Work-In-Progress: A Deep Learning Strategy for I/O Scheduling in Storage Systems

Ashkan Farhangi*, Jiang Bian*, Jun Wang, Zhishan Guo
Department of Electrical and Computer Engineering University of Central Florida , United States
{ash, bjbj11111}@knights.ucf.edu, {jun.wang, zsguo}@ucf.edu

***Abstract*—Under the big data era, there is a crucial need to improve the performance of storage systems for data-intensive applications. Data-intensive applications tend to behave in a predictable manner, which can be exploited for improving the performance of the storage system. At the storage level, we propose a deep recurrent neural network that learns the patterns of I/O requests and predicts the upcoming ones, such that memory contents can be pre-loaded at the right time to prevent cache/memory misses. Preliminary experimental results, on two real-world I/O logs of storage systems (from financial and web search), are reported—they partially demonstrate the effectiveness of the proposed method.**

I. Introduction

In the last two decades, we have observed a significant improvement in processor's executing capabilities due to the production of smaller transistors and the move towards multi-core processor platforms [1], [2]. However, such increase does not lead to the same amount of increase in computational performances [3]. This is partially due to the I/O bottleneck; i.e., the processor remains idle when the storage read and write commands are being completed [4]. Not only the local storage systems are heavily influenced by the I/O bottleneck, but cloud storage systems also suffer from the ever-increasing workload of data-intensive and parallel applications. To address the I/O bottleneck, much research (e.g., [3], [4]) has been done to leverage prefetching in bridging the speed gap between storage, memory, and the processor. Particularly, Data-intensive applications tend to behave in a predictable manner and the recent prevalent processor prefetch [3] intends to use deep learning method to learn patterns of specific workloads that are characterized by stable and repeatable patterns.

Most of the hardware storage prefetching is optimized for relatively simple I/O patterns. For some complicated I/O patterns, especially in the cloud storage systems, the large number of mixed I/O requests lead to difficulties in identifying and analyzing the patterns caused by a mixture of different workloads. Furthermore, limited attention has been given towards the temporal requirements of I/O prediction, which becomes an urgent topic due to the explosive growth of data stream in the era of real-time Internet of Things (IoT). In this case, understanding the underlying structure of I/O request correlations can aid in improving the performance of prefetchers. With the latest advancement in the deep learning models such as sequence to sequence learning (Seq2Seq) [5], we can now achieve higher accuracy in the prediction of I/O addresses [6]. As most storage problems are depended on caches and prefetchers, we intend to build a time-aware multi-task I/O predictor that delivers the upcoming I/O request to sequence prefetcher [4], [7].

However, there are several technical challenges in developing such a multi-task I/O predictor in storage systems. First, the storage access address is very sparse: depending on storage size, it can be as high as 10 million unique string of numbers. This extreme large data scale can lead to performance degradation in prediction [3] under traditional Seq2Seq models. Second, most Seq2Seq models are proposed for offline prediction, and thus timing performance is not a major consideration in their design [4]. Since sequential prefetching is tightly correlated with meeting the I/O requests (deadlines), we have to take the timing restrictions into account. As a result, the complexity of the model needs to be severely constrained, so that the delivery of the upcoming I/O request can be finished ahead of time.

To address the aforementioned problems, we propose a Long Short-Term Memory [8] (LSTM) network to retrieve I/O requests. Even though LSTMs can be used as a baseline model for sequential data, the gates are unable to adapt to the multi-output manner directly. To address the need for a multi-step ahead prediction, a specialized Multi-Input Multi-Output strategy [5] is adopted, where LSTM cells are stacked to improve the multiple-step ahead prediction. Specifically, the proposed multi-task I/O predictor follows two main phases: offline training and online simulation. In the offline training phase, the predictor starts learning the patterns that are recurring in the I/O traces; while in the online simulation phase, the predictor loads the upcoming I/O addresses in a timely manner. Lastly, the on-going study has the potential to provide new insights into reducing the load of Seq2Seq learning by unsupervised learning and timing analysis.

