SCALEML: Machine Learning based Heap Memory Object Scaling Prediction

Joongeon Park[1], Safdar Jamil[1], Awais Khan[1], Sangkeun Lee[2], Youngjae Kim[1]

[1]Dept. of Computer Science and Engineering, Sogang University, Seoul, Republic of Korea
[2]Computational Data Analytics Research Group, Oak Ridge National Laboratory, Oak Ridge, TN USA
{wnddjssla, safdar, awais, youkim}@sogang.ac.kr, lees4@ornl.gov

***Abstract*—Memory subsystem contributes 28-40% of total energy consumption. Several studies investigated energy prediction and consumption via profiling memory object access patterns. However, such profiling leads to higher energy consumption due to intense memory object-level profiling to achieve high prediction accuracy. Further, memory object access pattern prediction has been considered through analyzing the variation between memory object access patterns, referred to as scaling rate. The existing techniques for scaling rate prediction, such as Linear Scaling Rate (LSR), suffer from a high error rate in prediction with changes in access patterns, which leads to a high error rate of energy consumption prediction. In this paper, we compare and evaluate several memory object access pattern prediction models including LSR and machine learning (ML) models. Further, we propose SCALEML, a heap memory object scaling rate prediction mechanism that employs an ML model to achieve high access pattern prediction accuracy with variations in memory object access patterns. We evaluate SCALEML using various application benchmarks. The experimental results show that SCALEML achieves about 20% higher accuracy than the LSR model for predicting the scaling rate of object access patterns and energy estimation.**

***Index Terms*—Scaling Rate, Memory Object Access Patterns, Machine Learning**

I. INTRODUCTION

In recent years, the volume of data has grown exponentially. To entertain such data growth, organizations are deploying well-provisioned data centers [1]. A recent study stated that the volume of data being processed by data center has increased by 173% every year [2]. So, storing and analyzing such data at data centers also lead to high energy consumption and the emission of a tremendous mass of greenhouse gases. There are two major factors for the energy dissipation in data center systems, i.e., CPU and main memory. On a single server, CPUs account for 30%-60% of energy consumption, while main memory accounts for 28%-40% [3], [4]. Various works have been proposed to reduce the energy dissipation of CPU [5]–[7] and such techniques as powering down the memory banks and controlling base memory voltage and frequency [8], [9], are also adopted for memory-level energy consumption optimization.

In a hybrid memory system, software-based solutions to improve the memory-level energy efficiency have been proposed by considering the fine-grained object-level access patterns, such as size, lifetime, accessed volume, and last-level cache misses, of the applications [10]–[13]. However, object-level memory profiling of the applications is challenging as it is a time-consuming operation and elevates energy consumption. Further, whenever application workload changes, the object access patterns also vary [14]. Thus, predictions carried on the previously profiled access patterns become invalid, requiring additional time and profiling cycles.

A recent study [15] proposed Linear Scaling Rate (LSR) model to predict the memory object access patterns while minimizing the application profiling time. LSR assumes that object access patterns scale linearly with scaling application workloads. We refer to this variation as the *Scaling Rate* of memory object access patterns. For example, an application has a N size workload with a lifetime of the $object_i$ as L_i. If the workload size scales to double ($2 * N$), the lifetime of the $object_i$ will also scale linearly ($2 * L_i$). However, the linear scaling assumption does not hold for several applications resulting in a high prediction error rate, as discussed in Section II-B. A key reason includes variable behavior of memory objects access patterns towards the workload changes. Further, such randomness exists even among the memory objects generated by the same application. Therefore, it is critical to consider the scaling rate with respect to workload changes in order to predict access patterns and energy estimations accurately.

In this paper, we present a methodology of applying machine learning (ML)-based algorithms, in particular, Linear Regression (LR), Random Forest Regression (RFR), and K-Nearest Neighbor (K-NNR) for the prediction of the scaling rate of memory object access patterns. For this, we proposed SCALEML, a ML-based memory object access pattern's scaling rate prediction framework in conjunction with energy efficiency estimation. SCALEML adopts Two-Pass Memory Profiler (TPMP) [14] to extract object-level access patterns for various workloads of the applications and predicts optimal scaling rate for each memory object. For energy estimation,

978-1-7281-8482-1/20/$31.00 © 2020 IEEE

SCALEML utilizes the DRAM energy model proposed in [15]. Extensive evaluation, using two different benchmarks [16], [17] using different workloads (defined in Table III), shows that when using SCALEML, especially the RFR ML model shows a 20.1% higher memory object access pattern scaling accuracy and an 19.85% higher energy prediction accuracy than LSR.

