반도체디스플레이기술학회지 제22권 제2호(2023년 6월)
Journal of the Semiconductor & Display Technology, Vol. 22, No. 2. June 2023.

그래프 프로세싱을 위한 GRU 기반 프리페칭

시바니 자드하브[*]·파만 울라[*]·나정은[**]·윤수경[*,***†]

[*]전북대학교 컴퓨터인공지능학부, [**]연세대학교 학부대학, [***†]영상정보신기술연구센터

Gated Recurrent Unit based Prefetching for Graph Processing

Shivani Jadhav[*], Farman Ullah[*], Jeong Eun Nah[**] and Su-Kyung Yoon[*,*†]**

[*]Department of Computer Science and Artificial Intelligence, Jeonbuk National University, Jeonju, Republic of Korea,
[**]University College, Yonsei University, Seoul, Republic of Korea,
[***†]Center for Advanced Image Information Technology, Jeonbuk National University, Jeonju, Republic of Korea

ABSTRACT

High-potential data can be predicted and stored in the cache to prevent cache misses, thus reducing the processor's request and wait times. As a result, the processor can work non-stop, hiding memory latency. By utilizing the temporal/spatial locality of memory access, the prefetcher introduced to improve the performance of these computers predicts the following memory address will be accessed. We propose a prefetcher that applies the GRU model, which is advantageous for handling time series data. Display the currently accessed address in binary and use it as training data to train the Gated Recurrent Unit model based on the difference (delta) between consecutive memory accesses. Finally, using a GRU model with learned memory access patterns, the proposed data prefetcher predicts the memory address to be accessed next. We have compared the model with the multi-layer perceptron, but our prefetcher showed better results than the Multi-Layer Perceptron.

Key Words : Gated Recurrent Unit, Prefetcher, Traces, Memory Hierarchy, Cache Management

1. 서 론

최근 인공지능, 빅데이터 분석등과 같이 대규모의 데이터 처리를 위한 애플리케이션의 요구가 증가함에 따라, 오늘날의 폰 노이만 (Von Neumann) 구조의 컴퓨터 시스템은 프로세서와 메모리의 성능의 격차로 인해 메모리 병목 현상이 심화되고 있다[1,2]. 이러한 프로세서-메모리 사이의 성능 격차의 문제는 전통적으로 메인 메모리와 프로세서 사이에 상대적으로 빠른 캐시 메모리를 계층적으로 두어 해결해왔다. 계층적 메모리 구조에서는 공간지역성 (spatial locality) 및 시간 지역성 (temporal locality)의 특성을 활용하여 프로세서와 메인 메모리 사이의 접근 지연을 최소화하지만 오늘날의 데이터 중심 애플리케이션 (data-centric application)을 처리하기에는 충분하지 않다. 프로세서-메모리 사이의 성능 격차문제를 해결하기 위해 적용할 수 있는 또다른 방법 중에 하나는 프리페치 (prefetch) 기법을 도입하는 것이다. 프리페치 기법은 메모리 접근 시간을 숨기고 사이클 당 명령어 처리 횟수 (IPC – instruction per cycle)을 개선할 수 있다. 프리페치는 기본적으로 곧 사용될 데이터를 예측해서 해당 데이터를 미리 인출해 오는 것으로, 단순히 현재 미스 (miss)가 발생한 데이터의 다음 주소의 데이터를 인출하는 방식부터, 히스토리 테이블

[†]E-mail: sk.yoon@jbnu.ac.kr

(history table)을 기반으로 과거의 메모리 액세스 패턴을 분석하여 가까운 미래의 액세스 패턴을 예측하는 방식 등 다양한 기법을 취할 수 있다[4,7]. 다양한 방식으로 예측된 데이터는 해당 데이터가 프로세서에 의해 요청되기 전 미리 데이터를 메모리로부터 인출하여 프로세서 내 캐시에 저장해 놓음으로써, 프로세서가 해당 데이터를 요청했을 때 캐시 적중률을 높이고 메인 메모리로의 접근에 대한 미스 패널티(miss penalty)를 줄일 수 있다.

