International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-10 Issue-6, August 2021

Machine Learning-Based Cache Replacement Policies: A Survey

Pratheeksha P, Revathi SA

Abstract: ***Despite extensive developments in improving cache hit rates, designing an optimal cache replacement policy that mimics Belady's algorithm still remains a challenging task. Existing standard static replacement policies does not adapt to the dynamic nature of memory access patterns, and the diversity of computer programs only exacerbates the problem. Several factors affect the design of a replacement policy such as hardware upgrades, memory overheads, memory access patterns, model latency, etc. The amalgamation of a fundamental concept like cache replacement with advanced machine learning algorithms provides surprising results and drives the development towards cost-effective solutions. In this paper, we review some of the machine-learning based cache replacement policies that outperformed the static heuristics.***

Keywords: Belady's algorithm, Cache Replacement, Machine Learning

I. INTRODUCTION

With the rapid advancement in the field of high-speed processors and memory hierarchy, caches prove to be promising mechanisms in reducing the memory access latency. The performance gains shown by caches are one of the key reasons for their inclusion in most systems [1]. Caches are small (of about few MBs) and high-speed memory units to hold frequently used data. Cache performance can be enhanced by increasing cache size but is expensive. General approach is to increase the cache hit rates by data and instruction prefetching to prevent future cache misses and effective cache replacement strategy to judiciously discard cache items to make room for new entries. In this paper, we focus on various cache replacement policies and their performance. Several static cache replacement algorithms have been developed, but they are restricted to a subset of access patterns and perform poorly in complex circumstances which leads us to the discussion of modern technologies to advance in such challenging areas.

Machine learning and deep learning have shown remarkable improvements in the field of natural language processing, computer vision, speech recognition and time series analysis [2]. The use of these modern and powerful tools in computer architecture is not new and has been extensively studied to improve certain areas such as branch prediction, cache replacement, data prefetching [3]. However, there are few challenges when it comes to applying it to hardware predictors. Training a neural network consumes enormous amounts of data and resources and therefore necessitates offline training. But for a wide range of computer programs and with dynamic changes exhibited by the access patterns, the offline model proves to be less effective. Deploying the offline model on a hardware chip constrained by memory is another key issue. For time-critical systems, the model's prediction time can become a hindrance. Apart from these, some approaches require hardware modifications and may result in additional overheads. Nonetheless, machine-learning based cache replacement techniques have significantly outperformed the static heuristics and can be considered as a feasible solution to improve overall system performance and performance scaling in case of a multithreaded environment [4]. In this paper, we explore some of the recent cache replacement policies based on machine learning

II. BACKGROUND

Cache replacement policies are heuristics that evicts a data entry present in the cache to account for the new entry being fetched. The primary objective is to replace those entries that are least likely to be accessed sooner rather than later. The most optimal and efficient algorithm that always evicts data that will no longer be required in the near future is termed as Belady's algorithm. This is infeasible to be practically implemented as forecasting the future is difficult. Thus, any replacement algorithm should strive to closely resemble Belady's algorithm. Here, we review some of the common conventional cache replacement policies [5]:

First in first out (FIFO): The simplest of all policies which evicts the block in the order in which it was cached [6].

Random replacement (RR): This algorithm takes a data item from the cache at random and replaces it with the desired one. Both FIFO and RR policies do not take into account the history of the cache contents and hence costs less than the competing algorithms [7].

Manuscript received on May 24, 2021.
Revised Manuscript received on July 19, 2021.
Manuscript published on August 30, 2021.
* Correspondence Author
Pratheeksha P*, Student, Department of Computer Science, RV College of Engineering, Bangalore, India. Email: pratheekshap.cs17@rvce.edu.in
Revathi SA, Assistant Professor, Department of Computer Science, RV College of Engineering, Bangalore, India. Email: revathisa@rvce.edu.in

Least recently used (LRU): One of the most widely used algorithms, with a growing number of variants. As the name suggests, this caching algorithm evicts those entries which are least recently used by keeping track of the history of contents. This is implemented using aging bits and can be expensive as the state of the aging bits of every cache line changes after every cache reference [8].

