

IMPROVING CACHE REPLACEMENT WITH MACHINE LEARNING

K.S Devinda

229314J

Masters in Computer Science

Department of Computer Science
Faculty of Engineering

University of Moratuwa
Sri Lanka

June 2024

IMPROVING CACHE REPLACEMENT WITH MACHINE LEARNING

K.S Devinda

229314J

Dissertation submitted in partial fulfillment of the requirements for the
degree
Masters in Computer Science

Department of Computer Science
Faculty of Engineering

University of Moratuwa
Sri Lanka

June 2024

DECLARATION

I declare that this is my own work and this Dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature *UOM Verified Signature*

Date: 2024-06-13

The supervisor should certify the Dissertation with the following declaration.

The above candidate has carried out research for the Masters in Computer Science Dissertation under my supervision. I confirm that the declaration made above by the student is true and correct.

Name of Supervisor: Prof. G.I.U.S. Perera

Signature of the Supervisor:

Date: 14.06.2024

DEDICATION

I dedicate this research to my pillar of success, my wife Ojithma, who has been my strength and supported me throughout this course, enabling me to complete it.

ACKNOWLEDGEMENT

I dedicate this research to my supervisor, Prof. Indika Perera, for the guidance and valuable feedback provided, which enabled me to transcend this project.

ABSTRACT

The performance of caching systems heavily relies on the effectiveness of cache replacement policies, which aim to optimize the usage of cache storage and improve cache hit rates. Traditional cache replacement policies such as Least Recently Used (LRU) and Least Frequently Used (LFU) have been widely used but have limitations in adapting to changes in the workload and minimizing the use of expensive storage. Recently, researchers have proposed machine learning-based approaches to cache replacement, aiming to learn from the access history of cache blocks and make accurate predictions about the usefulness of cache blocks. These approaches have shown promising results in improving the cache hit rate and reducing the overall storage cost compared to traditional policies. The proposed solutions include various machine learning algorithms such as Linear Regression (LR), K-Nearest Neighbor (KNN), and Deep Learning (DL).

The proposed solution implements a dataset-specific prediction model which first derives a dataset using an algorithm which uses future references with very high accuracies. Then the supervised machine learning model will be used to derive the patterns on the derived dataset which has used the future reference pattern in cache replacement decision making process. By this method, the cache replacement strategy will be much more resilient as all the traditional and state of the art models uses cache data from the past only. The evaluation and the future work provides much stronger impression on this method as there are several factors of this research which may significantly improve the results and will help the current cache replacement strategies tremendously.

The document also explains about the technical environments which used to evaluate models. The machine learning model training and the dataset generation require significant memory and CPU performance requirement. Hence, dedicated GPU is required in some scenarios which this document explains on how to manage.

Keywords: Cache replacement, machine learning, performance optimization, cache

TABLE OF CONTENTS

Declaration of the Candidate & Supervisor	i
Dedication	ii
Acknowledgement	iii
Abstract	iv
Table of Contents	v
List of Figures	viii
List of Tables	x
1 Introduction	1
1.1 The types of cache	1
1.2 Cache replacement	2
1.3 Least recently used policy	3
1.3.1 Implementation	4
1.4 Least frequently used policy	6
1.4.1 Implementation	6
1.4.2 Previous Proposed Method	10
1.5 Research Problem	11
1.6 Research Objectives	13
1.7 Thesis Structure	14
2 Literature Review	15
2.1 Analysis of the heuristic algorithms	15
2.1.1 Drawbacks of LRU Algorithm	15
2.1.2 Drawbacks of LFU Algorithm	17
2.2 Related Work	18
2.2.1 Machine learning based approaches	18
2.2.2 Heuristic based approaches	28
2.2.3 Application based studies	31
2.2.4 Gap Analysis	38