그러나 처리해야 하는 데이터의 양이 증가하고, 데이터 중심 애플리케이션에 대한 수요가 증가함에 따라 워크로드들의 메모리 접근 패턴 (memory access pattern)은 점차 복잡해지고 불규칙한 패턴을 보이게 된다. 따라서 근미래에 사용될 데이터 주소를 예측하여 미리 가져오는 것이 점점 어려워지고 있다[3,5,6]. 이러한 문제를 해결하기 위해 기계학습을 프리페치 (prefetch) 기법에 적용하는 연구가 소개되고 있다.

최근 널리 사용되고 있는 기계 학습 (machine learning)인 공 신경망 (artificial neural networks) 메모리 접근 패턴을 정확하게 예측하여 프로세서로부터 곧 요청될 데이터를 예측하는데 탁월한 성능을 보여주고 있다[5,7]. 신경망은 구조화되지 않은 데이터를 이해하고 일반적인 관찰을 할 수 있다. 이 경우 프리페칭 문제를 패턴 예측 문제로 줄일 수 있다. 하지만 최근 워크로드의 메모리 접근 패턴의 복잡성으로 인해 신경망을 이용한 방법보다 강력한 학습 모델이 필요하다[7]. 선형 회귀 (linear regression), SVM (support vector machine), RNN (Recurrent Neural Network)및 GRU (Gated Recurrent Unit) 등의 모델은 예측에 있어 큰 잠재력을 보여준 기계 학습 및 딥 러닝 알고리즘들이다.

본 논문에서는 이러한 알고리즘 중 딥러닝 기반의 Gated Recurrent Unit (GRU) 알고리즘을 이용한 GRU 프리페치 기법을 제안하고, 그래프 프로세싱 워크로드들에 대한 성능 평가를 수행한다.

2. 관련 연구

기존에 제시된 프리페치 (prefetch) 기법 중 근 미래에 사용될 데이터의 예측은 메모리 접근 패턴 또는 이전에 액세스한 메모리 주소 값을 기반으로 한다. 베스트 오프셋 (Best Offset prefetcher) 프리페처[8]는 프리페치를 위해 메모리 주소의 오프셋 (prefetch offset) 뿐만 아니라 프리페치 적시성 (prefetch timeliness)까지 고려하여 사용될 가능성이 높은 메모리 주소를 예측한다. AMPM (Access Map Pattern Matching) 프리페처[7]는 메모리 접근 맵 (memory access map)을 기반으로 하여 액세스 패턴을 탐색하고[10], GHB (Global History Buffer) 프리페처[6] 및 VLDP 프리페처 (Variable Length Delta Prefetcher) [11]는 과거에 요청된 메모리 주소 패턴을 기반으로 프리페치 할 데이터를 결정한다. 베스트 오프셋 프리페처는 가장 빈번하게 보여지는 오프셋의 데이터를 가져오는 것을 목표로 하고 있으며, 지연 대기열 (delay queue)을 사용하여 각 오프셋에 대한 요청 정보를 기록하기 때문에 적시성 (timeliness)는 우수하다. 하지만 베스트 오프셋 프리페처는 빈도수가 가장 높은 오프셋 패턴만 가져오고 중요하지만 빈도수가 떨어지는 오프셋 패턴은 건너뛸 수 있기 때문에 정확성은 떨어지는 문제점을 가지고 있다.

스트라이드 기반 프리페치(stride-based prefetch) 기법은 각 메모리 주소를 통해 반복되는 스트라이드 또는 델타 (delta) 패턴을 사용하여 근 미래에 요청될 주소를 프리페치 한다[12-14]. AMPM과 SMS (Spatial Memory Streaming) [15]는 공간 지역성의 측면에서 반복적으로 사용되는 메모리 공간 영역에 초점을 맞춘 기법이다. 이러한 종류의 프리페처는 커버리지가 높기 때문에 좋은 성능을 보일 수 있지만, 프리페치의 적시성은 고려하지 않는다. 즉, 접근 순서에 대한 정보는 메모리 영역 내에 기록되지 않기 때문에, 초기에 많은 프리페치를 유발한다.