The main contributions are:

- We design a novel Seq2Seq learning-based prefetcher for real-time I/O prediction in storage systems.
- We intend to address the timing constraints of the prefetcher by proposing a multi-task I/O predictor.
- Preliminary experimental study is conducted based on real-world I/O request logs, which demonstrate that the strategy has the potential to handle complicated I/O patterns while meeting timing constraints.

2576-3172/19/$31.00 ©2019 IEEE
DOI 10.1109/RTSS46320.2019.00066

The rest of the paper is organized as follows. Section II presents the system model and strategies to improve the efficiency of the approach. Section III describes the preliminary results in model training and compares our model with existing related work. Section IV concludes the current efforts and highlight some on-going work.

II. I/O Predictor System

In this section, we introduce the detailed structure and procedures of the I/O predictor. In the meantime, we also provide some system backgrounds.

A. System Structure

We start with an overview of the overall predictor system structure. As shown in Figure 1, the system contains two major phases: **offline training** and **online simulation**. During offline training, the multi-task I/O predictor will learn the essential patterns that are recurring in the I/O requests. While during online simulation, the prefetcher loads the memory/cache with upcoming predicted I/O requests for preventing potential memory/cache misses.

Specifically, for **offline training**, we first employ data and time analysis to extract the most recurring essential patterns of I/O requests. Second, we use the K-means clustering method to assign the I/O sequences to differentiate various sectors in storage. Third, we employ Seq2Seq learning for each sector of storage. For **online simulation**, we develop a multi-task real-time I/O prefetcher to load the predicted memory pages by a specific time point. The following two subsections discuss the details of the two phases.

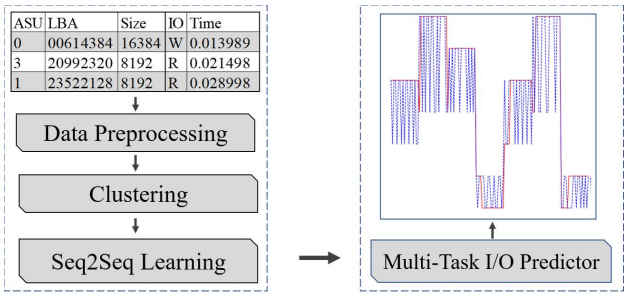
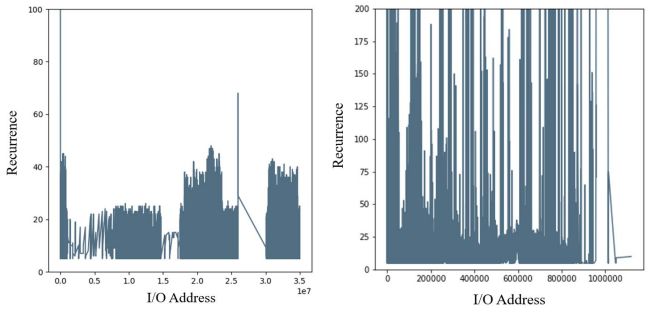


Fig. 1: The overall flow/structure of the I/O prediction and prefetching system. Where the input is sequences/log of I/O request, and the output (ideally) is a well prefetched memory/cache pipeline with high hit ratio (in the subfigure, x-axis is the timeline, while y-axis represents the addresses of I/O requests)—blue dashed line is actual I/O requests while red lines are the actual prefetch conducted (by the predictor) based on output of the predictor.

B. Offline Training

Data Preprocessing. To develop a general predictor that can handle complex I/O requests, in this subsection, we use two data sets to motivate and demonstrate the assessment. First, we analyze a type of storage system trace known as online transaction processing (OLTP) that is widely used in financial institutions. OLTP traces are commonly known as having a large memory footprint, where index searching becomes challenging and requires distinct attention. Moreover, OLTP systems are used for financial transactions, order entry, and account services [9]. Second, we examined I/O traces of a search engine storage system that contains a greater number of random I/O accesses [10]. For both I/O traces, the storage access address is very sparse, and cover more than 300,000 unique addresses. Note that any memory miss will cause a huge delay in fetching from the hard disc storage. Therefore, predicting the exact I/O address for preloading brings great benefits in terms of resource efficiency [4].

