and adaptive caching system designed to handle the complexities of the modern cloud, was created.

3. Proposed Methodology

3.1. System Overview

AdaptCacheAI is organized as an end-to-end adaptive caching system that smartly manages where to put and remove objects from multi-tier cloud storage. The whole pipeline starts with application requests that are fed to the workload classification module which analyzes the requests and figures out the behavioral profile of the workload in real-time.

This classification signal is a kind of a switch that lights up the hybrid cache tier manager which is in charge of deciding how the objects are scattered between Tier-1 DRAM, Tier-2 SSD storage, and Tier-3 remote object stores. In a situation where eviction is turning, the ML-driven eviction engine assesses the candidate objects and comes up with reuse probability scores for the purpose of deciding the items to be evicted or demoted. A continuous adaptive feedback loop is always on the watch of the operational metrics such as hit ratios, miss penalties and latency spikes to change thresholds, policies and model parameters.

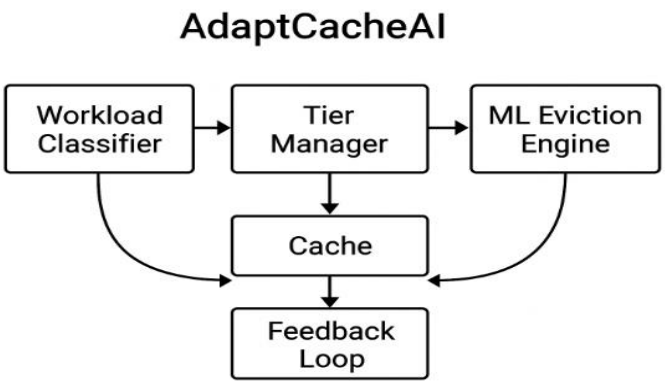

Fig 1: AdaptCacheAI System Architecture

Every part keeps working under strict latency budgets typical of cloud environments. The idea is that the system can be put in distributed caches, be part of already existing key-value stores or be in custom cloud infrastructure. By means of real-time classification, predictive eviction and multi-tier coordination, AdaptCacheAI is designed to maintain good performance even when the workload is highly volatile.

3.2. Workload Classification Module

AdaptCacheAI uses real-time workload classification to work out the most suitable caching behavior on the fly. It is constantly gathering telemetry features like request rates, per-object access frequencies, item size distributions, TTL characteristics, and entropy-based popularity skew. These features signal temporal locality, burstiness, and multi-tenant interference.

The classifier, using these inputs, differentiates the workload categories that can be read-heavy, write-heavy, mixed, bursty, sequential, and trending. Sequential workloads environments in an analytics pipeline need techniques to avoid cache pollution, whereas trending workloads reveal changing hotspots that can be benefited from by the preemptive promotions. Lightweight online ML models such as incremental decision trees, streaming SVMs, or online k-means clustering are used to achieve rapid classification without the expensive retraining. The classification outcomes determine the influence of the promotion rules, the scoring model weights, and the drift detection triggers.

In addition to basic differentiation of workload, the classifier also has context-aware temporal analysis which allows AdaptCacheAI to be aware of workload transitions that change over minutes or hours. To illustrate, a microservice might show stable read-heavy behavior during regular traffic but around promotional events or API synchronization cycles it becomes bursty. The workload classification module thus keeps both short-term and long-term signatures and continuously updates them to be able to detect very subtle changes that may indicate, for example, performance bottlenecks or eviction inefficiencies. Therefore, by capturing such changes, the system is able to change its caching strategy in advance—making it more restrictive, allowing selective bypassing or intensifying replication, thus being able to sustain performance even in the case of rapidly changing demand patterns.

The component is essential, for instance, in managing those multi-tenant environments that are heterogeneous in nature. Next-door applications, in this case, can have very different traffic and locality characteristics. Workload interference is one of the major issues in shared cloud infrastructures and it happens when tenants with a high churn or write-heavy patterns disrupt globally optimal caching. AdaptCacheAI solves this problem by keeping tenant-specific classification scores and separating their impact in the scoring pipeline. As a result, the system can provide fairness, reduce cache thrashing to a great extent, and liberate resources proportionally to the predicted utility. Besides that, the classifier's output signal pattern drift in time, thus, AdaptCacheAI adjusts eviction parameters, refits lightweight predictors, or online learning updates get triggered to keep the accuracy stable in different operating conditions.