메모리 주소에 대한 상관 관계를 통해 요청될 데이터를 예측하는 GHB와 VLDP는 운영체제 페이지 내의 두 인접 접근 사이의 오프셋을 기록한다. GHB는 하나의 델타 히스토리 (delta history) 만을 기반으로 예측할 수 있지만, VLDP는 가변 델타 히스토리 길이를 기반으로 예측을 가능하게 하기 위해 계단식 델타 프리페치 테이블(DPT)을 사용한다. 또한 SPP(Signature Path Prefetcher) [16]는 VLDP 프리페처를 개선하였다. 그러나 주로 프리페치 깊이를 증가시켜 보다 공격적인 프리페치를 수행하게 함으로서 성능을 향상시킨다.

인공 신경망 (Artificial neural networks)은 정확한 패턴 예측에서 큰 잠재력을 보여주고 있으며, 이미지 처리, 오디오 처리, 자연어 처리 등 다양한 분야에서 큰 파급력을 보여주고 있다. RNN 증강 오프셋 프리페치 (RNN augmented offset prefetching) 기법은 RNN의 예측 능력을 사용하여 오프셋 프리페치에 대한 시간적 참조 주소를 생성하여 오프셋 프리페치가 현재 주소와 예측된 다음 주소 모두에서 공간적으로 작동할 수 있도록 한다 [5]. 신경망 기반 하드웨어 프리페처인 LSTM (Long Short-term Memory) 프리페처는 메모리 접근 주소들에 대한 차이를 기록하여 LSTM 학습을 진행하기 때문에 높은 정확도와 커버리지를 보여주지만, 정확한 프리페치를 수행하기까지 긴 대기 시간이 필요하다는 단점을 가지고 있다 [5].

3. GRU (Gated recurrent unit) 프리페처

GRU [17]는 인공 신경망 학습 시 발생할 수 있는 기울기 소실(gradient vanishing) 및 폭주 (exploding) 문제를 해결하기 위해 제안되었다[17,18]. GRU는 게이트 (gate) 기능을 통합한 순환 신경망으로, RNN에 비해 더 적은 매개 변수를 사용하고, 훈련시간이 빠르다는 장점이 있다. Fig. 1에서 보는 것과 같이GRU는 업데이트 게이트 (update gate)와 리셋 게이트 (reset gate)를 가진다. 업데이트 게이트는 얼마나 많은 정보(메모리)를 유지하고 전달해야 하는지 결정하고, 재설정 게이트에서는 잊어버릴 이전 정보의 양을 결정한다.

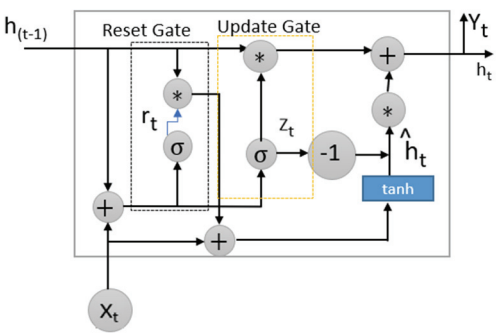

Fig. 1. Gate Recurrent Unit Cell.

다음의 식은 GRU의 전체 연산을 보여준다.

$$r_t = \sigma(w_r X_t + u_r h_{t-1} + b_r) \quad (1)$$

$$z_t = \sigma(w_z X_t + u_z h_{t-1} + b_z) \quad (2)$$

$$\widehat{ht} = \tanh(w_h X_t + u_h(r_t * h_{t-1}) + b_h) \quad (3)$$

$$Y_t = h_t = z_t * h_t + (1 - z_t) * h_{t-1} \quad (4)$$

여기서 r_t는 리셋 게이트 출력의 출력이고 z_t는 업데이트 게이트 (update gate)의 출력이다. X_t는 셀에 대한 입력, Y_t는 다른 셀에 $\widehat{ht}$형식으로 입력되는 출력, w와 u는 가중치 (weight), b는 편향값 (bias value)이다.