II. BACKGROUND AND MOTIVATION

In this section, we present DRAM energy consumption estimation model and then discuss the motivation of our work.

A. DRAM Energy Consumption

For application-level energy consumption model, internal hardware components of the memory device and detail memory object access patterns are required to be considered. DRAM cells are made of capacitors which hold charge in terms of data and to perform read/write operation DRAM cells are activated and pre-charged (dE_{A+P}). In addition, a read operation discharges the DRAM cell (destructive nature) and the data needs to be maintained using a refresh operation (dE_{REF}). These internal memory device operations can be utilized as the memory commands and the energy consumption of these memory commands is fixed. On the other hand, application allocates memory objects and these memory objects' access behavior majorly affects the energy consumption at memory-level. Some of the vital memory object access patterns include: size (S_i), accessed volume (AV_i), cache misses and lifetime (L_i). Equation 1 considers all these details of the energy consumption for application-level at DRAM. Table I provides the normalized energy consumption per memory command.

$$DE_i = dE_{Pre} \cdot AV_i + dE_{RW} \cdot AV_i + dE_{REF} \cdot S_i \cdot T_i \quad (1)$$

B. Motivation

The memory object access patterns include size, accessed volume, and lifetime which collectively define the characteristics of an application. These access patterns change with the change in workload [14]. The rate at which access patterns alter with a varying workload is referred as scaling rate of object access patterns. Since every object has a different scaling rate for its all-access patterns and to calculate it, [15] proposed Linear Scaling Rate (LSR) mode. If an application has 'N' workloads on which it can execute than the access pattern (ap_i) such as size can be calculated using the Equation 2.

$$avg\_grad = \sum_{i=1}^{N-1} \{(ap_{i+1} - ap_i)/(in_{i+1} - in_i)\}/(N-1) \quad (2)$$

TABLE I: Energy consumption of memory command per byte [18]

Notation	Memory Command	Energy
dE_{Pre}	DRAM Precharge	3.07
dE_{RW}	DRAM Read/Write	1.19
dE_{REF}	DRAM Refresh	0.35

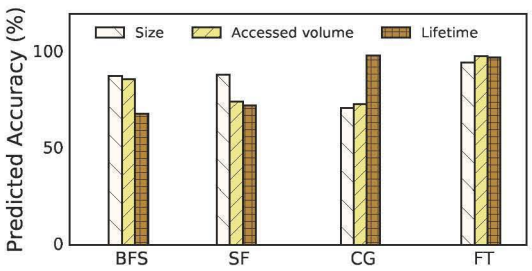


Fig. 1: Memory object patterns accuracy using LSR model

In Figure 1, we evaluated four applications from two benchmarks (explained in Table II) for the accuracy of the LSR predicted memory object access patterns using scaling rate from Equation 2 against the profiled memory object access patterns using TPMP [14]. The y-axis of Figure 1 represents the prediction accuracy and 100% represent perfect prediction. From Figure 1, we observe that all applications suffer from the error rate of the prediction of memory object access patterns through scaling rate with worst error rate of 68% for BFS application from PBBS benchmark. The high error rate in Figure 1 is due to little correlation between the memory object access patterns and the scaling workload. As the applications' workload increases the memory object access patterns do not scale linearly. Therefore, it is critical to consider the changes in the memory object access patterns with respect to workloads in order to accurately predict the scaling rate of the memory object access patterns. Thus, we evaluate various machine learning models for accurate scaling rate prediction to be utilized in the estimation of the energy efficiency.

III. MACHINE LEARNING MODELS

In this section, we provide an overview of the machine learning models including RFR, LR, and k-NNR which are candidate models for the comparative analysis of the prediction of scaling rate of the memory object access patterns.

A. Linear Regression (LR)

LR predicts new data by drawing the linear pattern that exists between the data-set and because of this linear pattern the accuracy of the prediction is high if there is a linear pattern. In addition, since the output is predicted based only on the linear pattern, the relationship between the patterns is not complicated and has a characteristic of a light-weight prediction model. In terms of predictive accuracy for the scaling rate of memory object access patterns, most of the object access patterns exhibit a linear relationship between workloads, so it can work well. As there is huge correlation between the object access patterns of the workload, LR does

predict the scaling accurately. Therefore, LR is suitable for predicting various object patterns because it can reflect the characteristics of object access patterns in prediction.

