Storage Cache Optimization using Machine Learning Techniques

Maryna Baida[2] Hiroki Ohtsuji[1] Erika Hayashi[1] Eiji Yoshida[1]

Abstract: This paper describes a method to optimize storage cache mechanisms using machine learning techniques. It is difficult for conventional algorithms to understand complicated parallel IO patterns. Recent research papers do not highlight the possibility for the use of machine learning techniques in the storage cache optimization very often. Our approach suggests applying LSTM networks for this purpose. We also apply different data manipulations and preprocessing techniques to understand how it is possible to use the result of IO classification and improve IO performance. This paper focuses on the method to detect changes of IO patterns to change the cache configurations optimally. Further investigation of the application of this paper results in the design of an effective caching algorithm for storage systems.

Keywords: Storage Systems, Machine learning, Cache parameter optimization

1. Introduction

In the previous years there have been many different articles published regarding the use of LSTM networks in the different areas of science [1-9]. But there were very few attempts to use this technique for the cache related prediction. 00 In this work we show that this technique is not only possible to use, but also brings fruitful results. Cache simulation takes quite a long time and is not fast enough to be used in the runtime environment. While determining the best cache size ratio policy needs few such simulations with different configurations to be run together and evaluated in parallel. The method we propose creates the basis for the prediction of the change of the optimal cache policy. With this method system takes much less time to produce the decision regarding the best cache size percentage during the runtime operation. Given the circumstances, availability of the storage resources may be changed according to the optimal solution calculated with the help of LSTM network running in parallel to the system and determining the best cache policy accordingly. LSTM network usage allows the cache simulation to be speed up with minor accuracy change in the cache hit ratio. Especially on the large volumes of data.

We thus focus on a generic problem to predict the points at which the optimal cache policy has changed and include only temporal, spatial and volume metrics of the traces for the prediction.

In this paper, we address the following questions:

What is the right cache size ratio for the optimal performance of the system at a each point of time?

Can the machine learning and LSTM network in particular be faster in the decision making of determining the change of the optimal parameters in the runtime environment?

We examine arc cache algorithm and four different cache size ratios with few different kinds of IO traces.

For high load systems, we can apply the proposed approach to intelligent cache size management.

Fig. 1 shows the overall flow of our approach. Our goal is to determine optimal cache size strategy and to propose a model that can later be used to predict the change of the optimal cache size ratio in the system faster than it is done by the simulation. In this paper, we run the simulation on IO traces with different settings, while measuring the simulation results at specific points of the trace. Afterwards we split the IO trace into chunks determined by the points of measurement and use these chunks in the training of LSTM model to determine the changing point.

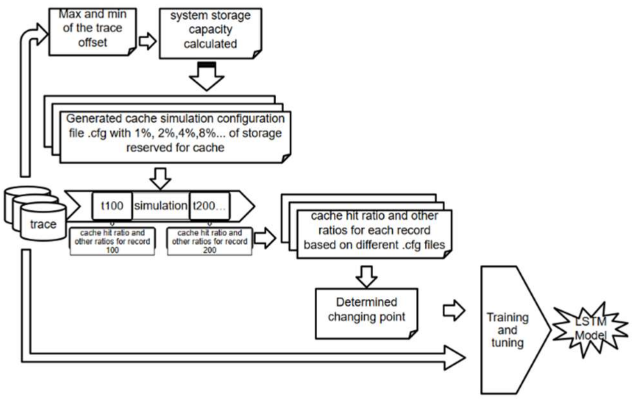
Max and min of the trace offset
system storage capacity calculated
Generated cache simulation configuration file .cfg with 1%, 2%,4%,8%... of storage reserved for cache
trace
t100
simulation
t200...
cache hit ratio and other ratios for record 100
cache hit ratio and other ratios for record 200
cache hit ratio and other ratios for each record based on different .cfg files
Determined changing point
Training and tuning
LSTM Model

Fig. 1 Structure of the workflow

We can include this model in the system and use it in runtime for monitoring of the optimal use of cache resources.

2. Related Work

Many recent papers describe some different approaches for caching [8-11]. Few of them focus on the caching strategies in the network infrastructures [13-16]. Others tried to use machine learning techniques in prefetchers [20-21]. And few proposed approaches for use of LSTM networks in caching [12][14]. Also there are some papers in the IO workflow analysis.