본 논문에서 제안하는 GRU 프리페처는 Fig.2에서 보는 것과 같이 3개의 히든 레이어 (hidden layer)를 사용한다. 첫 번째는 GRU 은닉 계층, 두 번째는 address binary hidden 계층이고, 세 번째는 이 두 계층의 연결이다.

GRU 프리페처는 Gated Recurrent Unit 모델을 기반으로 접근한 주소를 이용하여 델타 값을 예측한다. 이 값은 가장 높은 확률을 갖는 다음 델타 값을 예측하기 위해 GRU로 전달된다. 제안하는 프리페처는 라스트 레벨 캐시 (last level cache – LLC) 미스가 발생하여 메모리 액세스가 발생할 때마다 프리페치 요청을 하나씩 실행한다. GRU 프리페처 내의 델타 테이블에는 캐시 적중된 이전 접근 주소의 모든 델타 값이 저장되며, 이는 캐시 미스가 발생할 때까지 반복된다. 그리고 현재 액세스한 주소와 이전에 액세스한 주소를 사용하여 델타를 계산하고 이 델타 값을 사용하여 테이블을 검색한다. 캐시 적중된 이전 접근 주소의 델타 값들은 프리페처를 학습시키기 위해 다시 GRU 프리페처에 제공된다.

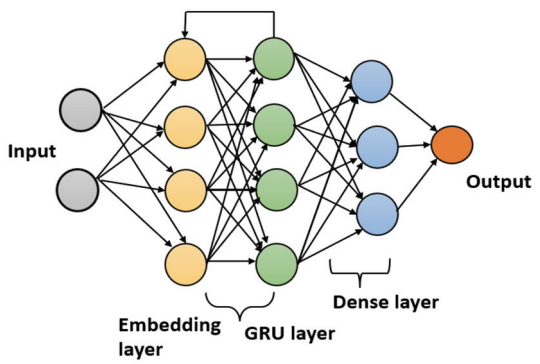

Fig. 2. GRU Dense Layers.

4. 실험 결과

본 논문에서 제안하는 GRU 프리페처의 성능을 평가하기 위해 본 논문에서는 GRU 프리페처 모델의 정확도 (accuracy)와 손실(loss)을 측정한다.

또한 성능 측정을 위해 사용한 메모리 트레이스는 그래프 프로세싱을 위한 벤치마크인 GraphBIG [19] 으로부터 triangle count (tc), degree centrality (dc), connected component (ccomp), pagerank (prank), k-core, breadth-first search (bfs), depth-first search (dfs), sssp (single-source shortest path) 워크로드를 추출하였고, GraphBIG 벤치마크 구동을 위한 데이터 셋 (data set) 으로 LDBC Graph [21]를 사용하였다. LDBC는 복잡하면서도 전체적인 그래프 연산을 을 위한 분석 프레임워크를 포함한다. 또한 이들 워크로드에 대한 메모리 접근 트레이스를 추출하기 위해 Intel PIN-tool 3.0 [20]을 사용하였다. Table 1은 본 논문에서 사용한 GraphBIG 워크로드에 대한 구체적인 설명이다.

본 연구에서 제안하는 GRU 프리페처는 다층 퍼셉트론 (MLP - multi-layer perceptron)을 이용한 프리페처 모델과 비교하여 성능을 측정하였다. MLP는 기본적으로 완전히 연결된 순방향 인공 신경망 클래스이다. 복수의 숨겨진 레이어가 있는 입력, 출력 레이어로 구성된다. Fig. 3에서 보는 것과 같이 제안하는 GRU 프리페처는 다층 퍼셉트론

(MLP) 모델 보다 평균적으로 8% 높은 정확도를 보인다. 또한 Fig. 4에서 보는 것과 같이 제안하는 모델의 손실(loss)이 비교 모델인 MLP 보다 작게 나타나는 것을 확인할 수 있다.

