DRAM-PCM 하이브리드 메인 메모리에 대한 동적 다항식 회귀 프리페처

ZHANG MENGZHAO, 김정근, 김신덕
연세대학교 컴퓨터과학과
symbrio@naver.com, junggeun@yonsei.ac.kr, sdkim@yonsei.ac.kr

Dynamical Polynomial Regression Prefetcher for DRAM-PCM Hybrid Main Memory

Mengzhao Zhang, Jung-Geun Kim, and Shin-Dug Kim
Dept. of Computer Science, Yonsei University

Abstract

This research is to design an effective prefetching method required for DRAM-PCM hybrid main memory systems especially used for big data applications and massive-scale computing environment. Conventional prefetchers perform well with regular memory access patterns. However, workloads such as graph processing show extremely irregular memory access characteristics and thus could not be prefetched accurately. Therefore, this research proposes an efficient dynamical prefetching algorithm based on the regression method. We have designed an intelligent prefetch engine that can identify the characteristics of the memory access sequences. It can perform regular, linear regression or polynomial regression predictive analysis based on the memory access sequences' characteristics, and dynamically determine the number of pages required for prefetching. Besides, we also present a DRAM-PCM hybrid memory structure, which can reduce the energy cost and solve the conventional DRAM memory system's thermal problem. Experiment result shows that the performance has increased by 40%, compared with the conventional DRAM memory structure.

1. Introduction

With the rapid development of big data analysis and the increasing volume of generated data, the requirement for database efficiency has become increasingly urgent. Considering the demand of low latency performance, several in-memory-based database systems such as SPARK have become a popular trend for data processing.

In-memory configuration offers nearly the best power consumption and performance compared to other parallel disk systems. To obtain a fast response [1], in-memory processing allocates all the data needed in the main memory to minimize the hard disk accessing time [2]. However, this worsens the bottleneck effect of the memory capacity. Unfortunately, in real database systems, the capacity of the main memory has limitations. When the memory space that the database requires is beyond the main memory's capacity, data in the main memory will be constrained to move to an auxiliary storage, which is a considerable drawback for in-memory processing. Various studies regarding prefetching between memory layers aimed to prevent this problem, but in real-world circumstances, accessing the auxiliary storage layer is inevitable. A few studies have focused on prefetching between the main-memory layer and the auxiliary storage layer.

Furthermore, in most commercial database applications, data are positively associated with large-scale graphs. Although several studies have focused on data analysis and mining, little attention has been paid to graph computing [3].

Therefore, in this study, a dynamic recognition prefetch engine associated with a DRAM-PCM hybrid memory is proposed. The prefetcher functions dynamically between the main-memory layer and the auxiliary storage layer with machine learning technology. Specifically, any memory request from the last-level cache will be preprocessed first, then a regression calculation will be performed. Finally, the prefetcher will determine whether to perform prefetching. Moreover, the main-memory layer includes a DRAM and PCM module that can provide exceptional cost efficiency and performance [3], [4].

Experiments show that the performance has increased by 40%, compared with the conventional DRAM memory structure. And also compared with the latest prefetch algorithm, it has also increased by 3%.

2. Related Work

Memory is always a bottleneck for the performance of an entire modern system. Thus, prefetchers are used to predict and pre-load data, which can then be accessed by the system to use directly instead of waiting to read data from a

lower-level storage.

Theories for predicting data are generally based on time and spatial locality. Prefetching schemes such as GHB-PC/DC and stride can compare differences between data addresses. They are highly accurate and efficient for regular access patterns [5], [6]. However, one of the most distinguishing characteristics of graph processing is irregular memory access patterns [7]. In addition, both SMS and Sarsa prefetchers do not perform adequately when workloads have either a low spatial locality or low semantic locality [8], [9].