B. K-Nearest Neighbor Regression (K-NNR)

K-NNR is a method of predicting new data from the average of the k nearest neighbors among existing data. Therefore, it is important to select the k neighbors in efficient manner. If k is small, this should be considered because it may overfitting by overreflecting the local characteristics of the data. In addition, when k is large, the learning model tends to be excessively normalized, causing underfitting. In terms of predictive accuracy for the scaling rate of memory object access patterns, If there are correlations between objects as neighbors, an accurate scaling rate can be predicted. K-NNR has selected as one of the models of ML to verify the correlation between memory objects within the application and the prediction of scaling rate. In addition, K-NNR is a light-weight model because it predicts the output through k neighbors that are not complex relationships like LR.

C. Random Forest Regression (RFR)

RFR is a learning method that uses ensemble techniques to predict regression. During the learning process, multiple decision trees are constructed and each tree consists of different data values and the output is the averaged values of the predicted values. The features of RFR makes it a suitable candidate for the prediction of the relatively accurate scaling rates for memory objects as RFR can independently learn the change in each access pattern of memory object as the workload changes. The special properties of RFR include: 1) Each tree gets random samples that are different from the whole data when it is split, so it has a randomness to avoid over-fitting. 2) Each tree is divided into different data-set which does not interact with each other, so we get independent results. These features make the model more accurate by reflecting the unique scaling rate of various object access patterns in the training phase. For instance, when the size of the workload changes, the number of trees that make up the RFR also changes according to the pattern of the change in scaling rate. The more irregular the object access pattern, the more trees are learnt to increase accuracy. Therefore, the number of trees constituting the RFR according to the object access pattern is learnt and reflected in the prediction.

IV. ScaleML Design

This section provides an overview of the proposed ScaleML's design and describes the methodology for the prediction of the scaling rate of memory object access patterns.

A. Overview

Figure 2 shows the workflow of ScaleML for extracting memory object access patterns and the prediction of the scaling rate of memory objects for the workload that are not profiled. The workflow of ScaleML is composed of profiling phase, Figure 2(a), and training and prediction phase, Figure 2(b).

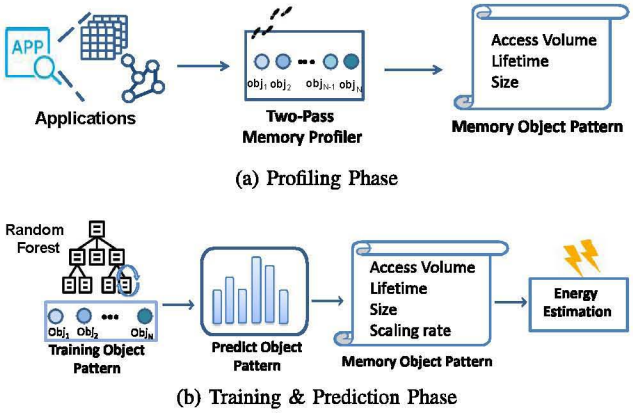


Fig. 2: ScaleML: scaling rate prediction workflow

Figure 2(a) shows how we extract the memory object access patterns and characterize them using the TPMP [14]. In the profiling phase, we adopted the TPMP [14] and extract the fine-grained object access information for each dynamically allocated memory object. Whereas, Figure 2(b) shows an applied ML model to train on the extracted memory object access patterns and obtain a scaling rate of each memory object for training and prediction phase. In the training and prediction phase, we use the ML model such as RFR to train on the extracted memory object access patterns and predict the optimal scaling rate of memory objects for new scaled workloads of the applications.

ScaleML can be utilized not only in a memory system composed of DRAM, but also in a memory system composed of other types of memory, i.e., Non-Volatile Memory (NVM). ScaleML predicts the optimal scaling rate, through which the object access patterns of an application can be predicted and by using Equation 1, the estimated energy consumption can be calculated. In addition, by utilizing the energy model of other memory devices such as NVM, estimation of energy consumption can be predicted as well.