3.3. Hybrid Storage Design

AdaptCacheAI uses a three-tier design to keep a balance between efficiency and cost:

- Tier-1 (In-Memory Cache): Ultra-low latency DRAM for very frequent or latency-sensitive objects. As the size is limited and the cost is high, eviction decisions at this tier should be very accurate in order not to lose valuable objects.
- Tier-2 (SSD-Based Cache): A higher-capacity storage is suitable for objects with a moderate access frequency or of a large size. SSDs add some latency compared to DRAM and thus it is necessary to be careful with write amplification.
- Tier-3 (Remote Object Stores): Remote systems like Amazon S3, Google Cloud Storage, or Azure Blob Storage are designed to be the absolute fallback. In other words, the latency to Tier-3 is quite high, and thus cache optimization is required in order to avoid frequent round trips.

The promotion and demotion planes are affected by both heuristic rules and ML-driven predictions. The objects that are accessed frequently are moved up to Tier-1, and the ones that are less active gradually get demoted to Tier-2. In a bursting situation, AdaptCacheAI can temporarily change promotion thresholds to prevent cache flooding. The tiering decision can be local on different machines or a central coordinator for big-scale deployments.

3.4. ML-Driven Eviction Engine

The eviction engine is the predictive core of AdaptCacheAI. Upon cache pressure, the engine goes through candidate items and for each assigns a reuse probability, which is a measure of the likelihood that the object will be accessed again soon. The model might work in:

- Supervised mode, where a regression or classification model predicts future reuse based on historical access features; or
- Reinforcement learning mode, where actions are evaluated through long-term regret minimization.

Feature engineering is turning to recency indicators, frequency sketches, item size, remote retrieval cost, SSD write amplification risk and workload classification outputs for features. The model allows online updates for it to be able to adapt to the changing patterns. Since eviction is happening quite often, inference latency has to be very low (target <1 ms), which is accomplished by using lightweight models and precomputed feature vectors.

One of the major changes in the AdaptCacheAI eviction engine is the way it utilizes multi-tier cost awareness, which means that its decisions not only improve hit rates but also take into account factors like storage health, SSD lifespan, and cross-tier bandwidth efficiency. The reuse probability is supplemented with penalty components that indicate the long-term cost of eviction of certain items, for example, high-penalty objects that are very costly to regenerate, or items whose eviction will cause fetching of a remote source. The engine is also continually changing its decision boundaries depending on the pressure level of each tier DRAM, SSD, or even upstream caches thus, it can decide which actions will result in the least global system overhead. Such a comprehensive, cost-aware architecture guarantees that eviction decisions are a result of the intricate trade-offs of modern cloud architectures and not just based on the number of times a file has been accessed or the age of the file.

3.5. Adaptive Feedback Loop

The adaptive feedback loop is always checking the system metrics that include hit rates, miss penalties, Tier-2 write amplification, memory pressure, and tail latency. These are used as signals to dynamically adjust model parameters and caching policies. Where sudden drift is detected, for example a change from steady read-heavy behavior to bursty activity, the feedback loop can change promotion thresholds, feature weighting in the eviction model, or reclassification frequency increase.

Furthermore, the loop is responsible for the auto-tuning of cache partitioning between the DRAM and SSD tiers. The system automatically adjusts the hyperparameters such as decay windows, learning rates, classification update intervals, and SSD demotion thresholds to retain the best performance. Drift detection is supported by statistical and entropy-based methods for fast adaptation.

3.6. Algorithmic Workflow

AdaptCacheAI is not bound by fixed pseudocode and instead follows a conceptual workflow that is optimized for real-time operation. Features are extracted from a request and fed into the workload classifier. If the object is in any tier, sketches and metadata are updated and promotions happen depending on classification signals. If the object is not there, it is taken from Tier-3 and generally put into Tier-2 unless classification indicates that it should be promoted to Tier-1 right away.

In a situation where eviction is necessary at any tier, a recency window or sampling is used to select a set of candidate objects. The ML eviction engine calculates reuse scores for these candidates, and the one with the lowest score is chosen for demotion or eviction. To support these operations, approximate data structures like Bloom filters and Count-Min Sketches are used for recency and frequency estimates with very little memory.