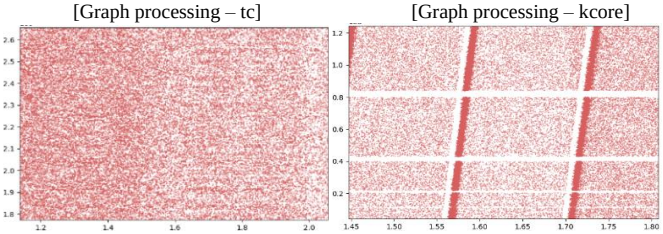

Fig. 1. Irregularity of memory request patterns

Fig. 1 presents the irregularity of memory requests in graph processing. The x-axis denotes the data accessing sequence in time and the y-axis indicates the address value. According to the figure, access patterns of graph processing cover a massive range of memory space without several visible regular characteristics.

To address these issues, we need a more intelligent and aggressive method than conventional techniques. However, aggressive prefetching methods come with risks of misprediction, which could lead to the failure of the entire system. Zhuang et al. and JT Yun et al. proposed a request table method to reduce the risk of inaccurate prefetching by distinguishing and discarding bad prefetching results [10], [11]. Moreover, JT Yun et al. took a further step considering machine intelligence by proposing a simple linear regression prefetcher with preprocessing

In JT Yun et al. 's structure, the data address requested from the last level cache to the main memory will be recorded in the request table. It pre-processes the data in the request table, divided into three groups according to the address value. Among the three groups A, B, and C, the most extensive data sample size will be regarded as a hot zone. Then use 2^20 (256 pages) address space as a threshold to denoise the data in the hot zone and remove the address value exceeding 256 pages [11]. At this point, the pre-processing is complete. After pre-processing, perform simple linear regression on the hot zone's denoising data to obtain the predicted value.

Although such an algorithm is simple and easy to calculate, it also contains several potential problems. First, the pre-processing method is too radical and straightforward. As shown in Figure 2, the graph processing access patterns are evenly and widely distributed in each address range. After simply dividing the data into three groups, the difference in data sample size between groups is not apparent. That is, the divergence between the hot zone and the cold zone is not clear. So, it cannot ensure that the next access address must be in the hot zone. Because of using that as a basis to perform linear regression calculation, the result obtained has a high risk of misprediction.

Secondly, the model uses a simple linear regression to make predictions. It takes the most recent access request as a reference and returns a single prediction result. In actual operation, the prefetcher preloads the data in the address space of the calculated prediction result and 4KB after it (i.e., 1 page). According to the linear regression method's property, if the access pattern conforms to the linear regression characteristics, the next actual access address value A should obey the Gaussian distribution with the predicted result $\hat{A}$ as the mean and the σ^2 as the variance.

$$A \sim N(\hat{A}, \sigma^2)$$

Selecting only the prediction result $\hat{A}$ and the address space of the following 4KB has an obvious error. Therefore, this study provides a scheme for dynamic prefetching based on the Gaussian distribution and confidence intervals.

Last but not least, in terms of machine intelligence algorithms, the model only uses simple linear regression as its predictive method. When the access pattern's linear feature is fragile (linearly independent), the model cannot provide correct prefetch results. For this reason, this article uses the polynomial regression scheme to ensure that linear regression will be performed when the access mode is linearly related, and the polynomial regression algorithm will be performed when linearly independent [12], [13].

3. Proposed Algorithm

3.1 Pre-processing

In our proposed model, the first step is to classify and preprocess the memory requests. Because the access patterns of the graph processing will cover an expansive memory space, comprehensive statistics on all requests will cause significant overhead and latency. Therefore, we applied a secondary table mechanism that matches the global and local history tables to comprehensively and efficiently collect memory requests.

The first-level table is a global table that records the higher 46 bits of the fetch address. The global table will monitor all the memory space and make updates following the first in first out (FIFO)'s principle according to time locality. The second-level table will correspond to each entry in the first-level table and record the lower 18 bits of the corresponding address, that is, the offset of each entry in the global table. The second-level table follows the spatial locality property and will update in the LRU mode as the local table. Grouping data according to high bits of addresses can not only monitor the entire address space, but also effectively reduce the amount of calculation for regression analysis.