In this paper we analyse the workflow of a MSR Cambridge trace, and its hm0 (Hardware Monitoring) server in particular in terms of optimal caching ratio policy change. We have considered other approaches of workflow analysis, though.

For example, the recent work on IO workflow analysis is focusing on the pattern identification of different traces. [17] In this work, first step is to calculate a set of 20 descriptive features of each of the trace parts of 8000 offset entries each. The resulting "feature" dataset is then evaluated by means of k-means algorithm and

1 Fujitsu Laboratories Ltd., Kawasaki, Kanagawa 211-8588, Japan
2 Budapest University of Technology and Economics, Budapest, 1111, Hungary

modified version of backward elimination for determining the most descriptive features of the dataset. The dataset of features is repeatedly clustered while determining the silhouette value of each feature and so the contribution of this particular feature to the clustering process. The feature which contributes least is then eliminated from the dataset to repeat the process again until no other features left. This kind of backward elimination algorithm is repeated with different number of clusters. In this way, the set of features and number of clusters with best silhouette value is determined. Then the resulting feature dataset is evaluated by means of 9 other clustering algorithms to determine the best algorithm suitable for the task. The resulting subsets of features in each cluster gives an idea about the structure of IO patterns. After that the tree structure is constructed with the feature value as the determining point for the IO pattern definition. After this the tree is evaluated on the dataset to determine the speed of the tree execution in the IO pattern detection. And it gives much faster result than the Naïve Bayes, K-nearest neighbours, SVM or Logistic Regression, when used for the same purpose.

Our approach also analyses IO trace pattern, but in terms of cache simulation, which in fact may be considered as a kind of feature of the trace. And base on this “feature” of the trace our method determines the change of the optimal point of the cache size ratio.

Regarding the works in the field of LSTM applications for caching process, DEEPCACHE [12] shows the ability of LSTM based models to predict the popularity of content objects. They evaluated it using two synthetic datasets under multiple settings out of which one tries to emulate realistic workloads. This work is using the seq2seq modelling to predict the future object popularity. They implement it with the help of LSTM Encoder-Decoder model. The way how it works is that the DEEPCACHE algorithm first predicts the objects with the help of LSTM model, and then merges the fake requests for those objects with existing cache replacement algorithms such as LRU or k-LRU. [12]

In our case we do not have the need for the Encoder-Decoder model as the possible outputs are only 0 and 1, optimal cache policy changed and optimal cache policy stayed the same, respectively.

Another paper which considered LSTM applicable for making caching more effective [14] is considering the cacheability of the data as the prediction target in their research. They consider this under the conditions of using WHD (windows hit density) a modified version of LHD (least hit density) instead of LRU as their eviction policy. WHD uses total access times in a window to replace hit probability based on probability model and uses the length of window to replace the expected time in cache. The output of the LSTM layer is designed to be passed into a hidden layer with softmax function, which finally outputs the cacheability of each data in the [0, 1] interval. And the model is trained by minimizing the cross-entropy error between the output of the whole network and the true value. The LSTM then performs classification, and the K highest cacheability items are chosen to be cacheable.0

In our case we use sigmoid function and the binary_crossentropy for training and prediction if the current cache size ratio policy is optimal or not.

Few other studies discuss the application of LSTM networks in the hardware prefetchers. These papers focus on the determining the PC (program counter) of each entry in the trace together with the offset. In our case the entry to LSTM network is the trace of 100 records in one row. Each record has the Timestamp, DiskNumber, Type, Offset, Size and ResponseTime properties in its raw state to determine the changing point of the optimal cache policy change.

3. Solution Design

In this paper, we applied LSTM network to extract the feature from IO traces. The purpose of our method is to detect the point where the optimal cache policy and configuration changes. The target performance factor is cache hit ratio. For the data preparation, we used a cache simulator to calculate the cache hit ratio of the IO trace files with multiple cache configuration. In the next step, gathering the set of IO trace and the cache simulation result to detect the changing point. If the cache hit ratio decreases rapidly with a certain configuration while the hit ratio with the other configuration increase at the certain point, we can define the point as changing point. The proposed design applies this detecting mechanism to multiple IO traces to generate multiple training datasets. The training model takes IO traces as the input and the binary value (changing point or not) as the output. Using the prediction model, the method can detect the changing point in the given IO trace file. During the normal operation of storage systems, when the prediction model finds a changing point, the system can start to find the process to select a different cache configuration. Our prediction model is faster than running the cache simulation so that we can accelerate the process to daily routine of the performance tuning of storage systems.