2.3	Datasets	41
3	Methodology	44
3.1	Model	44
3.1.1	Recurrent Neural Network Layer	44
3.1.2	Handling the large dimension space	48
3.1.3	Principal Component Analysis	48
3.1.4	Embedding Layer	49
3.1.5	LSTM as an Encoder	50
3.1.6	Model architecture	51
3.2	Dataset Creation	51
3.2.1	Reuse distance algorithm	52
3.2.2	Frozen cache algorithm	52
3.3	Proposed Cache Model	59
4	Implementation	62
4.1	Environments	62
4.1.1	Google Colab pricing	62
5	Results and Discussion	64
5.1	Evaluation Algorithm	64
5.2	Model Accuracy	65
5.2.1	Model overfitting	66
5.3	Proposed Cache Results	70
5.3.1	Reasons for low accuracy	71
5.3.2	Performance comparison	73
5.4	Experiments	74
5.4.1	The significance of the recurrent sequence order	74
5.5	Frozen Cache performance over Periodic data	77
5.6	Frequency Component Effect	79
5.7	Model performance with LRU	80
5.8	Limitations	81
5.8.1	Future reference drawbacks	81
5.8.2	Only relying on the future	83

5.8.3	Only using the cache state as the input	84
5.8.4	Ignoring time factor	85
5.8.5	Frozen cache algorithm shrinking the dataset length	88
5.8.6	Frequency-Recency score improvements	89
6	Conclusion	91
	References	93

LIST OF FIGURES

Figure	Description	Page
Figure 1.1	The data-flow between CPU and RAM	2
Figure 1.2	pseudo code for the LRU method	4
Figure 1.3	Python implementation for LRU algorithm	5
Figure 1.4	pseudo code for the LFU method	7
Figure 1.5	Python implementation for LFU algorithm	8
Figure 1.6	The sequence diagram of prediction cache update	11
Figure 1.7	The sequence diagram of prediction cache update	11
Figure 2.1	The drawback of LRU on periodic data	16
Figure 2.2	Cache Algorithm performances	40
Figure 2.3	Cache Algorithm performances	41
Figure 2.4	Dataset information	42
Figure 2.5	Top ten Reuse distances in web07	42
Figure 2.6	Top ten Reuse distances in web12	43
Figure 3.1	Long short term memory (LSTM) cell structure	46
Figure 3.2	The model architecture depicted by keras library	53
Figure 3.3	Reuse distance calculation algorithm	54
Figure 3.4	pseudo code for the frozen cache algorithm	55
Figure 3.5	Frozen cache put method when the cache is filled	56
Figure 3.6	pseudo code for the <code>get_removable_address</code> method	57
Figure 3.7	pseudo code for the <code>get_fr_score</code> method	57
Figure 3.8	The Hit rate comparison between LRU vs Frozen cache on Web07 dataset	58
Figure 3.9	The Hit rate comparison between LRU vs Frozen cache on Web12 dataset	58
Figure 3.10	pseudo code for the Proposed cache method	60
Figure 3.11	The overall architecture of Frozen cache and the Proposed cache	61
Figure 4.1	The Google Colab Runtime environments	63
Figure 5.1	Cache evaluation algorithm	64
Figure 5.2	Model validation accuracies per epoch on Web12 dataset	67
Figure 5.3	Model validation accuracies per epoch on Web12 dataset	68
Figure 5.4	Model training accuracies per epoch on Web12 dataset	69
Figure 5.5	Model training accuracies per epoch on Web12 dataset	70
Figure 5.6	Model accuracy per epoch on cache size 3000	71
Figure 5.7	Model loss per epoch	72
Figure 5.8	Cache performance over trace time steps	74
Figure 5.9	Cache performance over trace time steps	75

Figure 5.10	Expected Hit rate from Proposed cache	76
Figure 5.11	Expected vs Actual performance comparison	77
Figure 5.12	Expected vs Actual performance comparison	78
Figure 5.13	Cache performance comparison for Web07 with other studies	79
Figure 5.14	Cache performance comparison for Web12 with other studies	80
Figure 5.15	Hit rate based on cache size when the input is non-reversed	81
Figure 5.16	Comparison of Performance with Reversed and non-reversed sequences	82
Figure 5.17	Hit rates on Period being one address larger than the cache size	83
Figure 5.18	Hit rates on Period being hundred address larger than the cache size	84
Figure 5.19	Hit rates on Period being hundred address less than the cache size	85
Figure 5.20	Model accuracy without frequency component	86
Figure 5.21	Hit rates on without frequency component	86
Figure 5.22	Hit rates on without frequency component	87
Figure 5.23	Standard LRU comparison with ML based LRU	87
Figure 5.24	Input concatenation for Machine learning model	88

LIST OF TABLES

Table	Description	Page
Table 2.1	Reference information for the datasets	43
Table 3.1	Model parameter counts	52
Table 5.1	Frozen cache dataset length for cache size for the web12 dataset	89