3.2 Regular Patterns Prefetching

After preprocessing the data, we analyze each entry in the global table. Owing to an apparent feature of regular pattern prefetching, which is low overhead, we first determine whether memory access sequences are regular patterns. We adopt a method similar to comparing the offset differences in GHB-PC/DC and stride algorithms to analyze the data. If the delta between the request sequences has prominent regular characteristics, prefetching will be processed.

3.2 Polynomial Regression

If there are no prominent regular characteristics in the memory access sequence, we switch the prefetch mode to regression analysis. Regression analysis is a statistical analysis method widely used in the field of machine learning. It fits the most appropriate hypothetical curve to existing data and predicts the data's potential trend through this curve [14], [15]. When applied to machine learning, the existing data are

used as the training set; as the number of data increases, the regression parameters are continuously learned and modified to dynamically generate prediction results [16]. Regression analysis methods are divided into unary regression and multiple regression analyses.

In this study, we use the least-squares method for unary regression analysis. We consider the time sequence corresponding to each offset in the local table as X and the specific address value as Y, and then sort out a set of (X, Y) pairs as the training data for machine learning algorithms. The unary regression analysis includes linear regression and polynomial regression analysis.

To prevent excessive overhead and over-fitting, we limit the highest power of the polynomial regression to the second power and calculate the coefficient matrix by the Gaussian–Jordan elimination method [17].

3.3 Dynamical Prefetching

In linear regression analysis, the prediction result should obey the characteristics of the Gaussian distribution. We choose to dynamically increase the number of prefetched memory pages based on the confidence interval. When the prefetcher performs linear regression with the highest power of 1, we calculate the [Aˆ-σ, Aˆ+σ] interval, which has a confidence of 66.7%. Then, we compare this interval with the 12 KB address space (3 pages) and consider the intersection. As a result, the dynamical prefetcher could pre-load 1–3 pages depending on the confidence interval.

All prefetched memory pages are stored in an independent prefetch buffer. Before it is stored in the prefetch buffer, a redundancy check will be performed to ensure that there is no duplicate data in the DRAM-PCM main memory.

4. Performance Evaluation

4.1 GraphBig

In actual commercial database workloads, graphs play a key role because big data applications that consist of entities with internal links naturally form a graph.

For further investigating graph processing, Lifeng Nai et al. from IBM proposed the GraphBig dataset generated based on real-world cases (e.g., Facebook) as a benchmark.

4.2 Simulation Configurations

We chose GraphBig workload benchmarks to evaluate our proposed method and gathered the CPU access trace using PinTool [18]. The workloads are listed in Table 1. We used a trace-driven simulator to simulate the first, second, and third levels of cache as well as the DRAM-PCM hybrid main memory. The proposed system configuration and simulation parameters are listed in Tables 2 and 3 [19], [20].

<Table 1> Workloads of Graph processing

Workload	**Using cases**
BFS	Recommendation for Commerce
Connected com-ponent(CCOMP)	Social Media Monitoring
Degree centrality(Dcentr)	Social Media Monitoring
Shortest(SPath)	Smart Navigation
Triangle count(TC)	Data Curation for Enterprise

4.3 Experiment Result

Fig. 2 and 3 present the comparison results of the execution time and energy consumption. We measured the execution times and energy consumptions of different prefetching algorithms under different workloads, compared the conventional DRAM-PCM memory structure without prefetching as the benchmark value, and calculated the normalized execution time and energy consumption.

According to the resulting figures, our model reduced the execution time by a maximum of 50% compared to the conventional DRAM-PCM structure without prefetcher, and the energy consumption was reduced by 21%. Our model also reduced the execution time by 40% and the energy consumption by 18%, on average. Compared to the model proposed by JT Yun, which is state of the art, it excelled in performance by approximately 3% in both execution time and energy consumption.