3.1 Using LSTM to detect changing points in IO traces

As there are only two options in the inference process: changing point or not, this problem can be treated as a binary classification which is simpler than the multiclassification in low-level cache system. In this paper we analyze some statistical properties of accessing data including response time, read-write ratio and the size of data. The input of LSTM prediction model we proposed are based on these statistical properties.

3.2 Stateful vs stateless LSTM

In our research we applied both stateful and stateless LSTM versions to determine which one would give better results on our dataset.

The main difference between stateful and stateless LSTM is that stateless LSTM puts zeros to all inputs after one batch is processed. Stateful LSTM in its turn saves the output of the states of the previous batch to be fed into the next batch. So generally the stateful LSTM is the LSTM which tries to take into account whole sequence of the training dataset, while stateless one only takes into account the sequence inside one batch.

3.3 Software

We use the Keras on TensorFlow for the LSTM model training and prediction, as well as Numpy, Sklearn, Pandas libraries for the dataset preprocessing.

For the cache simulation purposes we have modified the following open-source project. [22] This simulator is capable of simulating the cache by the arc strategy. The output of cache simulation contains following fields: Level of the cache, Total References, Total Reads, Total Writes, Page Read Hit, Page Read Miss, Block Read Hit, Block Read Miss, Page Write Hit, Page Write Miss, Block Evict, Cold2Cold, Cold2Hot, DirtyPage, SeqEviction, LessSeqEviction, Total Seq Evicted Dirty Pages, Total Non Seq Evicted Dirty Pages, Total Evicted Clean Pages, Real Total Evicted pages, Page Read Cache Hit Ratio, Page Write Cache Hit Ratio, Total Cache Hit Ratio. Our modification was to make the simulator output these statistics after some specified number of lines pointed in the trace, and not only after all trace was processed.

4. Details of implementation

We use MSR Cambridge dataset for analysis. The dataset contains following properties of each trace entry: Timestamp, Hostname, DiskNumber, Type, Offset, Size, ResponseTime.

For analysis we do not normalize the data, do not apply any clusterization technique or any other preprocessing, except excluding the Hostname from the analysis and converting the “Write” or “Read” Type column into 0 or 1 respectively to prevent the need for language processing on this matter.

According to the obtained results given that the LSTM-based model is to be implemented as part of an IO system eventually, we should not ignore the computational overhead of splitting and pre-processing the parts of the IO trace. Second issue is that LSTM network itself does not accept arbitrary number of columns for processing. To the best of our knowledge it breaks at approximately 3000 columns representing a part of the trace. To address this problem, we split the trace into small chunks of the size 100 which we transpose and feed into the network. With this method, the network still gives quite good results while the number of the measurements is still sufficiently low.

The cache percent size is determined from the minimum and maximum number of the offset, which is considered the overall size of the volume. From that we calculate the 1, 2, 4 and 8 percent size of the cache, and generate configuration file for the simulator with the obtained values as the size of the cache. Using each of these configurations we run cache simulations. After that we determine the maximum local cache hit ratio, at each point of measurement. The point of measurement can be variable, but we stopped at the number 100, due to difficulties in processing bigger chunks of data using LSTM network.

As the Cache Hit Ratio in the output of the simulator is calculated based on the whole simulation done before, to calculate the value of Local Cache Hit Ratio we use the outputs of the simulator in the neighbouring steps of measurement (in our case 100 lines of the trace).