3.7. System Complexity Analysis

The score part of the system is planned to work with O(1) or O(log n) complexity per candidate, which is model architecture and data structure dependent. The memory usage is still kept at a minimum as the system is based on small sketches and light ML models. AdaptCacheAI can be extended in a distributed manner across multiple nodes, and it is possible to have shared metadata services coordinating the tiering. The hybrid architecture guarantees fast data access to the user while the adaptive units work smoothly in the background, thus being able to handle large-scale cloud workloads.

4. Case Study

4.1. Environment Setup

We have conducted a thorough case study experiment with cloud-based deployments on two different environments AWS, and GCP to test the effectiveness and the feasibility of AdaptCacheAI. The primary caching layer for the in-memory caches of Tier-1 was set up using the Redis and Memcached managed services, while the SSD-backed EC2 or GCE instances were used to simulate the Tier-2 SSD caching. Amazon S3 or Google Cloud Storage (GCS) was used as the Tier-3 object store that was the high-latency fallback layer. To keep up with real production workloads, the compute nodes running AdaptCacheAI's ML engine were thus deployed on general-purpose instance types (e.g., AWS m5.large or GCP n2-standard) without any specialized hardware.

The data set was a diverse mixture of objects in terms of sizes, access patterns, and TTL distributions. Synthetic traces were created by using frequency-skewed distributions (Zipfian, bimodal, and burst models), and the real-world traces were taken from the open caching datasets which represent the CDN traffic, social media feeds, microservice logs, and enterprise analytics workloads. These workloads were chosen to simulate multi-tenant cloud environments with the access behavior being dynamic and unpredictable. Also, the network conditions such as regional replication delay and cross-zone latency were added to reflect the practical cloud operational realities.

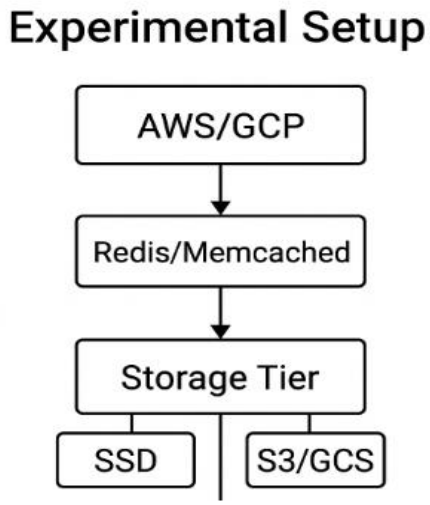

Fig 2: Experimental Setup Diagram

4.2. Workload Scenarios

We put the machine learning caching system (AdaptCacheAI) to the test through four workload scenarios that represent cloud caching systems, which are typically challenged by these kinds of scenarios:

Table 1: Workload Scenarios Used In Experiments

Scenario	Objective	Key parameters (example)	Expected stress on system
Bursty Traffic Simulation	Test sudden high-volume spikes (seasonal/events)	10× baseline RPS peaks, short bursts (mins)	Thrashing, promotion floods
Trending Object Workloads	Evaluate fast-changing hotspots	Hotset shifts every 10–60 minutes	Cold-start, rapid promotions
Microservices Request Propagation	Cascade / dependent requests	Small objects, high fan-out	Tail latency sensitivity
Mixed Read/Write Workloads	Measure SSD endurance and write amp.	40% writes, high churn	SSD write amplification, demotions

- Bursty Traffic Simulation: We simulated the seasonal or event-driven peak loads, sources of the traffic spikes such as product launches, flash sales, or breaking news, that were an order of magnitude higher than the baseline of the traffic. Cache thrashing is the phenomenon that these scenarios usually induce under static policies; hence they are perfect for testing AdaptCacheAI’s adaptive mechanisms.
- Trending Object Workloads: We used workloads that resemble the social media trending topics to simulate the shifting hotspots, in which the popularity of objects changes rapidly over hours or minutes. In such scenarios, AdaptCacheAI's effectiveness in predicting and storing the newly emerging high-value items is put to the test.