<Table 2> Proposed System Configuration

Processor	Quad-cores, 4GHz
Cache Layer	L1I & L1D: 32KB, 8-ways L2: 256KB, 8-ways L3: 8MB, 16-ways 64byte block size, LRU replacement
Prefetch Buffer (DRAM)	16MB, 4KB page size, Fully associativity LRU replacement
DRAM Memory	128MB, 4KB page size, Fully associativity LRU replacement
PCM Memory	2GB, 4KB page size, Fully associativity LRU replacement

<Table 3> Simulation Parameters

Parameter	DRAM	PCM	HDD
Write Latency	20~50ns	~1us	~5ms
Read Latency	20~50ns	~50ns	~5ms
Write Energy	1.2J/GB	6J/GB	65J/GB
Read Energy	0.8J/GB	1J/GB	56/GB
Idle power	~100mW/GB	~1mW/GB	~10W/TB
Density/Cost	1x / 4x	4x / 1x	N/A

Fig. 4 presents the PCM lifetime under different prefetching algorithms. The system life calculation formula is as follows [cal18]

$$\text{PCM Lifetime} = \frac{W * S}{B}$$

Where S is the size, B is the traffic of writes, and W is the limitation of cell endurance. Compared with other algorithms, our model improves the lifetime of PCM by 6-24% on average.

5. Conclusion

In summary, this article provides a prefetch model for the DRAM-PCM hybrid main memory structure that can dynamically prefetch multiple pages based on the machine learning algorithm of polynomial regression. In our model, the memory request from LLC will be preprocessed first, followed by a regression calculation. Finally, the prefetcher will determine whether to perform prefetching, as well as the number of prefetched pages according to the confidence interval of the calculation.

The experiments proved that this model has sufficient adaptability to irregular storage models. Compared with the performance of conventional DRAM-PCM memory without prefetcher, the performance of our model increased by 40%; compared with the latest prefetch algorithm, it also increased by 3%.

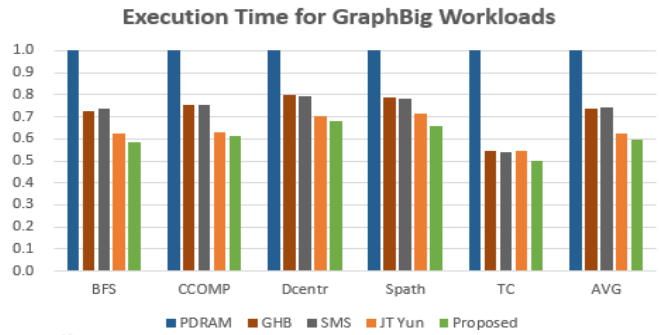

Fig. 2. Norm. Execution Time

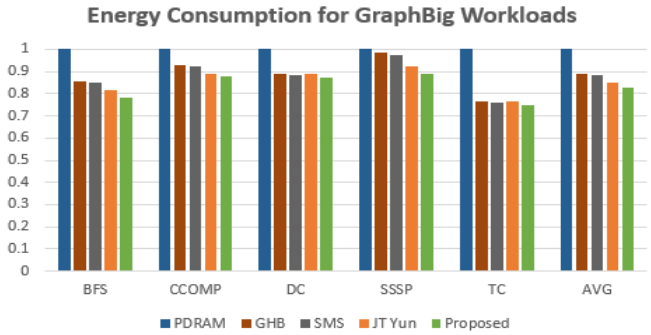

Fig. 3. Norm. Energy Consumption

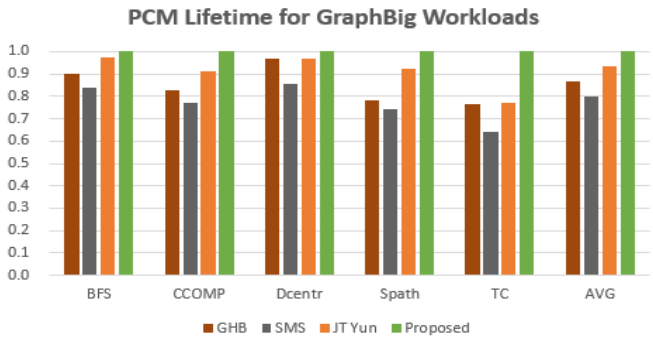

Fig. 4. Norm. PCM Lifetime

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (NRF-2019R1A2C1008716).