The formula for the definition of the local cache hit ratio is as follows:

$$\mathrm{LCHR} = \frac{((\mathrm{PWH1} - \mathrm{PWH0}) + (\mathrm{PRH1} - \mathrm{PRH0}))}{((\mathrm{TW1} - \mathrm{TW0}) + (\mathrm{TR1} - \mathrm{TR0}))}, \text{where}$$

Table 1 LCHR explanation of symbols in formula

LCHR	local cache hit ratio (at the point of measurement)
PWH	page write hit
PRH	page read hit
TR	total reads
TW	total writes
…1	of current measurement point
…0	of previous measurement point

So, if 1% cache size ratio policy gives best cache hit ratio in the first 100 records, but in the next 100 records the best cache hit ratio will be obtained by applying the cache size ratio policy of 2%, then the latter row of 100 records will be marked as changing point. And so on. Like it is shown in Table 2.

Table 2 Cache size ratio optimal policy changing point definition

trace part	local cache hit ratio with 1% cache	local cache hit ratio with 2% cache	local cache hit ratio with 4% cache	local cache hit ratio with 8% cache	optimal parameter	optimal cache size ratio policy changed
100	12	18	24	40	8	0
200	12	18	24	40	8	1
300	12	12	12	12	1	0
400	12	15	16	16	4	0
500	3	14	14	14	2	0
600	3	14	14	14	2	1

4.1 Details of LSTM network implementation.

We evaluated different types of LSTM network with different parameters in use in our approach. First, we tried to configure the network with the categories in mind, and so the classification issue was the initial point. But, in the end the binary crossentropy and binary accuracy were chosen as the training parameters. As for the batch size we keep it at the rate of 1, as it gives better accuracy and also is easier to find a multiplicator for the higher degree in case of stateful LSTM network. For now, stateless and stateful LSTM networks give comparable results with maximum accuracy value of 61,3%.

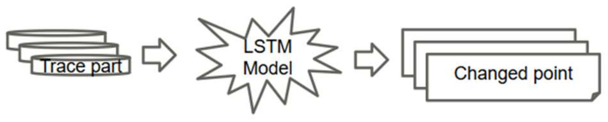

Fig. 2 Prediction process

As expected, the proposed method requires the smallest amount of time to complete the task of determining if the system still works with the optimum cache strategy in mind. The advantage in running time obviously comes from the features of LSTM network.

When executed in the real system, the proposed approach would make the performance estimations of the cache policy without actually running the cache simulation itself.

Just for notice, the cache simulation program takes hours to run in the best case scenario, taking a lot of system resources, with the trained LSTM network optimal cache size ratio changing point is calculated in few minutes maximum.

The network structure is still in the process of tuning so we suppose that such accuracy is not the limit of this approach.

5. Evaluation

This section describes the result of the experiment with our proposed model. The input dataset includes the 600 columns (100 results in one row, each having 6 fields: Timestamp, DiskNumber, Type, Offset, Size, ResponseTime.) and the label defining if cache size policy is optimal at the time of processing these 100 trace lines or not.

For the configurations, the following settings were used: one layer of LSTM nodes with the number of hidden units of 100, and the Dense layer of 1 node (as there are only 0s and 1s in label). The LSTM layer has the tanh activation function, The Dense layer has sigmoid activation function. Loss function for the model compilation is binary_crossentropy. We used Adam optimizer for network training. Batch size is determined by the type of the model in question. If the model is stateful then the batch_input_shape is a tuple (1, size_of_one_row,1). If the model is stateless, then the batch_size is 1. The split to test and train data is performed using the train_text_split function of the sklearn.model.selection module. Test size is 0.2.

For the stateful model we used ResetStatesCallback to reset the states after each sequence of 600 fields processed (number of columns) and send it to next batch.0

We have run the model on two datasets. For now results are as it is shown on Fig. 3. The hm0, 1 part contains 2000 rows 600 fields each. The training accuracy for the stateful and stateless models are approximately the same level - 0.808. As a baseline we took the mean value to be sure that the model does not always output the same number. Mean for this case is 0.7795. For the case with the bigger chunk of data (hm0 39934 rows, 600 columns) the accuracy is approximately 0.613, while having mean of 0.6177. We suppose that the model training could go better if the size of the LSTM layer was bigger compared to the number of the fields in one row. But trials on the network with a higher number of hidden nodes, for example, 876, have not succeeded for the stateless model on the dataset of hm0. Notice:

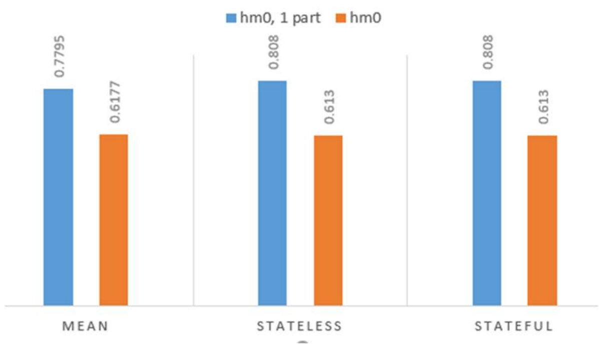

Fig. 3 Experimental results

stateless model needs less resources to train, so if Python kernel fails at stateless model than running the model with stateful settings is pointless. Possible solutions for this trouble would be to output the results of a simulation more often than each 100 line of the trace, and so to split the data and train the LSTM network on the dataset with fewer columns.

Table 3 Software and hardware specifications

Python	3.7.7
Tensorflow	2.1.0
tensorflow-gpu	2.1.0
cuda toolkit	10.1.243_h74a9793_0
CUDA version	10.2

5.1 Discussion

As for the effectiveness of this approach, the current structure of the network and the data chunk size may be changed according to the needs of the system. There is a possibility that adding different datasets into the network would diversify the applicability of this method. It is necessary to train the model with various types of training data sets and we are currently working on extra training cases.

Possible improvements would include trials to build the network with different cell structure like GRU or RUM [23] and applying the error and outlier detection technologies for the rarely happening changing points in the cache size ratio policy. [25-26]

6. Conclusion and Future Work

In this paper we have proposed the optimal caching size ratio change problem. We showed that the changing point of the optimal cache size ratio of the system can be successfully determined by means of LSTM network algorithm. The result shows that the proposed method takes at least 2 times less time than the cache simulation itself. The future work is applying the proposed method to real storage systems to know the actual advantages on real workloads.

References

[1] Dmytro Zhelezniakov, Viktor Zaytsev, Olga Radyvonenko,"Acceleration of Online Recognition of 2D Sequences Using Deep Bidirectional LSTM and Dynamic Programming", Springer LNCS, Vol. 11507, 2019

[2] Yaroslav Shkarupa, Robert Mencis, Matthia Sabatelli., “Offline Handwriting Recognition Using LSTM Recurrent Neural Networks”, THE 28TH BENELUX CONFERENCE ON ARTIFICIAL INTELLIGENCE November 10-11, 2016

[3] E. Fedorov, V. Patrushev, O. Patrusheva, “METHOD FOR THE PROMOTION OF INTERNET ACCOUNT OF THE SHOP ON THE BASIS OF A COMPLETE LONG-TERM SHORT-MEMORY MEMORY, TRAINING BY A GENETIC ALGORITHM”, Наукові праці ДонНТУ №2 (25), 2017 Серія “Інформатика, кібернетика та обчислювальна техніка”, p 118-125, 2017

[4] Le, Xuan-Hien; Ho, Hung V.; Lee, Giha; Jung, Sungho. 2019. "Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting." Water 11, no. 7: 1387.

[5] Mohammed Salah Al-Radhi, Tamás Gábor Csapó, Géza Németh "Deep Recurrent Neural Networks in Speech Synthesis Using a Continuous Vocoder" International Conference on Speech and Computer SPECOM 2017: Speech and Computer pp 282-291, 2017

[6] “Forecasting stock prices with long-short term memory neural network based on attention mechanism”. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0227222, (accessed 2020-06-25).

[7] “Traffic congestion anomaly detection and prediction using deep learning”. https://arxiv.org/abs/2006.13215, (accessed 2020-06-25).

[8] “Time Series Analysis and Forecasting of COVID-19 Cases Using LSTM and ARIMA Models”. https://arxiv.org/abs/2006.13852, (accessed 2020-06-25).

[9] “Implementing A Neural Cache LSTM”. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2760222.pdf, (accessed 2020-06-25).

[10] “Model and Machine Learning based Caching and Routing Algorithms for Cache-enabled Networks”. https://arxiv.org/abs/2004.06787, (accessed 2020-06-25).

[11] “Domain-Specialized Cache Management for Graph Analytics”. https://arxiv.org/abs/2001.09783, (accessed 2020-06-25).