Table 1. Workloads description

Workload	Category
bfs	Graph traversal
dfs	Graph traversal
prank	Graph traversal
sssp	Graph traversal
kCore	Graph analytics
ccomp	Graph analytics
triangle count	Graph analytics
degree centrality	Social analysis

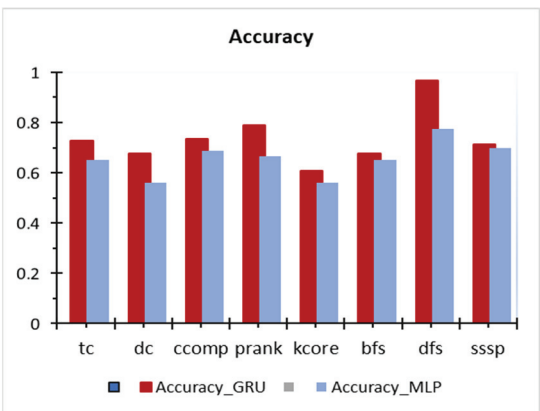

Fig. 3. Accuracy comparison between GRU and MLP.

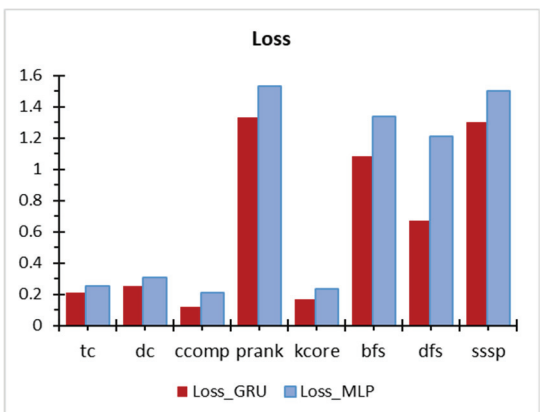

Fig. 4. Loss Comparison between GRU and MLP.

Table 2는 GRU 및 MLP에 대한 성능 메트릭 (performant metric)을 보여준다. Mean squared error (MSE), root mean squared error (RMSE), mean absolute percentage error (MAPE) 모두 MLP 프리페처에 비해 GRU 프리페처가 더 낮은 값을 달성한 것을 확인 할 수 있다.

Table 2. Performance Metrics

Performance Metrics	GRU	MLP
MSE	0.147	0.196
RMSE	0.281	0.350
MAPE	0.112	0.341

5. 결 론

본 연구는 불규칙 하고 복잡한 메모리 접근 패턴을 보이는 워크로드에 대해 메모리 액세스 패턴을 정확하게 예측하기 위한 GRU 프리페처를 제안한다. GRU는 긴 시퀀스 접근 패턴을 기억하는 데 우수한 성능을 보이는 것을 실험을 통해 확인하였다. 또한 본 논문은 GraphBIG 워크로드 메모리 트레이스를 사용하여 제안된 GRU 알고리즘을 학습하고 MAE, RMSE 및 MAPE 성능 지표를 비교하였다. 최적화된 델타 값을 계산한 다음 이러한 매개 변수를 사용하여 GRU를 학습하고 미래의 최적 추적 값을 예측한다.

이 논문에서 제안한 GRU 프리페처는 메모리 중심적 애플리케이션에서 복잡하고 불규칙한 메모리 접근을 예측하는 데 좋은 예측자 역할을 한다는 것을 보여준다. 본 논문에서 제시한 GRU 프리페처는 대용량의 메인 메모리를 구성 하는데 하나의 대안이 될수 있는 비휘발성 메모리 소자를 활용한 하이브리드 메인 메모리 환경에서 성능 개선을 위해 사용 될수 있다.

감사의 글

This research was supported by National Research Foundation of Korea (NRF) Grant funded by the Korean Government (Ministry of Science and ICT) NRF-2021R 1I1A3059832.

참고문헌

1. Wulf, Wm A., and Sally A. McKee. “Hitting the memory wall: Implications of the obvious.” *ACM SIGARCH computer architecture news* 23, no. 1 (1995): 20-24.