REFERENCES

[1] Hasso Plattner, Alexander Zeier, “In-Memory Data Management”, Heidelberg, Germany, Spriger, 2011.

[2] T. Jiang, Q. Zhang, R. Hou, L. Chai, S.A. McKee, Z. Jia, N. Sun, “Understanding the behavior of in-memory computing workloads”, IEEE International Symposium on Workload Characterization (IISWC), 2014, pp. 22–30.

[3] M. K. Qureshi, et al., “Scalable high performance main memory system using phase-change memory technology”, in Proc. 36th Annu. Int. Symp. Comput. Archit., 2009, pp. 24–33.

[4] S.-K. Yoon, et al., “Optimized memory-disk integrated system with DRAM and nonvolatile memory”, IEEE Trans. MultiScale Comput. Syst. vol. 2, no. 2, pp. 83–93, Apr.-Jun. 2016.

[5] K. J. Nesbit and J. E. Smith, “Data cache prefetching using a global history buffer”, in Proc. Int Symp. High Perform. Comput. Archit., 2004, pp. 96–105.

[6] J. W. C. Fu, et al., “Stride directed prefetching in scalar processors”, in Proc. IEEE/ACM Int. Symp. Microarchitecture, 1992, pp. 102–110.

[7] L. Nai, et al., “GraphBIG: Understanding graph computing in the context of industrial solutions”, in Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal., 2015, pp. 1–12.

[8] S. Somogyi, et al., “Spatial memory streaming”, in Proc. Int. Symp. Comput. Archit., 2006, pp. 252–263.

[9] LIANG Yuan, et al., “Prefetching Algorithm of Sarsa Learning Based on Space Optimization”, Computer Science, vol. 46, no. 3, pp. 327-331, Mar. 2019

[10] X. Zhuang, et al., “A hardware-based cache pollution filtering mechanism for aggressive prefetches”, Proc. 32nd Int. Conf. Parallel Process., 2003, pp. 286–293.

[11] Ji-Tae Yun, et al., “Regression Prefetcher with Preprocessing for DRAM-PCM Hybrid Main Memory”, IEEE COMPUTER ARCHITECTURE LETTERS, vol. 17, no. 2, pp. 163-166, JULY-DECEMBER 2018

[12] KANI CHEN, ZHEZHEN JIN, “Local polynomial regression analysis of clustered data”, Biometrika, vol. 92, no. 1, pp. 59-74, 2005

[13] Eva Ostertagová, “Modelling using polynomial regression”, Procedia Engineering, vol. 48, no.1, pp. 500-506, 2012

[14] Mosteller, F. and Tukey, J.W. Data Analysis and Regression: A Second Course in Statistics. Reading, MA: Addison-Wesley, 1977

[15] Gelman, A. and Hill, J. Data Analysis Using Regression and Multilevel/Hierarchical Models. Cambridge University Press, 2006

[16] Caruana, R., Niculescu-Mizil, A.: An empirical comparison of supervised learning algorithms. In: ICML, pp. 161168, 2006

[17] Peters, G.,Wilkinson, J.H.: On the stability of Gauss-Jordan elimination with pivoting. Comm. Assoc. Comput. Mach. 18, 2024,1975

[18] C.-K. Luk, et al., “Pin: Building customized program analysis tools with dynamic instrumentation”, in Proc. ACM SIGPLAN Conf. Program. Language Des. Implementation, 2005, pp. 190–200.

[19] S. Chen, P. B. Gibbons, and S. Nath, “Rethinking database algorithms for phase change memory”, in Proc. CIDR, 2011, pp. 21–31.

[20] K. H. Park, et al., “Mn-mate: Resource management of many cores with dram and non-volatile memories”, Proc. 12th IEEE Int. Conf. HPCC, Sep. 2010, pp. 24–34.