Machine Learning-Enhanced CLOCK Cache Replacement Policy for Improved Efficiency

Muhammad Haekal Muhyidin Al-Araby
Department of Computer Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
5024221030@student.its.ac.id

Reza Fuad Rachmadi, S.T., M.T., Ph.D
Department of Computer Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
fuad@its.ac.id

Dr. Arief Kurniawan, S.T., M.T.
Department of Computer Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
arifku@ee.its.ac.id

***Abstract*—This paper investigates the integration of machine learning into the CLOCK cache replacement algorithm to enhance its efficiency by reducing unnecessary object promotions. Modern computing demands advanced cache management techniques, and machine learning offers a promising avenue to optimize existing algorithms. We developed a custom simulation software based on LibCacheSim to collect data from various real-world traces and synthetic Zipfian traces. A logistic regression model was trained to predict the likelihood of a wasted promotion based on object metadata. This model, exported in ONNX format and run with ONNX Runtime, was integrated into the CLOCK algorithm. Our experimental results demonstrate that the machine learning-enhanced CLOCK algorithm significantly reduces the total number of promotions by approximately 10% without a substantial increase in the miss ratio. This indicates improved cache management efficiency, as the system intelligently avoids promoting objects that are unlikely to be re-accessed, thereby preserving valuable cache space for more relevant data. The findings suggest that even lightweight machine learning models can provide meaningful performance improvements in classical cache algorithms, paving the way for more adaptive and efficient cache systems.**

***Index Terms*—cache replacement, CLOCK algorithm, machine learning, logistic regression, ONNX, LibCacheSim.**

I. Introduction

The rapid advancement in computing technology, particularly the shift towards multi-core architectures, necessitates sophisticated techniques for object cache management. While traditional cache replacement policies like Least Recently Used (LRU) and First-In First-Out (FIFO) have been widely used, they present challenges in multi-core environments due to locking overheads and suboptimal eviction decisions. The CLOCK algorithm, a practical approximation of LRU, offers a balance between low miss ratio and high scalability by using a circular buffer and reference bits. However, CLOCK still suffers from unnecessary promotions, where frequently accessed objects prevent less relevant items from being evicted, leading to inefficient cache utilization.

This work was supported by [Funding Agency, if any].

This research explores the integration of machine learning (ML) into the CLOCK algorithm to mitigate these unnecessary promotions. By leveraging ML models to predict the utility of an object's promotion, we aim to enhance the CLOCK algorithm's decision-making process, making it more adaptive and efficient. The primary goal is to reduce wasted promotions without significantly increasing the cache's miss ratio, thereby improving overall system performance. This paper details the methodology, implementation, and evaluation of a machine learning-enhanced CLOCK cache replacement policy.

II. Related Work

The CLOCK algorithm, developed in the early 1970s, has been a cornerstone in page-replacement systems due to its lower overhead compared to LRU. Subsequent advancements include Clock with Adaptive Replacement (CAR), which dynamically balances recency and frequency using two separate lists, and Clock-Pro, which categorizes objects into Hot, Cold, and Test lists for more nuanced eviction decisions. These variations aim to achieve LRU-like performance with FIFO-like overheads.

The application of machine learning in caching has gained traction. Previous studies have explored various ML techniques to predict object re-access probability or optimize cache policies. Our work builds upon this by specifically targeting the reduction of wasted promotions within the CLOCK algorithm using a lightweight supervised learning approach. We aim to demonstrate that even a simple ML model can significantly improve the efficiency of a well-established cache replacement policy.

III. Methodology

Our research involved developing a custom cache simulation software, collecting and processing data, developing a machine learning model, and integrating it into the simulator.

A. Simulation Software Development

A custom simulation software was developed using **LibCacheSim**, a C/C++ library for cache simulation. This software was designed to:

1) Integrate machine learning models efficiently within the cache's eviction function.
2) Support standard CLOCK and Offline CLOCK algorithms for performance comparison.
3) Collect crucial data from Offline CLOCK simulations, including miss ratio, promotion counts, and object metadata with 'wasted' or 'not wasted' labels for each promotion decision. This data was exported to CSV for further analysis.

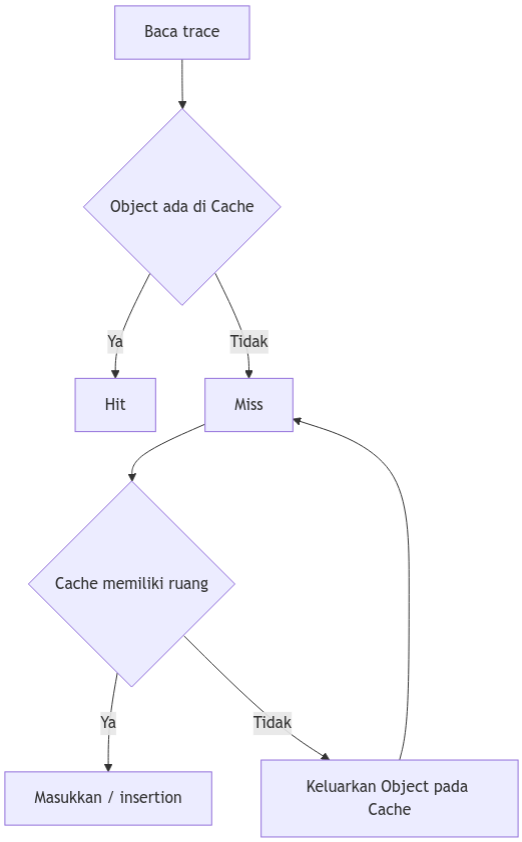

Fig. 1. Flowchart of the developed simulation program.