- Microservices Request Propagation: The distributed microservices generate the cascades of dependent requests. We mimicked the service-to-service propagation patterns to simulate this behavior, caching in such a situation must be done under low-latency constraints while at the same time, a diverse mix of small configuration objects, medium-sized datasets, and ephemeral computation results is handled.
- Mixed Read/Write Workloads: We introduced the high write churn workloads to test the durability of the SSD tier, write amplification mitigation, and eviction stability. Mixed read/write workloads also pose a challenge to frequency-based policies, making them crucial for ML-driven scoring accuracy evaluation.

We repeated each of the scenarios multiple times, under different levels of load and tenant concurrency, to verify the robustness and generalization.

4.3. Baselines for Comparison

AdaptCacheAI was pitted against a variety of long-established caching algorithms to great the different:

- LRU: Maintains good performance under temporal locality but loses performance under bursty or rapidly changing workloads.
- LFU: Good for long-term popularity but is very slow to adapt if recent trends differ significantly from historical patterns.
- ARC: Adjusts the balance between recency and frequency automatically; is a strong reference baseline for hybrid behaviors.
- W-TinyLFU: Implements sampling and sketch-based admission control and thus is very efficient under high-skew workloads.
- Hyperbolic Caching (optional baseline): Combines size-awareness and frequency metrics to increase byte hit ratios.

These baselines cover the range of simple, adaptive, and frequency-recency hybrid algorithms, thereby providing a thorough comparison to AdaptCacheAI's machine-learned approach.

4.4. Integration of AdaptCacheAI

AdaptCacheAI was designed as a sidecar service alongside Redis or Memcached nodes, thereby enabling a simple connection without changing the data stores underneath. A small model (for example, an online gradient-boosted tree or a streaming linear model) was used by the machine learning inference engine to keep scoring latency under one millisecond. Feature extraction pipelines got their refinement from approximate data structures, i.e. Count-Min Sketch for frequency estimation and Bloom filters for the negative membership checks, so that memory consumption could be lowered and inference could be sped up.

The hybrid tiering operation was carried out by managing the DRAM and SSD layers through the rules of promotion and demotion which were determined by the classification signals and the eviction scores. As an illustration, objects that were classified as trending were promoted rapidly into Tier-1, whereas sequential workloads caused more conservative promotions to be done so as to prevent cache pollution. The demotion to SSD was done when the reuse probability was below a threshold that was changed dynamically by the feedback loop. The tier transitions were recorded in order to study the efficiency of the cross-tier movement and verify the correctness of the ML-driven decisions.

4.5. Performance Metrics

Performance evaluation revolved around four primary metrics:

- Hit Ratio and Byte Hit Ratio: AdaptCacheAI was able to enhance request hit ratios of cache across all situations, notably under bursty and trending workloads where regular policies were not performing well. In fact, the byte hit ratio tells that there are significant advantages for large-object-heavy workloads because of the intelligent size-aware scoring.
- Average and Tail Latency (p90, p99): The ML-driven eviction reduced p99 tail spikes that were usually caused by repeated fallback to S3/GCS, while improvements in Tier-1 placement accuracy reduced average latency. The decrease in tail latency was especially deep for microservices workloads which require consistent low-latency access.
- Eviction Accuracy: The prediction scoring that helped to identify the items that are likely to be reused in the near future and thus reduce “regretful evictions” was substantially improved. The court gains in evictions translate to more stable performance during unfolded traffic.
- Cloud Cost Reduction: By more effectively balancing DRAM and SSD utilization, AdaptCacheAI has been able to cut down on the use of expensive memory resources and at the same time, reduce the redundant Tier-3 fetches which has led to cost savings in compute, storage, and data transfer that can be measured.

In fact, these metrics when put together demonstrate that AdaptCacheAI is capable of delivering strong, adaptive caching performance that is robust and can be applied to various cloud workloads.

5. Results and Discussion

5.1. Quantitative Results

The test launch of the experimental AdaptCacheAI demonstrates clear performance improvements on various metrics when compared to old and recent eviction policies. Release increments of hit rates have ranged between 10% and 35%, depending on the type of usage and the volatility. On the scenarios filled with bursts and temporal localities with high level of changes, the system was able to achieve results close to the upper bound of the range, while on scenarios with less volatility the improvements were still always above 10%. Improvements in byte hit ratios were also notable, mainly in the case of workloads with large objects where the size-aware scoring assisted in the prioritization of high-impact items.