We start the preprocessing by neglecting the low-frequency I/O addresses from the training data and focus our attention on the most recurring I/O addresses. As shown in Figure 2, we display the I/O addresses with more than 5-time-recurring of a particular OLTP storage trace (Financial 1) and a web search storage trace (WebSearch 2) [11]. These I/O addresses will be used as a vocabulary for the prediction model.



(a) WebSearch 2 (b) Financial 1

Fig. 2: Recurrence numbers (along the y-axis) of I/O addresses for each trace dataset.

As demonstrated in Figure 1, the system model starts with a stream of I/O requests. At the storage level, our primary concern is to deliver the most useful I/O commands to cache when there is a significant change in storage accesses. Therefore, the importance of time analysis appears in the preprocessing stage, where it could act as an indication of a change in I/O sequences. which can be seen as a shift toward slower, larger storage or that another application has started.

As shown in Figure 3, by choosing an appropriate time window, we can group the I/O requests that are close to each other as a single sequence [12]. The sequencing method will reduce the size and load of the system model and trains the system to model for the sequences of I/O addresses that will be missed by the on-chip cache.

Clustering. Unfortunately, I/O accesses often come from many users/applications running which makes the pattern prediction complicated. We hope to separate each sector of

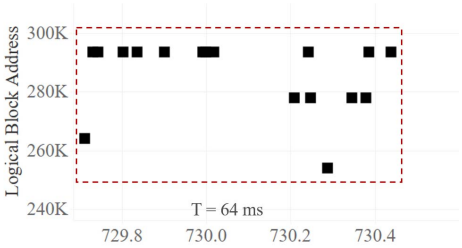


Fig. 3: The use of time window for sequencing the I/O requests.

storage so that Seq2Seq learning is applied more specifically and aid us in reducing the vocabulary size, and increasing the prediction rate [3]. In our study, we use the K-means method to cluster the sequences of I/O addresses that are spatially close to each other. Clustering before learning has many other benefits. For example, in a storage system that contains more random access (such as WebSearch), the complexity of the system can be reduced, such that learning of its spatial patterns can be very effective. Also, this enables parallel processing during the learning/prediction phase, as discussed below.

Transferred Seq2Seq Learning. After clustering each sector of storage, we develop a transferred Seq2Seq model that learns the temporal pattern for each sector. In most NLP problems, the time of appearance of each word in a sentence does not hold any value. However, the exact time of appearance for I/O addresses holds a great value that cannot be dismissed (temporal information is critical in I/O scheduling). As we chunk the I/O stream into smaller sequences, the Seq2Seq model learns to adapt to the changes that happen with time for each I/O request. This is similar to natural language processing (NLP), where words get extracted as features for embedded information in the training phase [12]. Here we employ our time analysis method to build sequences of data that are important and can result in adequate accuracy. Based on the statistical language model in NLP, similar I/O correlation can be exploited and transferred for prediction. A neural network architecture (shown in Figure 4) with a stacked LSTM Encoder-Decoder is proposed. It consists of two LSTM networks that act as an encoder-and-decoder pair.

C. Online Simulation

As discussed in the offline training part, Seq2Seq learning conducts training on the existing temporal and spatial patterns of I/O requests. During the online simulation, we intend to use the trained Seq2Seq model for predicting the upcoming I/O requests in the near (but not immediate) future in a real-time manner. This is to leave enough time to preload the predicted storage pages into the cache/memory and reduce miss ratios.