B. Profiling Phase

The memory-level energy consumption of an application can be estimated using the detailed memory object access patterns of the application. Specifically, accessed volume, lifetime, and size of the memory objects play a vital role in the energy consumption. To extract the required information of memory objects, we adopted TPMP [14] which profiles the application's heap memory objects and gives all the necessary information required to estimate the energy consumption. Figure 3 shows the working flow of the TPMP. TPMP consists of two operational phases, *Fast Pass* and *Slow Pass*. Both passes are executed before the actual execution of the application. In the fast pass, the object identifiers and preliminary information of the objects such as lifetime and size are extracted. In the slow pass, the detailed profiling of the application is done to extract all the required information with the help of customized Pin-Tool [19].

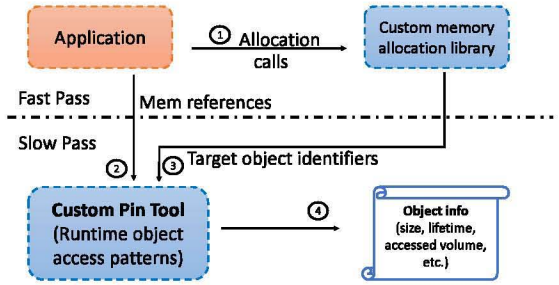


Fig. 3: Workflow of TPMP [14]

TPMP provides a wrapper library to hook the dynamic memory (de)allocation calls, such as malloc, calloc, realloc and free, and uses the call stack of the application to create object identifiers in terms of hash values. Fast pass creates these hash values while the slow pass identifies the target memory objects using the hash identifiers. In addition, TPMP also provides a customized Pin-Tool library so that the required information can be extracted at the instruction level. Using the customized Pin-Tool, one can extract only those information that is necessary and skip other unrelated access patterns. On one hand, profiling applications using the TPMP gives the required object-level access pattern but on the other hand, the TPMP consumes a huge amount of time to profile due to instruction-level profiling which leads to higher performance overhead and energy consumption. Therefore, to increase the efficiency and reduce the profiling time, we collect object access patterns and predict the scaling rate using the RFR model of ML for scaled workloads of applications that are not profiled.

C. Training and Prediction Phase

In this phase, we train our compared ML models using the object access patterns extracted from the profiling phase to calculate the scaling rate of each memory object for the scaled workloads of the application. We train candidate ML models such as LR, RFR, K-NNR. The input data used for training is information about each memory object's hash value (object identifier), accessed volume, size, lifetime, and scaling rate. The input data-set for training has fields like, hash value (object identifier), accessed volume, size, lifetime, and LSR-based scaling rate.

In addition, the correlation between memory object access patterns is also considered and reflected in the training phase. The correlation between memory objects access patterns means if the change in one access pattern affects the other access pattern information. For example, among all the object access patterns being considered, there is a positive correlation between the accessed volume and the lifetime of the objects as the increase in one will increase the other automatically. On the other hand, the size of the memory objects of application does not have correlation with other memory object access patterns so while predicting the scaling rate of the memory object's size correlation is not taken into account.

In addition, for RFR, for the accuracy of the training model, the number of trees constituting the RFR are also being considered. Since the accuracy of the model varies depending on the number of trees, each application finds a point where the accuracy is saturated according to the number of trees and reflects it in the training process. Models with high accuracy are selected and used in the prediction phase. Details on changing the accuracy are described in Section V.

In last, from the compared ML models, the model that obtains the highest prediction accuracy of the scaling rate of the memory object access patterns is chosen. Once the scaling rate is predicted, the workload access patterns can be estimated using the scaling rate for each memory object and the energy consumption can be approximated using the Equation 1 from section II-A.

TABLE II: Experimental setup

Parameter	Configuration
CPU	Intel Core i7 8700 CPU, 6 core, 3.2GHz
Main Memory	16GB DDR4 1340MHz
Interface	PCIe 3.0 x8
Benchmark	PBBS [16], NPB [17]

TABLE III: Application workload configurations of Benchmarks [16], [17]

BFS(Vertex)	SF(Vertex)	CG(Num of row)	FT(Grid size)
$25 * 10^4$	$25 * 10^4$	$14*10^2$	64 x 64 x 64
$50 * 10^4$	$50 * 10^4$	$70 * 10^2$	128 x 128 x 32
$10 * 10^5$	$10 * 10^5$	$14 * 10^3$	256 x 256 x 128
$20 * 10^5$	$20 * 10^5$	$75 * 10^3$	512 x 256 x 256
$40 * 10^5$	$40 * 10^5$	$15 * 10^4$	512 x 512 x 512

V. Evaluation

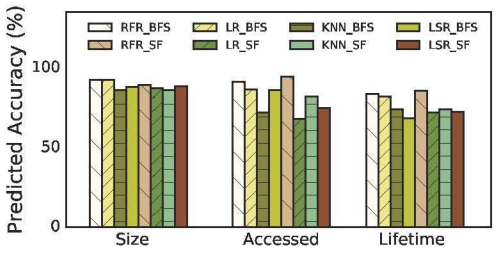
This section provides the experimental setup followed by an extensive evaluation of the proposed methodology.