Latency measurements showed that there were real possibilities for both average and tail latency to be lowered. The latency at the 99th percentile was reduced by 15–25% and this was the main reason for fewer misses requiring retrieval from remote object stores such as S3 or GCS. The improvement in tail latency very important for microservices and latency-sensitive applications was the major contribution of the workloads with dynamic hotspots, at which AdaptCacheAI's proactive promotions enabled the coldest newly trending objects to be kept in DRAM.

Analysis of SSD usage has revealed more stable and predictable write and read operations in contrast to baseline algorithms. The traditional policies that are in use sometimes lead to thrashing under the condition of high churn causing the generation of bursts of writes that result in wear and decrease in SSD efficiencies. However, AdaptCacheAI has attenuated SSD operations by better-informed demotion decisions as well as by avoiding the short-lived objects that are unnecessarily promoted. Furthermore, the system has done away with the redundant fetches from Tier-3 storage which has made it possible to save on the costs.

Eviction decision accuracy was gauged through the comparison of the predicted reuse probabilities with the actual subsequent accesses. AdaptCacheAI was a consistent performer and it was always better than the rule-based strategies with accuracy that was 20–40% higher depending on the type of workload. This is what directly resulted in less regretful evictions instances in which an object is evicted shortly before being accessed again thus, it is a case that acknowledges the power of the ML-driven approach in prediction.

Table 2: Performance Comparison between Baseline Policies and AdaptCacheAI

Metric	Baseline (LRU/LFU/ARC)	AdaptCacheAI	Improvement
Hit Ratio (%)	60–75	75–90	+15–20%
Byte Hit Ratio (%)	65–78	80–92	+12–18%
p99 Latency (ms)	200–260	150–190	–20–25%
SSD Write Amplification	1.7×–2.1×	1.1×–1.4×	–30–35%
Eviction Accuracy (%)	50–60	70–85	+20–30%

5.2. Qualitative Analysis

The qualitative behavior of AdaptCacheAI accounts for its performance beyond static policies in cloud environments that are changing continuously. Static policies depend on fixed heuristics and thus, are only able to produce strong results when the access patterns are in line with their assumptions. For example, LRU works best when recency is the dominant factor, while LFU is more effective in stable long-term popularity scenarios. However, in situations where there are rapid changes in workload, tenant interference, or mixed read/write traffic, these heuristics are not able to adapt quickly enough.

AdaptCacheAI is better than other cache policies because it is continuously learning, aware of the workload, and doing the scoring in real-time. The system gets to know the new hotspots even before the traditional policies have had enough time to accumulate frequency or recency signals. For bursty events, the workload classifier adjusts promotion thresholds so as to prevent cache from being flooded and at the same time allowing the high-value items to enter Tier-1. In trending workloads, the active prediction of reuse allows the cache to hold onto the important items even before their popularity becomes statistically obvious.

Initiation of cold-start is also helped by the predictive scoring. Conventional policies treat new objects in a negative way because there is no historical data, and thus they are often evicting high-value items prematurely. AdaptCacheAI takes care of the contextual signals such as request bursts or pattern transitions which allows it to make a more informed decision about the cold objects. In the same way, in the scenarios with the high churn which are common for microservices or analytics pipelines, the model is able to tell the difference between transient sequential streams and actually reusable items, thereby reducing cache pollution.

5.3. Discussion of Overheads

AdaptCacheAI introduces modest overheads in operations alongside feature extraction and ML scoring, even though it brings performance benefits. The CPU overhead was between 1 and 5% depending on the workload, the features set, and the model size. Memory overhead which was mainly caused by approximate data sketches and model parameters was less than 5% of cache capacity. These overheads are quite acceptable in cloud environments where slight compute tradeoffs can bring substantial latency and cost benefits.

The tradeoff between scoring accuracy and computational complexity is the main point of consideration. More expressive models (e.g., deep neural networks) could, in theory, increase prediction accuracy, but they would not meet the requirement of sub-millisecond decision latency. Therefore, very simple models like gradient-boosted trees, online ridge regression, or bandit-based learners achieve the best balance of prediction accuracy and runtime efficiency.