Pseudo LRU (PLRU): A variant of LRU in which the core principle remains the same but the age of the cache block is approximated rather than maintained as an exact value. It can be implemented in two ways namely, Tree-PLRU and Bit-PLRU. Tree-PLRU constructs a binary search tree where each node contains a flag that aids in traversing the tree to find the PLRU element. Bit-PLRU maintains one bit called MRU-bit for each cache line. The bit is set to 1 if a reference is made to that line and whenever that last zero bit is set to 1, the remaining bits are set to 0. During cache misses, the algorithm replaces the leftmost line whose bit is 0. PLRU is considered to be better than LRU in terms of low power consumption and low overheads. However, it has the worst miss ratio.

Segmented LRU (SLRU): This technique depends on the cache design and is divided into protected and probationary segments [9]. Fetched data is first cached in the probationary segment and with the next hit is moved to the protected segment. Since the size of the protected segment is fixed, the line at LRU end of protected segment is moved to MRU end of probationary segment when the protected segment is full. SLRU replaces the LRU end of the probationary segment.

Least frequently used (LFU): This caching algorithm is based on the frequency of access and entries with least frequencies are evicted. The eviction of entries with the same frequency are done arbitrarily.

CLOCK: LRU uses a single global lock to serialise cache hits. The CLOCK method seeks to alleviate lock contention and emulates LRU, resulting in increased concurrency and throughput [10].

Adaptive replacement cache (ARC): The caching algorithm is a hybrid of LRU and LFU that serves as an adaptive filter for tracking temporal locality and offers the advantage of both recency and frequency. Its implementation is similar to LRU but outperforms LRU tremendously and is scan-resistant in nature [11].

Clock with adaptive replacement (CAR): By adopting the features of ARC and CLOCK, this caching technique achieves high performance with low overheads [12].

While the list of standard replacement policies and their variants is extensive, the algorithms described above provide a solid overview for understanding and evaluating machine learning-based cache replacement techniques.

III. ML BASED CACHE REPLACEMENT POLICIES

Commonly used machine learning techniques to enhance cache replacement are reinforcement learning (RL) [13] and recurrent neural networks (RNNs), particularly long short - term memory (LSTMs) [14]. RL can be viewed as a typical cache replacement solution where a sequence of actions that serves past cache access are assessed and learnt to generate a new policy. In any intermediate state, no action is regarded as the best; an action is deemed to be good if it leads to a good policy. LSTMs are neural networks that learn sequential data in order to predict the output at the next time step. In the context of cache replacement, the history of cache accesses forms the sequential data. One way of classifying ML-based cache replacement policies are PC-based (program counter), non PC-based and models trained using traditional static heuristics as experts. Some of these techniques are discussed below:

A. LeCAR

LeCAR is an ML based cache replacement algorithm designed for small cache sizes (relative to the workload) that exploits the benefits offered by the well-known static heuristics, LRU and LFU [15]. Every cache miss is served by either of the policies based on the probability distribution of weights of LRU and LFU. LeCAR framework is modelled as a reinforcement learning problem with regret minimization. The fundamental principle of regret minimization is “sometimes regretting is a good way to improve”. Regret refers to the course of action that should have been adopted.

LeCAR maintains a FIFO queue that holds the recent evictions (history) from the cache. Every entry in the queue is labelled by the policy that led to its eviction i.e, LRU or LFU. If a reference made is found in the queue/history, the regret associated with the policy is increased and the weights of the other policy is updated indicating that a better decision could have been made. It’s an online model where the model learns after every miss to minimize regret. When evaluated against ARC, LeCAR consumes 3x the amount of space yet outperforms it by 18 times with a cache size of (1/1000)th the working set.