A. Experimental Setup

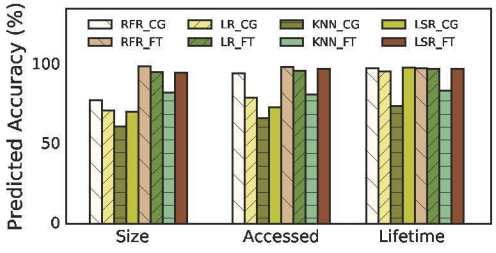
The experimental setup and benchmark are listed in Table II. We used two scientific application benchmarks, Problem Based Benchmark Suite (PBBS) [16] and NAS Parallel Benchmark (NPB) [17]. Breadth First Search (BFS) and Spanning Forest (SF) are the applications from PBBS while Conjugate Gradient (CG) and 3D fast Fourier Transform (FT) are from NPB benchmark. Additionally, for ML, we adopted the ASCENDS Tool [20]. Table III shows five different workload metrics for each application. We used four workloads for training the ML models, whereas, last workload metric is used for prediction accuracy estimation.

B. Prediction Accuracy of ML Models

Figure 4 shows the accuracy of the scaling rate of the model trained through RFR, K-NNR, LR, and LSR. Figure 4(a) shows the comparison result for PBBS benchmark where RF_BFS, LR_BFS, LRS_BFS, KNN_BFS, RF_SF, LR_SF,

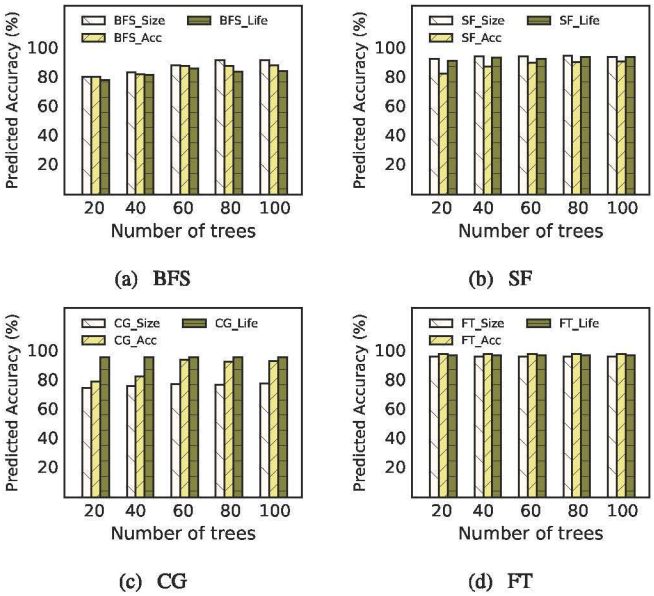


(a) Accuracy of ML Model in PBBS benchmark



(b) Accuracy of ML Model in NPB benchmark

Fig. 4: Comparative analysis of prediction accuracy of various ML models



(a) BFS (b) SF

(c) CG (d) FT

Fig. 5: Prediction accuracy with varying number of trees in RFR model. X_Size, X_Acc and X_Life depicts application with specific object property, i.e., size, access, and lifetime.