Specifically, a **Multi-Task I/O Predictor** is deployed to choose the appropriate cluster, compute the prediction, and deliver the upcoming I/O requests to the cache/memory While the key challenge in the multi-task I/O predictor is the notion of **timeliness**. *The prediction must be further enough in time such that enough time remains for predicting and prefetching.*

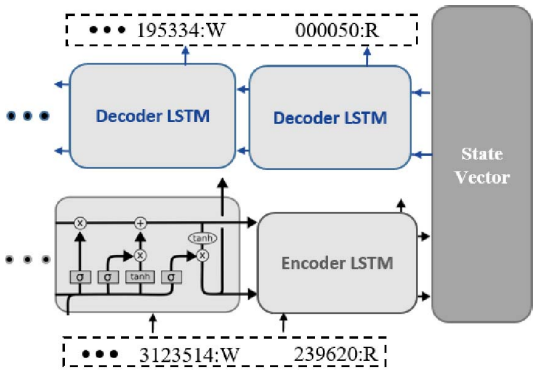


Fig. 4: I/O stream will be encoded and decoded in a sequence that can be used for Seq2Seq learning.

In other words, if the prefetching of correct sequences is applied too early, there is a chance that the correct data gets evicted from the processor soon and thus causes a miss later on when actual I/O request arrives [3]; while the prefetching cannot be too late for obvious reasons. In the two examples considered in this paper, the prediction is made around a few hundred steps ahead of the current memory access. This will give us the ability to move the new predicted I/O stream to the processor at the correct time, so the predicted I/O accesses could be utilized [13].

III. Experiments and Discussion

In this section, we evaluate our I/O predictor system on two real-world data sets: Financial 1 and WebSearch 2 [11]. For comparison, we include two popular algorithms aforementioned as the baselines, which are Block2Vec [13] and C-Miner [12] in the experiment. Then, we report the prediction accuracy and the computational speed of our proposed model. Finally, we discuss the current results and future improvements that can be further conducted.

Implementation & Current Results. As shown in figure 4, the offline training starts by feeding the sequence of I/O streams to the encoder, which maps a variable-length of sequences to a fixed-length vector. Then, the decoder maps the vector representation back to a variable-length target sequence. These two networks are trained jointly to maximize the conditional probability of the target sequence for a given sequence [5]. To increase the reliability of the prediction, we used a random batch generation in both training and evaluation. Also, we conducted an unsupervised clustering on the I/O accesses. As shown in Figure 5, clustering is helpful in generalizing the spatial patterns of I/O accesses, as the Seq2Seq model will learn the I/O addresses a specific section of storage. Moreover, for the Financial 1 data set, the clustering method reduced the computational speed by almost 24% and increased the accuracy by 14%.

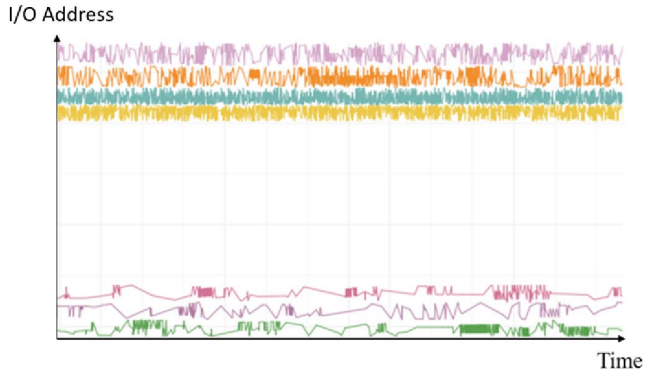


Fig. 5: The use of clustering to specify a section of storage for Seq2Seq learning.

Another way to increase the accuracy of our prediction is to improve the optimization of our loss function. During the training phase, the network will improve its precision by adjusting its weighted cross-entropy loss for a sequence of I/O addresses that were prepossessed. The training and testing were done on 250,000 and 25,000 I/O request respectively. For each sequence, 130 I/O commands are included that would satisfy our multi-step ahead prediction. As shown in Table 1, the evaluation of our model shows that the improved optimization yield great accuracy compared with the baseline algorithms. As for the Financial dataset in 130, I/O commands, on average, nearly 100 of these requests get predicted accurately. Lastly, the accuracy of the proposed model for WebSearch 2 is lower than Financial 1 due to having more random accesses.

