Amphi: Practical and Intelligent Data Prefetching for the First-Level Cache

Xuan Tang*, Zicong Wang*, Shuiyi He, Dezun Dong†, Xiangke Liao
College of Computer Science and Technology
National University of Defense Technology
Hunan, China
{tangxuan19, wangzicong, heshuiyi24, dong, xkliao}@nudt.edu.cn

***Abstract*—Data prefetchers play a crucial role in alleviating the memory wall by predicting future memory accesses. First-level cache prefetchers can observe all memory instructions but often rely on simpler strategies due to limited resources. While emerging machine learning-based approaches cover more memory access patterns, they typically require higher computational and storage resources and are usually deployed in the last-level cache. Other intelligent solutions for the first-level cache show only modest performance gains. To address this, we propose Amphi, the first practical and intelligent data prefetcher specifically designed for the first-level cache. Applying a binarized temporal convolutional network, Amphi significantly reduces storage overhead while maintaining performance comparable to the SOTA intelligent prefetcher. With a storage overhead of only 3.4 KB, Amphi requires only one-eighth of Pythia's storage needs. Amphi paves the way for the broader adoption of intelligence-driven prefetching solutions.**

***Index Terms*—data prefetcher, first-level cache, machine learning**

I. Introduction

Hardware prefetching is a crucial technique for hiding long memory access latency. Among diverse levels of prefetching technologies, the data prefetching of the first-level data cache (L1D) presents substantial advantages on account of its proximity to the core and higher speed, thereby leading to significant performance.

Currently, there are two main categories of solutions for L1D prefetching: traditional methods and intelligent methods. Traditional prefetchers can easily capture regular and simple memory access patterns. These methods typically utilize tables to learn memory access patterns, which limits the learning capability due to constraints imposed by the size and representational ability of the tables. Intelligent methods often serve as auxiliary components in L1D to enhance performance. Some use perceptron to design a prefetch filter [4], and some create off-chip predictors to improve the overall performance [2]. Generally, intelligent prefetchers are often deployed in the last level cache (LLC). Although intelligent prefetchers can learn more complex or irregular memory access patterns compared to traditional prefetchers and achieve higher prediction accuracy, they are not deployed in L1D due to their significant storage overhead and resource consumption.

*Both authors contributed equally to this work.
†Corresponding author.

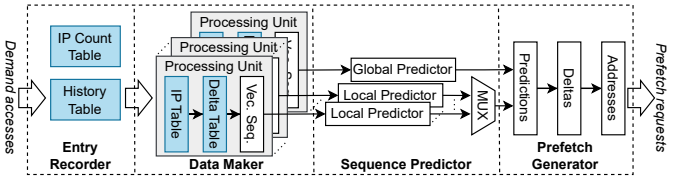

Fig. 1: Overview architecture of Amphi

We are committed to changing this situation by seeking an intelligent prefetcher that can be deployed in L1D. To further explore this, we have two key inspirations.

Inspiration 1: Data prefetching primarily revolves around the sequence of memory access patterns, which can be effectively optimized by sequence modeling analysis. Temporal convolutional network (TCN) is a typical sequence model, which can effectively capture long-range dependencies via dilated convolutions, and maintain stability in gradient propagation. The straightforward architecture and fewer hyperparameters also simplify design and tuning. Therefore, we choose TCN as the base model.

Inspiration 2: The introduction of binary neural network (BNN) has further advanced the field of model quantization, presenting new opportunities for intelligent prefetchers in L1D [5]. By quantizing the weights and activations of the model to binary, we can achieve substantial compression while maintaining performance.

To the best of our knowledge, **Amphi is introduced as the first practical and intelligent data prefetcher for the first-level cache**. Amphi integrates BNN with TCN to create a lightweight prefetcher, effectively balancing overhead and performance. We make the following contributions: 1) We propose Amphi, the first intelligent data prefetcher that can be deployed in the first-level cache. 2) Amphi is the first work to explore the application of BNN in data prefetching. 3) The evaluation demonstrates that Amphi outperforms the baseline by 23.79% in IPC with only 3.4 KB storage overhead.

II. Methodology

Figure 1 describes the overview architecture of Amphi, which mainly consists of four components: 1) Entry Recorder; 2) Data Maker; 3) Sequence Predictor; 4) Prefetch Generator.

Entry Recorder: The Entry Recorder consists of two tables: History Table (HT, 512 entries), and IP Count Table (IPCT, 16

978-3-9826741-0-0/DATE25/© 2025 EDAA

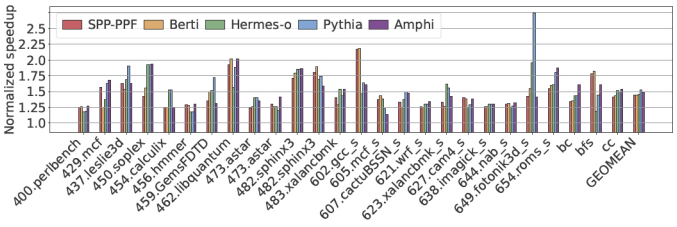

Fig. 2: Single core performance (IPC)

entries). The HT logs all memory access records, storing the current access instruction pointer (IP) and address. For each new demand access, the Entry Recorder creates and inserts a new entry to the HT. The IPCT records the frequency of occurrences for each IP as entries are incorporated into the HT for subsequent referencing.