LSR_SF and KNN_SF, represents RF, LR, LSR, and KNN models being applied for the prediction of BFS and SF applications scaling rate. Figure 4(b) shows the comparison result for NPB benchmark where RF_CG, LR_CG, KNN_CG, LSR_CG RF_FT, LR_FT, LSR_FT, and KNN_FT represents RF, LR, and KNN models being applied for the prediction of CG and FT applications scaling rate. When predicting the patterns of all considered memory objects, the accuracy using the RFR is highest. In particular, when predicting the accessed volume in Figure 4(b), the RFR_FT shows a difference of up to 16% accuracy compared to the LR_FT. Also, when predicting the lifetime in Figure 4(b), RFR_CG shows a difference of up to 23.6% compared to K-NNR_FT. In BFS application, the RFR showed an average accuracy of 2.2% and 11.6% higher than the LR and K-NNR, and the SF application averaged 11.8% and 8.3% higher. The RFR of CG applications showed an average of 9.7% and 21.2% higher accuracy than the LR and K-NNR, and the average of 0.3% and 16.1% of accuracy in the FT application. For all the applications, we observed that RFR outperforms other ML models. RFR is more accurate than other ML models because it independently predicts object patterns with irregular scaling rates. Therefore, RFR is used as a learning model for predicting memory object patterns.

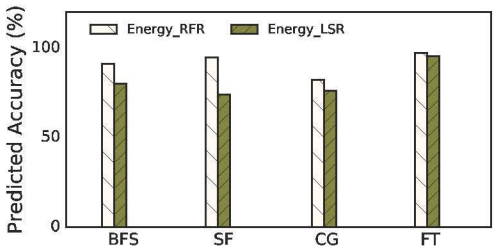
C. Prediction Accuracy with Varying RFR Trees

In this experiment, we evaluated the RFR model with varying number of trees to find the optimal number of trees for high prediction accuracy. The accuracy represents the difference between the actual profiled object access patterns using TPMP and the predicted object access patterns using scaling rate.

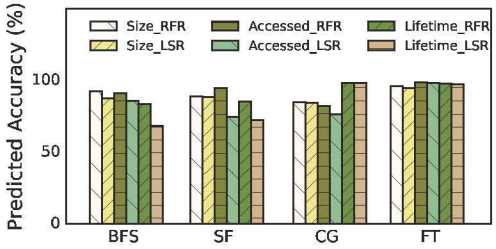
Figure 5 shows the point of saturation of accuracy through changes in the tree that composes the RFR for four applications from two benchmarks and their object access patterns. The BFS_Size and BFS_Acc are saturated when the number of trees are 80 as shown in Figure 5(a) as these object access patterns do not scale linearly. Besides, the BFS_life saturates at 60 number of trees as some of the object's lifetime scales linearly, while some object's lifetime do no scale linearly. This is the reason that BFS_life saturates at less number of trees than BFS_Size and BFS_Acc. Furthermore, the object access patterns of SF and CG applications also saturate at 60 decision trees of RFR. On the other hand, memory object access patterns of FT application and CG_life show a constant accuracy regardless of the number of trees due to linear scaling rate of memory object access pattern. This means that even if we train the data-set randomly, it is irrelevant to the number of trees because all object access patterns show the similar scaling rate. When training the BFS_Size, it showed up to 11.8% accuracy improvement, and when learning the CG_Acc, the accuracy improved by at least 3.6% over other memory object access patterns of the applications, respectively. Therefore, for accurate object access pattern prediction, training is performed with 80 number of trees.

D. Energy Consumption Comparison

In this experiment, we compared the accuracy of the scaling rate and estimated energy consumption of predicted memory object access patterns by ML and LSR approach. As shown in the 6(a), The goal is to estimate the energy consumption with high accuracy by using the accurate memory object pattern predicted through the scaling rate.



(a) Comparison of energy consumption with ML model



(b) Comparison of prediction accuracy with ML model

Fig. 6: Prediction accuracy and estimated energy consumption of RFR and LSR model.

Figure 6(a) shows the comparison between the estimated energy efficiency computed using the RFR model and the LSR model. In particular, it shows that the RFR model has higher prediction accuracy of energy estimation than the LSR method. In SF application, the prediction accuracy of RFR model is up to 19.84% higher than that of the LSR method. The reason why estimated energy consumption accuracy of RFR model is higher than LSR model is shown in Figure 6(b). It shows the prediction accuracy comparison of memory object access patterns computed using the scaling rate predicted by RFR and the LSR. It indicates that the RFR prediction is more accurate than the LSR method. In particular, when predicting the accessed volume in SF applications, the RFR model prediction is 20.1% higher than the scaling rate predicted through the LSR method. The accuracy of the memory object access pattern predicted using the RFR model is 92.85% on average, and the accuracy of estimated energy consumption is 91.3% on average higher than LSR, respectively. Therefore, the prediction of the memory object access pattern and estimated energy consumption using the RFR model is highly efficient than the LSR method.