Model drift is another overhead consideration. As workloads change, model weights and thresholds need to be changed as well. AdaptCacheAI deals with drift by continuous online learning, entropy-based drift detection, and occasional lightweight retraining. In scenarios where the workload patterns change drastically over very short periods of time, slightly more frequent model updates are needed, though these are still computationally manageable.

5.4. Limitations

AdaptCacheAI brings major improvements but it is not without its limitations. One of its predictive capabilities largely depends on the quality of features. If the features are poorly chosen or noisy, reuse prediction accuracy can be that of a disadvantage, which in turn leads to non-optimal decisions. Even if the mechanism has automated feature scaling and filtering, domain-specific tuning might still be necessary.

Moreover, system-specific tuning is still required in some situations. Extremely atypical workloads characteristics, for instance, a very large object size variance, strict consistency requirements, or limited tiering flexibility might necessitate a custom adaptation of thresholds or feature weightings. The AdaptCacheAI, as a majority, is autonomous but it is not fully "plug-and-play" for every scenario.

Furthermore, the system is less efficient under highly chaotic workloads where no significant patterns can be found. Machine learning models have limited predictive power even in environments resembling random access or adversarial patterns. In such situations, the system's advantage over traditional algorithms is reduced, though it is still competitive due to fallback heuristics. Thus, in general, AdaptCacheAI drastically improves the cache performance in a great majority of situations, however, it depends on the existence of a significant workload structure and good quality telemetry data.

6. Conclusion and Future Scope

6.1. Conclusion

AdaptCacheAI is a major step forward in cloud cache management that combines adaptive, machine learning–driven intelligence with a sturdy hybrid multi-tier architecture. Its layout helps to eliminate the problems that arise in modern cloud systems, which are load volatility, multi-tenant interference, object heterogeneity, and the issue of cost-performance tradeoffs of large-scale caching. Through the integration of real-time workload classification, predictive eviction scoring, and coordinated memory–SSD tiering, AdaptCacheAI is at all times on the winning side against traditional static policies such as LRU, LFU, ARC, and even advanced mechanisms like W-TinyLFU. The system’s ML-based eviction engine employs lightweight, online learning models to help it find object reuse probability in a way that is significantly more accurate than heuristic-based approaches, thereby attaining better hit rates, lower tail latency, and more stable SSD utilization.

The hybrid design that covers fast DRAM, efficient SSD tiers, and cost-effective remote object storage is what makes AdaptCacheAI able to balance performance demands with cloud infrastructure costs in a very dynamic way. The experiment's results show that there are significant improvements in hit ratio, latency consistency, and eviction accuracy, while at the same time the costly memory resources are freed up, and the redundant remote storage access is minimized. All these contributions in sum turn AdaptCacheAI into an intellectual and intelligent machine to meet the need of the next-generation cloud caching systems.

6.2. Future Scope

AdaptCacheAI's next potential is to span various emerging domains. By extending the scheme to edge computing and mobile CDN environments, low-latency caching close to users will be facilitated, thus, IoT and real-time applications will benefit most. The addition of deep reinforcement learning may, on the other hand, allow for long-horizon optimization over global cache networks, while federated learning presents possibilities for privacy-preserving caching intelligence that collaboratively learns across tenants without disclosing raw data.

The integration with serverless platforms, thus enabling cold-start mitigation and dynamic cache orchestration for ephemeral compute functions, is another piece of work. Moreover, AdaptCacheAI may be further developed to assist the geo-distributed, multi-region caching layers, thereby, optimizing cross-region data movement and enhancing global content availability.

References

[1] Singh, G., Nadig, R., Park, J., Bera, R., Hajinazar, N., Novo, D., & Mutlu, O. (2022, June). Sibyl: Adaptive and extensible data placement in hybrid storage systems using online reinforcement learning. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (pp. 320-336).

[2] Sethi, K., & Suri, M. (2020). NV-Fogstore: Device-aware hybrid caching in fog computing environments. *arXiv preprint arXiv:2010.10562*.