B. CACHEUS

CACHEUS is the adaptive version of LeCAR that adopted a gradient-based stochastic hill climbing approach to compute the learning rate. While LeCAR proved to be efficient for only certain types of workloads (LRU and LFU friendly), CACHEUS was designed to accommodate other working sets, scan and churn [16]. Scan refers to a set of cache entries that are accessed only once. Churn refers to a set of cache entries that are accessed repeatedly with equal probability. CACHEUS first used state-of-the-art caching algorithms LFU, LIRS, and ARC as experts, similar to how LeCAR picked two policies, LRU and LFU, to make the eviction decision. However, the performance across a wide range of workloads were enhanced with the design of a novel adaptive scan resistant LRU (SR-LRU) and churn resistant LFU (CR-LFU).

When tested for webmail workloads, cache hit rates of ARC, LIRS, LeCAR were 30.08%, 40.71% and 42.08%, while CACHEUS with SR-LRU and CR-LFU as experts showed a cache hit rate of 43.95% and proved to be the most consistent algorithm [16].

C. Hawkeye

Hawkeye formulates cache replacement as a binary classification problem where the cache line is classified as cache-friendly or cache-averse [17].

Hawkeye consists of 3 main components namely OPTgen, Hawkeye predictor and a sampler. OPTgen can be viewed as a simulator which effectively emulates Belady's algorithm to generate inputs. Inputs basically represent the history of memory accesses. If OPTgen identifies a particular line as a cache hit, then any PC that access those lines are considered as positive examples, and otherwise negative. These examples are used to train the Hawkeye predictor/binary classifier based on Belady's optimal policy. If the classifier computes cache-friendly, the lines are marked as high priority while cache-averse lines are flagged as eviction/replacement candidates. The most difficult aspect of this method was in constructing OPTgen which was addressed using a novel approach called liveness intervals. Another challenge imposed by OPTgen was the long history requirement to compute optimal decisions which was resolved using the Set Dueling sampler by sampling a subset of cache lines that was sufficient to mirror optimal cache and reduce storage requirements. For SPEC 2006 CPU benchmarks, Hawkeye reduces miss rates over LRU by 8.4% with 2MB LLC and by 15% for four core systems with 8MB LLC [17].

D. Glider

Glider cache replacement policy is built on Hawkeye and significantly improves prediction accuracy. Glider is modelled as a sequence labelling problem where each memory access in a sequence is assigned a binary label [18]. The input sequence is a series of loads represented by their program counters (PC). The objective of the learning model is to predict whether a particular PC accesses cache lines that are cache-friendly or cache-averse.

Three step approach - offline caching model, offline analysis and an online model is employed. Offline model is based on LSTM with attention mechanisms to predict the important program counters in the input sequence. The weights are then analyzed and input feature is encoded compactly as caching decisions does not depend on the long history of input sequences but only on a few memory accesses. These insights are used to train a SVM based online model to predict important PCs and the accuracy was comparable to the offline LSTM model. The reason to build an SVM model on encoded input features was that the LSTM was large and slow and could not be trained or deployed on hardware predictors. For SPEC 2006 and 2017 programs, Glider reduces the miss rate by 8.9% over LRU in a single core configuration, whereas Hawkeye only reduces it by 6.5 percent. Glider reduces the miss rate over LRU by 14.7 percent in a multicore scenario, whereas Hawkeye reduces it by 13.6 percent [18].

E. Reinforcement Learning Replacement (RLR)

While Hawkeye and Glider are two of the most effective PC-based predictors, it is critical to have cost-effective replacement strategies with little overhead and hardware upgrades. In RLR, initially, a set of target features were derived by training a reinforcement learning (RL) agent and hill climbing analysis [19]. The features preuse distance, line last access type, line hits since insertion and line recency were selected based on the neural network weights. The process of feature selection was completely automated and allowed the RL agent to be adaptive to dynamic changes in access patterns. A replacement policy was then designed using these limited features by assigning priority levels to cache lines. PC was intentionally excluded from the feature set as it adds to hardware complexity as well as communication overhead incurred in transmitting PC data to LLC. RLR outperformed the existing non-PC based predictor DRRIP. With negligible overhead, RLR enhances single-core and four-core system performance by 3.25 percent and 4.86 percent, respectively, over LRU [19].