[12] Haibo Wu, Jun Li, Jiang Zhi, Yongmao Ren, Lingling Li, "Edge-oriented Collaborative Caching in Information-Centric Networking", IEEE Symposium on Computers and Communications, 2019.

[13] Ying Liu, Ting Zhi, Haidong Xi, Wei Quan, Hongke Zhang, "A Novel Cache Replacement Scheme against Cache Pollution Attack in Content-Centric Networks", 2019 IEEE/CIC International Conference on Communications in China (ICCC), 2019

[14] Arvind Narayanan, Saurabh Verma, Eman Ramadan, Pariya Babaie, Zhi-Li Zhang "DeepCache: A Deep Learning Based Framework For Content Caching", NetAI'18: Proceedings of the 2018 Workshop on Network Meets AI & MLAugust 2018 Pages 48–53, 2018

[15] Seok Won Kang, Kyi Thar, Choong Seon Hong, "Unmanned Aerial Vehicle Allocation and Deep Learning based Content Caching in Wireless Network", 2020 International Conference on Information Networking (ICOIN), 2020

[16] Jiawei Fei, Yang Shi, Mei Wen, Chunyuan Zhang, "SACC: Configuring Application-Level Cache Intelligently for In-Memory Database Based on Long Short-Term Memory", 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2019

[17] Kuo Chun Tsai, Li Wang, Zhu Han, "Caching for Mobile Social Networks with Deep Learning: Twitter Analysis for 2016 U.S. Election", IEEE Transactions on Network Science and Engineering (Volume: 7 , Issue: 1 , Jan.-March 1 2020), 2020

[18] Yang You, Yuxiong He, Samyam Rajbhandari, Wenhan Wang, Cho-Jui Hsieh, Kurt Keutzer, James Demmel, "Fast LSTM Inference by Dynamic Decomposition on Cloud Systems", 2019 IEEE International Conference on Data Mining (ICDM), 2019

[19] Bumjoon Seo, Sooyong Kang, Jongmoo Choi, Jaehyuk Cha, Youjip Won, Sungroh Yoon, "IO Workload Characterization Revisited: A Data-Mining Approach", IEEE Transactions on Computers (Volume: 63 , Issue: 12 , Dec. 2014) , 2014

[20] "Long short-term memory". https://en.wikipedia.org/wiki/Long_short-term_memory, (accessed 2020-06-29).

[21] "Understanding LSTM Networks". https://colah.github.io/posts/2015-08-Understanding-LSTMs, (accessed 2020-06-29).

[22] Yuan Zeng, Xiaochen Guo, "Long short term memory based hardware prefetcher: a case study", MEMSYS '17: Proceedings of the International Symposium on Memory SystemsOctober 2017 Pages 305–311, 2017

[23] Milad Hashemi, Kevin Swersky, Jamie Smith, Grant Ayers, Heiner Litz, Jichuan Chang, Christos Kozyrakis, Parthasarathy Ranganathan, "Learning Memory Access Patterns", Proceedings of the 35th International Conference on Machine Learning, PMLR 80:1919-1928, 2018

[24] "sim-ideal". https://github.com/arh/sim-ideal, (accessed 2020-07-01).

[25] Rumen Dangovski, Li Jing, Preslav Nakov, Mićo Tatalović and Marin Soljačić, "Rotational Unit of Memory: A Novel Representation Unit for RNNs with Scalable Applications", 2019 Association for Computational Linguistics,   The MIT Press Journals, 2019

[26] “Stateful LSTM in Keras”. http://philipperemy.github.io/keras-stateful-lstm/, (accessed 2020-07-03).

[27] “Automatic model generation technology for error detection” https://pr.fujitsu.com/jp/news/2020/03/16.html, (accessed 2020-07-03).

[28] Oleksandr I. Provotar, Yaroslav M. Linder, Maksym M. Veres, "Unsupervised Anomaly Detection in Time Series Using LSTM-Based Autoencoders", 2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT), 2019

[29] Fahimeh Farahnakian, Jukka Heikkonen, "A deep auto-encoder based approach for intrusion detection system", 2018 20th International Conference on Advanced Communication Technology (ICACT), 2018