B. Data Collection and Processing

Initial data collection involved running standard CLOCK simulations to establish baseline performance metrics. Concurrently, a dataset for ML model training was generated by logging object metadata at the point of promotion decisions and labeling them based on Offline CLOCK's insights into wasted promotions. We utilized open-source traces from various real-world workloads (e.g., BlockStore, KVStorage, CDN, Proxy, Photo Storage from Twitter, Meta, Tencent, Alibaba) and synthetic Zipfian traces. Python scripts with Plotly and Pandas were used for data processing, visualization, and interactive HTML reporting.

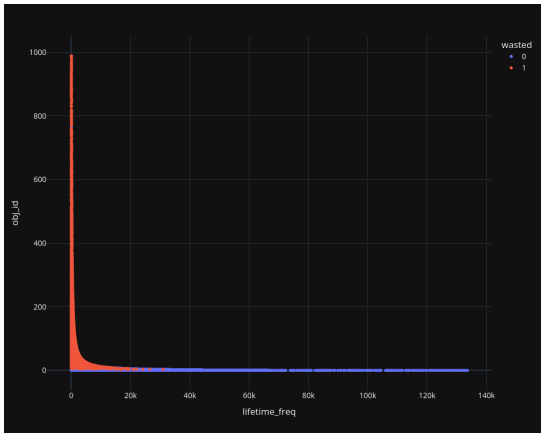

Fig. 2. Example visualization of `lifetime_freq` metadata distribution.

C. Machine Learning Model Development and Integration

A logistic regression model was developed using the processed dataset. The model takes object metadata (e.g., access frequency, time since last access, object size) as input and outputs a probability (0-1) indicating the likelihood of a wasted promotion. Higher values suggest a higher probability of a wasted promotion.

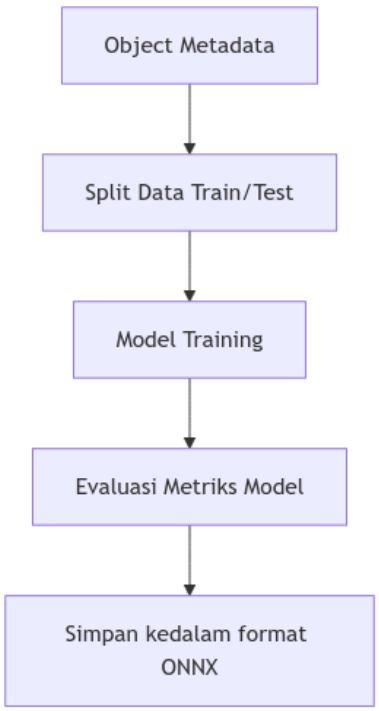

Fig. 3. Diagram of the machine learning model training process.

The trained model was then integrated into the C++ simulation environment. The model was exported to **ONNX (Open**

Fig. 4. Architecture of the machine learning model used.

Neural Network Exchange) format and executed within the simulator using **ONNX Runtime**. During the CLOCK algorithm's eviction process, before an object is promoted, its metadata is fed into the ML model. If the model's output exceeds a predefined threshold, the object is immediately evicted instead of being promoted, thus preventing a potentially wasted promotion.

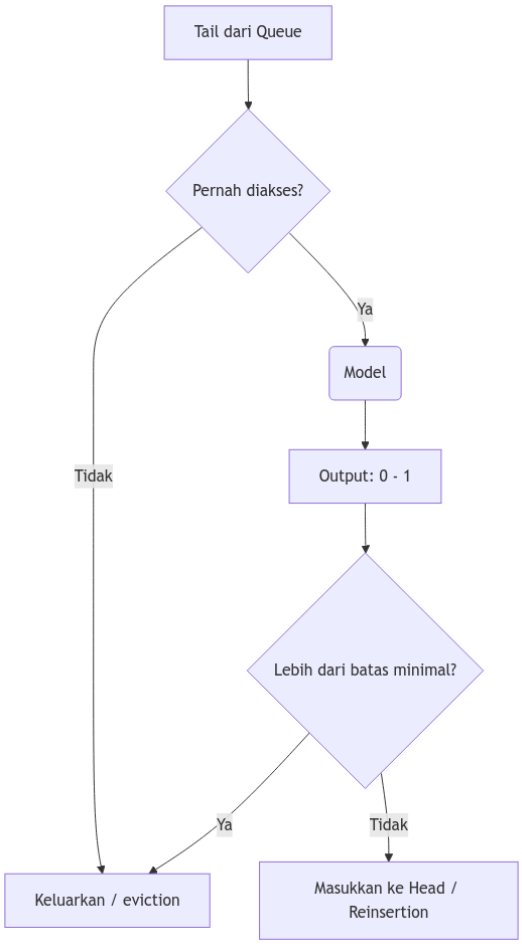

Fig. 5. Flowchart of model implementation in the simulator.