F. PARROT

PARROT is the first method to construct an end -to-end cache replacement policy using imitation learning that approximates Belady's [20]. The algorithm begins by converting the input i.e, cache accesses (embedded memory address and embedded PC) into states. To minimize compounding errors caused by the difference in state distribution during train and test periods, the conversion adopts the DAgger approach. The states are sampled and initialised as hidden states in the LSTM model, which is subsequently trained using the BPTT algorithm. Applying the initialised replacement policy to the remaining states yields the loss function. Loss function is given by the sum of rank loss and loss incurred while predicting the reuse distance. The weight parameters are updated based on this loss function, allowing the replacement strategy to be learned. For PARROT, the number of previous accesses to be considered to accurately approach Belady's was discovered to be 80, after which the improvement saturates. PARROT improves cache hit rates by 61 percent over a traditional LRU policy for the web search benchmark. Unfortunately, the practical application of this model is limited due to its large size and high latency.

IV. CONCLUSION

Cache replacement policies are constantly evolving with the aim to approach theoretical results defined by Belady's algorithm. The trend is towards applying ML/DL to solve cache replacement problems. The goal while designing any cache replacement policy must be to build cost-effective solutions with little hardware modifications, reduce off-chip bandwidth needs and minimize overheads. The algorithms must also strive towards reducing the model size and low latency so that the benefits offered by the policies are not obscured which would eventually enable practical deployment.

The paper starts with a discussion on some of the conventional cache replacement algorithms to facilitate easy interpretation of the machine learning based techniques. A survey on numerous ML based cache replacement algorithms showed significant enhancement over conventional methods but each policy is optimized to cater to only certain types of workloads. Designing a well generalized policy that works for all workloads still remains a challenging task. Furthermore, most of these techniques are developed for only single level caches leaving the development of hierarchical level cache replacement policies as the hot research topic in ML.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication

REFERENCES

1. Swadhesh Kumar, P K Singh, "An overview of modern cache memory and performance analysis of replacement policies", IEEE International Conference on Engineering and Technology (ICETECH), Coimbatore, India, Mar 2016, pp. 210-214
2. Sheena Angra, Sachin Ahuja, "Machine learning and its applications: A review", International Conference on Big Data Analytics and Computational Intelligence (ICBDAC), Chirala, Andhra Pradesh, India, Mar 2017, pp. 57-60
3. Nan Wu, Yuan Xie, "A Survey of Machine Learning for Computer Architecture and Systems", arXiv preprint arXiv: 2102.07952 , Feb 2021.
4. Rashidah F. Olanrewaju, Asifa Baba, Burhan Ul Islam Khan, Mashkuri Yaacob, Amelia Wong Azman, Mohammad Shuaib Mir, "A study on performance evaluation of conventional cache replacement algorithms: A review", 4th International Conference on Parallel, Distributed and Grid Computing (PDGC), Waknaghat, India, Dec 2016, pp. 550-556
5. Qaisar Javaid, Ayesha Zafar, Muhammad Awais, Munam Shah, "Cache Memory: An Analysis on Replacement Algorithms and Optimization Techniques", Mehran University Research Journal of Engineering & Technology, vol. 36, no. 4, pp. 831-840, Oct 2017
6. Wang, Q., "WLRU CPU Cache Replacement Algorithm", Doctoral Dissertation, The University of Western Ontario London, 2006
7. Bhattacharjee, A., and Debnath, B.K., "A New Web Cache Replacement Algorithm", Proceedings of IEEE Conference on Pacific Rim Conference on Communications, Computers and Signal Processing, pp. 420-423, 2005.
8. Ahmed, M.W, Shah, M.A., "Cache Memory: An Analysis on Optimization Techniques", International Journal of Computer and IT, vol. 4, no. 2, pp. 414-418, 2015
9. Gao, H, Wilkerson C, "A Dueling Segmented LRU Replacement Algorithm with Adaptive Bypassing", 1st JILP Worshop on Computer Architecture Competitions: Cache Replacement Championship, 2010
10. Andhi Janapsatya, Aleksandar Ignjatović, Jorgen Peddersen, Sri Parameswaran, "Dueling CLOCK: Adaptive cache replacement policy based on the CLOCK algorithm", Design, Automation & Test in Europe Conference & Exhibition (DATE 2010), Dresden, Germany, Apr 2010, pp. 920-925.
11. Nimrod Megiddo, Dharmendra S. Modha, "ARC: A self-tuning, low overhead replacement cache", in Proc. 2nd USENIX Conference on File and Storage Technologies, San Francisco, USA, Mar 2003, pp. 115-130.
12. Sorav Bansal, Dharmendra S. Modha, "CAR: Clock with Adaptive Replacement", in Proc. 3rd USENIX Conference on File and Storage Technologies, San Francisco, USA, Mar 2004
13. Wang Qiang, Zhan Zhongli, "Reinforcement learning model, algorithms and its application", International Conference on Mechatronic Science, Electric Engineering and Computer (MEC), Jilin, China, Aug 2011, pp. 1143 – 1146
14. Alex Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network", arXiv preprint arXiv: 1808.03314v9 , Aug 2018.
15. Giuseppe Vietri, Liana V. Rodriguez , Wendy A. Martinez, Steven Lyons, Jason Liu, Raju Rangaswami, Ming Zhao, Giri Narasimhan, "Driving Cache Replacement with ML-based LeCaR", in Proc. 10th USENIX Conference on Hot Topics in Storage and File Systems, Berkeley, CA, United States, Jul 2018
16. Liana V. Rodriguez, Farzana Yusuf, Steven Lyons, Eysler Paz, Raju Rangaswami, and Jason Liu, Ming Zhao, Giri Narasimhan, "Learning Cache Replacement with Cacheus", in Proc. 19th USENIX Conference on File and Storage Technologies, Santa Clara, USA, Feb 2021, pp. 341-354
17. Akanksha Jain, Calvin Lin, "Leveraging Belady's Algorithm for Improved Cache Replacement", in Proc. ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), Seoul, Korea, Jun 2016, pp. 78-89
18. Zhan Shi, Xiangru Huang, Akanksha Jain, Calvin Lin, "Applying Deep Learning to the Cache Replacement Problem", in Proc. 52nd Annual IEEE/ACM International Symposium on Microarchitecture, Columbus, USA, Oct 2019, pp. 413–425
19. Subhash Sethumurugan, Jieming Yin, John Sartori," Designing a Cost-Effective Cache Replacement Policy using Machine Learning", in Proc. IEEE International Symposium on High-Performance Computer Architecture (HPCA), Seoul, Korea, Apr 2021, pp. 291-303
20. Evan Zheran Liu, Milad Hashemi, Kevin Swersky, Parthasarathy Ranganathan, Junwhan Ahn, "An Imitation Learning Approach for Cache Replacement", arXiv preprint arXiv:2006.16239, Jul 2020.

AUTHORS PROFILE

Pratheeksha P, is a data science enthusiast with a strong desire to solve complex system architecture challenges in critical applications with a real-world impact. She is currently pursuing her BE in Computer Science from RV College of Engineering. Bangalore. Her technical interests include Deep Learning, Cloud Computing and Big Data Analytics and has worked on multiple projects, Email: pratheekshap.cs17@rvce.edu.in

Revathi S A, has over 8 years of teaching and research experience. Presently, she is an Assistant Professor at RV College of Engineering, Bangalore and is focused in educating her students about cutting-edge technology through an application-oriented approach while simultaneously emphasizing moral development. She has guided multiple UG projects and publications in national and international conferences. Her research interests include Machine Learning, Digital Image Processing, Medical Imaging and Computer Networks, Email: revathisa@rvce.edu.in