	Financial 1	Websearch 1
C-Miner [13]	48%	-
Block2Vec [12]	72%	-
Multi-Task I/O Predictor	78%	23%
Gradient Loss	0.79	1.54
Batch Size	256	512
Number of Clusters	3	7
Window Size	64ms	32ms

Table 1: Comparison of the multi-task I/O predictor's performance with previous studies and the training parameters.

IV. Conclusion & Future Work

In this study, we showed that the storage access patterns could be learned and predicted using the multi-task I/O predictor. Furthermore, these patterns can be saved as a guide map for future accesses, where the latency and cache miss occurs [10]. Since most application behaves predictably, the predictor uses Seq2Seq model to learn the storage access patterns and predict the next sequence of I/O stream. In addition, clustering and time analysis gave us great insight into identifying the most important I/O requests and the underlying structure of storage access patterns. Although the computation of these methods is challenging and time-consuming, the predictor employs them in offline training as they are a great asset in reducing the complexity of the predictor and increase its performance.

Although the preliminary results are promising, the work still faces challenges in meeting real-time requirements. For example, in the Financial 1 time analysis, delivering a 64ms ahead prediction requires more than 10 seconds of computation on our current platform. As the aforementioned significance of timeliness in I/O scheduling, we need to further fulfill the requirement of exact-time prediction by controlling the sequence length and window size. Also, additional hardware-software co-design is to be conducted to fulfill the real-time requirements.

In future work, we plan to tackle the memory wall problem that is caused by the bottleneck effects in memory [3]. Finally, the goal for that the proposed multi-task predictor is to be used as a hardware prefetcher that will revolutionize the performance of the application in cloud storage systems.

Acknowledgement

This work is supported by NSF grants CNS 1850851, C-Accel 1937833, CCF 1527249, 1717388, 1907765 and a startup grant from the University of Central Florida.

References

[1] G. E. Moore *et al.*, "Cramming more components onto integrated circuits," 1965.

[2] G. H. Loh, "3d-stacked memory architectures for multi-core processors," in *ACM SIGARCH computer architecture news*, vol. 36, no. 3. IEEE Computer Society, 2008, pp. 453–464.

[3] M. Hashemi, K. Swersky, J. A. Smith, G. Ayers, H. Litz, J. Chang, C. Kozyrakis, and P. Ranganathan, "Learning memory access patterns," *arXiv preprint arXiv:1803.02329*, 2018.

[4] B. S. Gill and D. S. Modha, "Sarc: Sequential prefetching in adaptive replacement cache." in *USENIX Annual Technical Conference, General Track*, 2005, pp. 293–308.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[6] J. Zhou, D. Dai, Y. Mao, X. Chen, Y. Zhuang, and Y. Chen, "I/O characteristics discovery in cloud storage systems," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, 2018, pp. 170–177.

[7] J. Li, X. Chen, E. Hovy, and D. Jurafsky, "Visualizing and understanding neural models in nlp," *arXiv preprint arXiv:1506.01066*, 2015.

[8] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[9] J. Xu, Xu, and McDermott, *Block Trace Analysis and Storage System Optimization*. Springer, 2018.

[10] M. Abebe, K. Daudjee, B. Glasbergen, and Y. Tian, "Ec-store: Bridging the gap between storage and latency in distributed erasure coded systems," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 255–266.

[11] UMASS. (2002) Oltp application i/o, search engine i/o. [Online]. Available: http:traces.cs.umass.edu/index .php/Storage/Storage

[12] D. Dai, F. S. Bao, J. Zhou, X. Shi, and Y. Chen, "Vectorizing disks blocks for efficient storage system via deep learning," *Parallel Computing*, vol. 82, pp. 75–90, 2019.

[13] Z. Li, Z. Chen, S. M. Srinivasan, and Y. Zhou, "C-miner: Mining block correlations in storage systems." in *FAST*, vol. 4, 2004, pp. 173–186.