D. Evaluation

The performance of the ML-enhanced CLOCK algorithm was compared against the standard CLOCK algorithm and the theoretical upper bound of Offline CLOCK. Key metrics for comparison included the total number of promotions and the miss ratio.

IV. Results and Analysis

Our primary expectation was that the machine learning model would significantly reduce the total number of object promotions within the CLOCK algorithm without substantially increasing the miss ratio. This would indicate a higher efficiency in cache management by eliminating unnecessary promotions while retaining relevant data.

A. Offline CLOCK Experiment Results

Initial experiments with the Offline CLOCK algorithm on the Meta Key-Value trace established a performance upper bound. As shown in Figure 6, this simulation revealed a drastic reduction in both promotion count and miss ratio. This significant decrease strongly suggests that the standard CLOCK algorithm performs numerous wasted promotions, which not only consume computational resources but also "pollute" the cache with objects unlikely to be re-accessed.

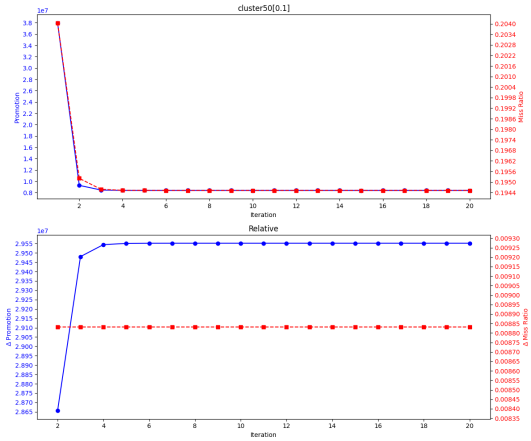

Fig. 6. Results from Offline CLOCK simulation on Meta Key-Value trace.

B. Initial Model Development Results

For model development, we employed logistic regression due to its simplicity and interpretability. The model's weights and threshold were tuned for optimal performance on validation data. Figure 7 illustrates the comparison of results after applying the model.

Figure 7 demonstrates a significant reduction in the average number of promotions, approximately **10%**, while the miss ratio remained stable and largely unaffected. These promising initial results align with our research objective: to reduce promotions without compromising the hit ratio. This indicates that the ML-enhanced CLOCK algorithm effectively identifies and prevents wasted promotions, leading to more efficient cache utilization.

V. Conclusion

This research successfully demonstrated that integrating a machine learning model into the CLOCK cache replacement algorithm significantly enhances cache management efficiency. The developed logistic regression model effectively identified

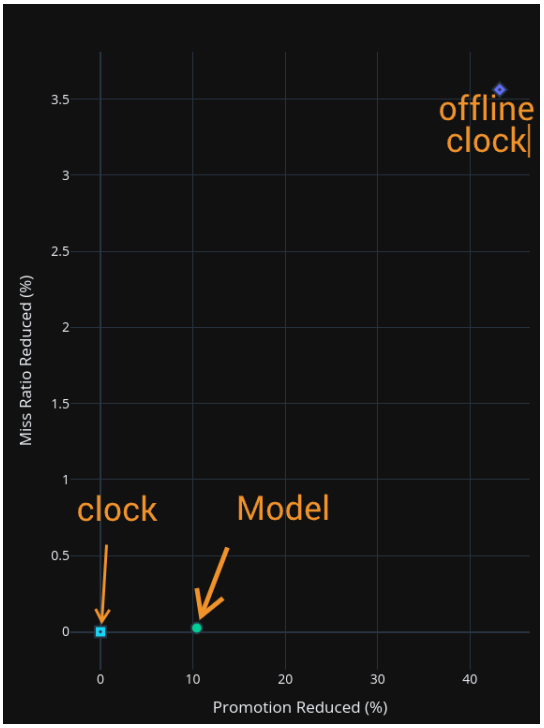

Fig. 7. Comparison of results from the logistic regression model.

and prevented unnecessary object promotions, leading to an approximate 10% reduction in total promotions on test data. Crucially, this reduction was achieved without a significant negative impact on the system's miss ratio, thus maintaining the cache's primary function of serving data requests efficiently.

Our findings highlight that even relatively simple and lightweight machine learning models can yield substantial performance improvements in classical cache algorithms. This opens up opportunities for similar implementations in real-world systems where computational overhead is a critical consideration. Future work could explore more complex ML architectures, such as neural networks or ensemble models, investigate reinforcement learning approaches for adaptive eviction policies, and validate the proposed algorithm in real production environments.

ACKNOWLEDGMENT

The authors would like to thank [Optional: individuals or groups who provided support, e.g., for data, discussions, or infrastructure].

REFERENCES