VI. Conclusion

Memory object access patterns define the characteristics of the application and can be utilized to estimate the object-level energy consumption. The estimation of the energy dissipation requires fine-grained profiling information of memory object access patterns. Memory object access patterns vary with the workload of the application which leads to additional profiling overhead and huge energy consumption. In this paper, we have proposed ScaleML, which provides a comparative analysis of LSR, LR, K-NNR, and RFR for the prediction and accuracy of the scaling rate of memory object access patterns. ScaleML predicted the accuracy of scaling rate of the memory object access patterns and estimated energy consumption and showed that it has 20.1% and 19.85% higher than the LSR and other ML models, respectively.

Acknowledgements

This research was supported by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (2017M3C4A7080243). Y. Kim is the corresponding author.

References

[1] A. Khan, M. Attique, and Y. Kim, "iStore: Towards the optimization of federation file systems," *IEEE Access*, pp. 65652–65666, 2019.

[2] Z. Jia, L. Wang, J. Zhan, L. Zhang, and C. Luo, "Characterizing data analysis workloads in data centers," in *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*, pp. 66–76, 2013.

[3] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys Tutorials*, pp. 732–794, 2016.

[4] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. W. Keller, "Energy management for commercial servers," *IEEE Computer*, vol. 36, no. 12, pp. 39–48, 2003.

[5] E. Calore, A. Gabbana, S. Fabio Schifano, and R. Tripiccione, "Evaluation of DVFS techniques on modern hpc processors and accelerators for energy-aware applications," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 12, p. e4143, 2017.

[6] E.-Y. Chung, L. Benini, A. Bogliolo, Y.-H. Lu, and G. D. Micheli, "Dynamic power management for nonstationary service requests," *IEEE Transactions on Computers*, no. 11, pp. 1345–1361, 2002.

[7] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott, "Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling," in *Proceedings of the Eigth International Symposium on High Performance Computer Architecture*, pp. 29–40, 2002.

[8] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, "Memory power management via dynamic voltage/frequency scaling," in *Proceedings of the 8th ACM International Conference on Autonomic Computing*, ICAC '11, (New York, NY, USA), pp. 31–40, ACM, 2011.

[9] I. Hur and C. Lin, "A comprehensive approach to DRAM power management," in *IEEE 14th International Symposium on High Performance Computer Architecture*, pp. 305–316, 2008.

[10] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *Proceedings of the IEEE 10th International Symposium on Workload Characterization*, pp. 171–180, 2007.

[11] B. Liu, Yinan Lin, and Y. Chen, "Quantitative workload analysis and prediction using google cluster traces," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2016.

[12] W. Fang, Z. Lu, J. Wu, and Z. Cao, "RPPS: A novel resource prediction and provisioning scheme in cloud data center," in *Proceedings of the IEEE Ninth International Conference on Services Computing*, pp. 609–616, 2012.

[13] A. Narayan, T. Zhang, S. Aga, S. Narayanasamy, and A. Coskun, "MOCA: Memory object classification and allocation in heterogeneous memory systems," in *Prooceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2018.

[14] X. Ji, C. Wang, N. El-Sayed, X. Ma, Y. Kim, S. S. Vazhkudai, W. Xue, and D. Sanchez, "Understanding object-level memory access patterns across the spectrum," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 25:1–25:12, 2017.

[15] T. Kim, S. Jamil, J. Park, and Y. Kim, "Optimizing heap memory object placement in the hybrid memory system with energy constraints," *IEEE Access*, vol. 8, pp. 130323–130339, 2020.

[16] J. Shun, G. E. Blelloch, J. T. Fineman, P. B. Gibbons, A. Kyrola, H. V. Simhadri, and K. Tangwongsan, "Brief Announcement: The problem based benchmark suite," in *Proceedings of the Twenty-fourth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '12, pp. 68–70, 2012.

[17] D. H. Bailey, *NAS Parallel Benchmarks*, pp. 1254–1259. Boston, MA: Springer US, 2011.

[18] E. Kültürsay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating STT-RAM as an energy-efficient main memory alternative," in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 256–267, 2013.

[19] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building customized program analysis tools with dynamic instrumentation," in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '05, 2005.

[20] S. Lee, J. Peng, A. Williams, and D. Shin, "Ascends: Advanced data science toolkit for non-data scientists," *Journal of Open Source Software*, p. 1656, 2020.