Data Maker: The Data Maker preprocesses the data from the HT for the Sequence Predictor. The process begins by fetching the top n most frequent IPs from the IPCT. Each IP is associated with a Processing Unit responsible for handling memory accesses. Memory accesses for these IPs are extracted into the IP Table (16 entries). The entries in the IP Table are then sent for differential operations, and the results are stored in the Delta Table (15 entries). Finally, Data Maker hashes these entries into 8-bit vector sequences as the local dataset. Additionally, the Data Maker conducts the same operations on the global memory access stream to generate a global dataset.

Sequence Predictor: Amphi applies binarized TCN as the core predictor. The output of Data Maker is represented with binary vectors, and the weights of the models are also set to binary values. Amphi employs two types of models: local predictor and global predictor, to accommodate local and global datasets, respectively. In Amphi's design, we implement n local predictors and one global predictor.

Prefetch Generator: The Prefetch Generator takes the predictions and translates them into deltas[1]. Using the current initial address, it computes the prefetch addresses and subsequently issues the prefetch requests. Importantly, if the model is not yet trained, prefetching is performed using the stride prefetcher to fill in the gaps when the model's training is insufficient.

III. Evaluation and Conclusion

We apply Amphi in the ChampSim Simulator [6]. The SPEC06, SPEC17 [8], and GAP [1] benchmark suites are chosen for evaluation. We compare Amphi with Berti [7], Pythia [3], Hermes [2], SPP-PPF [4]. Berti is a traditional L1D SOTA prefetcher, and Pythia is an intelligent prefetcher in LLC. Hermes is an off-chip predictor with a perceptron in L1D, and SPP-PPF applies a perceptron filter in L1D.

A. Experimental results

Figure 2 compares the IPC achieved by various prefetchers. Amphi outperforms the baseline (IP-Stride) by 23.79% and surpasses Berti by 4.31%. Among intelligent methods, Amphi exceeds Hermes by 3.19% and SPP-PPF by 4.29%, while trailing Pythia by 3.68%. Amphi achieves the best performance in 14 of 26 applications. For example, it improves performance by 43% in $429.mcf$, outperforming all designs, and more than doubles the performance gains in bc compared to others.

[1]Delta is defined as the difference between two adjacent addresses.

B. Overhead analysis

The storage overheads of different prefetchers are shown in Table I. By leveraging the extreme compression of binarized networks, Amphi's storage overhead is only 3.4 KB, which is comparable to that of the traditional SOTA prefetcher Berti, while being significantly lower than other intelligent methods such as Pythia and SPP-PPF. As a prefetcher deployed in L1D, Amphi achieves performance on par with other prefetchers and even outperforms them in certain workloads, while maintaining a design characterized by less storage requirements.

Prefetcher	Storage overhead
Amphi	3.4 KB
Berti [7]	2.5 KB
Pythia [3]	25.5 KB
Hermes [2]	4 KB
SPP-PPF [4]	39.3 KB

TABLE I: Storage overhead comparison

C. Conclusion

In conclusion, this paper introduces Amphi, the first practical and intelligent prefetcher for the first-level cache, which is integrated with binarized TCN models. Amphi achieves lower overhead and better performance, paving the way for the broader adoption of intelligence-driven prefetching solutions.

IV. Acknowledgment

This work was partially supported by the National Key Research and Development Program of China under Grant No.2022YFB4501702 and the National University of Defense Technology Research Project under Grant ZK22-23.

References

[1] S. Beamer *et al.*, "The GAP benchmark suite," *CoRR*, vol. abs/1508.03619, 2015.

[2] R. Bera, *et al.*, "Hermes: Accelerating long-latency load requests via perceptron-based off-chip load prediction," in *MICRO*, 2022, pp. 1–18.

[3] R. Bera *et al.*, "Pythia: A customizable hardware prefetching framework using online reinforcement learning," in *MICRO*, 2021, pp. 1121–1137.

[4] E. Bhatia *et al.*, "Perceptron-based prefetch filtering," in *ISCA*, 2019, pp. 1–13.

[5] M. Courbariaux *et al.*, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," *arXiv*, vol. abs/1602.02830, 2016.

[6] N. Gober *et al.*, "The championship simulator: Architectural simulation for education and competition," 2022.

[7] A. Navarro-Torres *et al.*, "Berti: an accurate local-delta data prefetcher," in *MICRO*, 2022, pp. 975–991.

[8] S.P.E.Corporation, "Standard performance evaluation corporation," 2023, https://www.spec.org/benchmarks.html.