2. Chen, Yong, Surendra Byna, and Xian-He Sun. “Data access history cache and associated data prefetching mechanisms.” In *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, pp. 1-12. 2007.
3. LARSSON, MATTIAS. “Data Prefetcher Based on a Temporal Convolutional Network.” (2022).
4. Srivastava, Ajitesh, Angelos Lazaris, Benjamin Brooks, Rajgopal Kannan, and Viktor K. Prasanna. “Predicting memory accesses: the road to compact ml-driven prefetcher.” In *Proceedings of the International Symposium on Memory Systems*, pp. 461-470. 2019.
5. Zeng, Yuan, and Xiaochen Guo. “Long short term memory based hardware prefetcher: a case study.” In *Proceedings of the International Symposium on Memory Systems*, pp. 305-311. 2017.
6. Nesbit, Kyle J., and James E. Smith. “Data cache prefetching using a global history buffer.” In *10th International Symposium on High Performance Computer Architecture (HPCA'04)*, pp. 96-96. IEEE, 2004
7. Ishii, Yasuo, Mary Inaba, and Kei Hiraki. “Access map pattern matching for high performance data cache prefetch.” *Journal of Instruction-Level Parallelism* 13, no. 2011 (2011): 1-24.
8. Zhang, Pengmiao, Ajitesh Srivastava, Benjamin Brooks, Rajgopal Kannan, and Viktor K. Prasanna. “Raop: Recurrent neural network augmented offset prefetcher.” In *The International Symposium on Memory Systems*, pp. 352-362. 2020.
9. Michaud, Pierre. “Best-offset hardware prefetching.” In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 469-480. IEEE, 2016.
10. da Cruz, Eduardo Henrique Molina, Marco Antonio Zanata Alves, Alexandre Carissimi, Philippe Olivier Alexandre Navaux, Christiane Pousa Ribeiro, and Jean-François Méhaut. “Using memory access traces to map threads and data on hierarchical multi-core platforms.” In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, pp. 551-558. IEEE, 2011.
11. Shevgoor, Manjunath, Sahil Koladiya, Rajeev Balasubramonian, Chris Wilkerson, Seth H. Pugsley, and Zeshan Chishti. “Efficiently prefetching complex address patterns.” In *Proceedings of the 48th International Symposium on Microarchitecture*, pp. 141-152. 2015.
12. Dahlgren, Fredrik, and Per Stenstrom. “Effectiveness of hardware-based stride and sequential prefetching in shared-memory multiprocessors.” In *Proceedings of 1995 1st IEEE Symposium on High Performance Computer Architecture*, pp. 68-77. IEEE, 1995.
13. Ros, Alberto. “Berti: A per-page best-request-time delta prefetcher.” *The 3rd Data Prefetching Championship* (2019).
14. D. Joseph and D. Grunwald, “Prefetching using markov predictors,” IEEE Transactions on Computers, vol. 48, no. 2, pp. 121–133, 1999
15. Stephen Somogyi, Thomas F Wenisch, Anastassia Ailamaki, Babak Falsafi, and Andreas Moshovos. 2006. Spatial memory streaming. ACM SIGARCH Computer Architecture News 34, 2 (2006), 252–263
16. Bhatia, Eshan, Gino Chacon, Seth Pugsley, Elvira Teran, Paul V. Gratz, and Daniel A. Jiménez. “Perceptron-based prefetch filtering.” In *Proceedings of the 46th International Symposium on Computer Architecture*, pp. 1-13. 2019.
17. Dey, Rahul, and Fathi M. Salem. “Gate-variants of gated recurrent unit (GRU) neural networks.” In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pp. 1597-1600. IEEE, 2017.
18. Hochreiter, Sepp. “The vanishing gradient problem during learning recurrent neural nets and problem solutions.” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6, no. 02 (1998): 107-116.
19. Nai, Lifeng, et al. “GraphBIG: understanding graph computing in the context of industrial solutions.” *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2015.
20. Luk, Chi-Keung, et al. “Pin: building customized program analysis tools with dynamic instrumentation.” Acm sigplan notices. Vol. 40. No. 6. ACM, 2005.
21. A. Iosup, Alexandru, et al. “Ldbc graphalytics: A benchmark for large-scale graph analysis on parallel and distributed platforms.” Proceedings of the VLDB Endowment 9.13 (2016): 1317-1328.

접수일: 2023년 3월 21일, 심사일: 2023년 3월 31일, 게재확정일: 2023년 4월 18일