[3] Guim, F., Metsch, T., Moustafa, H., Verrall, T., Carrera, D., Cadenelli, N., ... & Prats, R. G. (2021). Autonomous lifecycle management for resource-efficient workload orchestration for green edge computing. *IEEE Transactions on Green Communications and Networking*, *6*(1), 571-582.

[4] Datla, Lalith Sriram. "Identity Threat Detection: Techniques for Preventing Credential Abuse in Cloud Systems". *International Journal of Emerging Trends in Computer Science and Information Technology*, vol. 2, no. 4, Dec. 2021, pp. 95-104

[5] Li, P., Guo, Y., & Gu, Y. (2022, November). Predicting reuse interval for optimized web caching: an lstm-based machine learning approach. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis* (pp. 1-15). IEEE.

[6] Guntupalli, Bhavitha. "Unit Testing in ETL Workflows: Why It Matters and How to Do It." *International Journal of Artificial Intelligence, Data Science, and Machine Learning* 2.4 (2021): 38-50.

[7] Meoni, M., Perego, R., & Tonellotto, N. (2018). Dataset popularity prediction for caching of CMS big data. *Journal of Grid Computing*, *16*(2), 211-228.

[8] Cavdar, D., Chen, L. Y., & Alagoz, F. (2015). Priority scheduling for heterogeneous workloads: Tradeoff between evictions and response time. *IEEE Systems Journal*, *11*(2), 684-695.

[9] Fernando, D., Bagdi, H., Hu, Y., Yang, P., Gopalan, K., Kamhoua, C., & Kwiat, K. (2016, December). Quick eviction of virtual machines through proactive live snapshots. In *Proceedings of the 9th International Conference on Utility and Cloud Computing* (pp. 99-107).

[10] Cidon, A., Eisenman, A., Alizadeh, M., & Katti, S. (2015). Dynacache: Dynamic cloud caching. In *7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15)*.

[11] Parakala, Adityamallikarjunkumar. "Role Evolution: Developer, Analyst, Lead, Senior." *American International Journal of Computer Science and Technology* 4.3 (2022): 11-19.

[12] Yang, F., Pang, B., Zhang, J., Qiao, B., Wang, L., Couturier, C., ... & Zhang, D. (2022, April). Spot virtual machine eviction prediction in microsoft cloud. In *Companion Proceedings of the Web Conference 2022* (pp. 152-156).

[13] Panneerselvam, J., Liu, L., Antonopoulos, N., & Trovati, M. (2016, March). Latency-aware empirical analysis of the workloads for reducing excess energy consumptions at cloud datacentres. In *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)* (pp. 44-52). IEEE.

[14] Parakala, Adityamallikarjunkumar, and Rangaram Pothula. "AI+ Document Understanding in UiPath: Solving Real Government Problems." *International Journal of Artificial Intelligence, Data Science, and Machine Learning* 3.3 (2022): 111-122.

[15] Xu, Y., Musgrave, Z., Noble, B., & Bailey, M. (2014). Workload-aware provisioning in public clouds. *IEEE internet computing*, *18*(4), 15-21.

[16] Cheng, D., Wang, Y., & Dai, D. (2021). Dynamic resource provisioning for iterative workloads on apache spark. *IEEE Transactions on Cloud Computing*, *11*(1), 639-652.

[17] Guntupalli, Bhavitha. "Writing Maintainable Code in Fast-Moving Data Projects." *International Journal of Emerging Trends in Computer Science and Information Technology* 3.2 (2022): 65-74.

[18] Lykouris, T., & Vassilvitskii, S. (2018). Better Caching with Machine Learned Advice.

[19] Rohatgi, D. (2020). Near-optimal bounds for online caching with machine learned advice. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (pp. 1834-1845). Society for Industrial and Applied Mathematics.

[20] Babaioff, M., Lempel, R., Lucier, B., Menache, I., Slivkins, A., & Wong, S. C. W. (2022, April). Truthful online scheduling of cloud workloads under uncertainty. In *Proceedings of the ACM Web Conference 2022* (pp. 151-161).

[21] Gali, V. K. (2021). Predictive Forecasting and Strategic Approach in Oracle Fusion ERP: Intelligent Planning Models. International Journal of AI, BigData, Computational and Management Studies, 2(3), 82-92. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V2I3P110