1 Summary

A Hierarchical Neural Model of Data Prefetching (2021): The authors in this paper introduce Voyager, a neural network model trained for data prefetching. Their neural model has less overhead than previous models; it is 110-200× more compact in terms of storage, and 15-20× less computationally expensive. In their approach, they re-frame the data prefetching problem as a probabilistic prediction problem, and the outputs (address prefetches) are represented as probability distributions, rather than prefetching from grounded, labeled truths. Some of the unique benefits of their design, compared to previous prefetchers, is that Voyager can learn both address correlations (temporal prefetching) and delta patterns (strides) because of the hierarchical structure of the implemented embedding layers. The techniques they introduce to their model address some of the common problems found in using data prefetchers. The class explosion problem in data prefetching (having to predict addresses from among 2^{64} unique addresses) is dealt with by decomposing memory address prediction into two subproblems (embedding layers) in their hierarchical model, called page prediction and page-aware offset prediction. For compulsory misses for addresses requested at low frequency, the model is able to learn the delta patterns to prefetch them, rather than spend the work learning address correlations. And Voyager addresses the labeling problem by predicting the most predictable addresses when given multiple possible labels using the authors' multi-label training scheme. The authors show comparisons against other data prefetching neural models, as well as hardware prefetchers; while they conclude that their neural network, like previous models, is still too computationally expensive to be in hardware prefetchers, Voyager is a significant contribution towards the development of more practical prefetchers.

- Voyager is a neural network for data prefetching that can learn delta correlations and address correlations, i.e. this one can perform temporal prefetching, which was not previously done in other models, as rule-based prefetchers are limited by pre-determined strides and fixed methods of correlating addressed, and other ML-based prefetchers do not perform well for irregular data prefetches or lacked sufficient measurements that consider practical accuracy and timeliness.
- Class explosion problem (too large input and output space) addressed by decomposing address prediction into 1) page prediction and 2) offset prediction, and using an attention-based embedding layer so that pade prediction can provide context for offset prediction.
- Labeling problem addressed by using multi-label training scheme for model to learn from multiple possible labels and it learns the most predictable label.
- Neural models not yet practical for use in hardware data prefetchers.
- Their paper is first to 1) show IPC benefit of LSTM prefetching, 2) show a meural model that combines both delta patterns and address correlations, and 3) their multi-labeling scheme provides a richer set of labels, 4) their model is more compact and less computationally expensive than prior neural solutions.

- They formulate the data prefetching problem as a probabilistic prediction problem, and the output as a probability distribution.
-

2 Strengths

- The paper picks a good variety of other baseline prefetchers to compare Voyager against, as well as discussing the metrics or parameters that are specifically relevant to data prefetching.
- Testing their model on cases where previous models performed inadequately or were never used at all, i.e. irregular memory accessing benchmarks from SPEC06 and GAP, and on enterprise-scale applications of Google search and ads. The results of the later were limited to the memory traces of production Google servers, so they couldn't simulate IPC on ChampSim, just accuracy and coverage, for which they used a new unified definition of both (prediction is correct only when the model corretly predicts the next load address, i.e percent of addresses that are predicted to be prefetched; they use this unified definition for other benchmarks too). They did not include how the other prefetchers performed with the enterprise-scaled applications compared to Voyager.
- They compared Voyager against a variety of (idealized baseline) prefetchers: spatial prefecter the Best Offset (BO) Prefetcher, temporal prefetchers (STMS, ISB, Domino), impractical neural prefetchers (Delta-LSTM). And compared the metrics to those of a non-prefetching system on accuracy, coverage, and IPC.
- Computational cost are compared against those of a temporal prefetcher and a non-hierarchical neural network implementation.
- they compare storage and computational efficiency against Delta-LSTM neural prefetcher and ISB, what about BO?

3 Weaknesses

- Certain metrics and applications the authors selected for analyzing Voyager's performance against baseline prefetchers appear biased, and much of the results discussion focused more on comparing Voyager to one other (arbitrarily) picked baseline prefetcher (ISB) without explanation, despite earlier mentions of 4 other baseline prefetchers.
- The figures weren't very well captioned or annotated.
- with their new unified definition of accuracy and coverage (in order to include measurements with Google search and ads applications), they should have expanded or quantified the definition more, in addition to their justification for the unified definition.

- They only compared the prefetching degree variance of Voyager against those of ISB (address correlations) and ISB+BO (address correlations + compulsory misses) (Fig 9). How they implemented the hybrid is unclear, but it may be obvious. These were also arbitrarily picked from their categories, while if they had others, it should be shown, since in the temporal prefetchers (ISB, STMS, Domino), they all perform clearly differently on the workloads for accuracy, coverage, and IPC.

- they used a crippled Voyager w/o deltas in order to test memory access patterns (Sec. 5.3) over ISB, in order to isolate the benefits of Voyagers temporal access patterns. It is clear ISB is better than the other temporal prefetchers for most of the workloads in terms of accuracy, coverage, and IPC. But they didn't show a figure for where the 19.4% better coverage than ISB came from (which benchmarks, and what about Accuracy and IPC)

- They did not test spatial patterns agains the spatial prefetcher (BO) but again against the temporal prefetcher (ISB). If it was not worth noting or if ISB was better than the other prefetchers overall, they did not mention. Even if figure 5-8 generally showed ISB was better than all the others, they did not explain why or how.

- Fig 15 comparison of individuak labeling schemes against the multi-labeling scheme supports their hypothesis well that Voyager's scheme using probability distribution works better. But they only briefly discussed how different benchmarks preferred certain labeling schemes, but did not show results for those. If they referenced past work for this justificatoin, then okay. I would have liked to see individual benchmarks run with their preferred scheme, agains Voyager's hybrid scheme, in addition to the aggregate data in Fig 15. .

4 Rating: 4

5 Comments

This work yields a model that not only performs as well as, if not better than, previous models for irregular access patterns, but also works with more reduced overhead than any previous model. While the paper details a design that sensibly addresses problems of class explosion, offset aliasing, and labeling, the way in which they present overall performance compared to the 5 other baseline prefetchers make their experiments seem biased. For example, to compare performance in terms of accuracy, coverage, and IPC, the paper only picks specific sets of irregular benchmarks from SPEC06 and GAP; the paper did not include other applications where the other baseline prefetchers' features are tuned to or are known to have previously performed well. Second, when discussing how Voyager performed when varying the prefetching degree (Fig. 9), the paper only compares it to ISB and a hybrid ISB+BO; the authors are not clear why the temporal prefetcher, ISB, was selected, unless the audience should know or assume ISB was the "best" of the 5 baseline prefetchers. Likewise, a similar preference for only comparing Voyager to ISB is seen when they discuss performance on spatial/non-spatial patterns, but no explanation is given as to why BO, STMS, Domino, and Delta-LSTM are excluded. And finally, the results in Figure 15, which compares performance (given by their own unified definition of accuracy and coverage) of individual labeling schemes to Voyager's hybrid multi-labeling scheme, they do not clarify how they arrive at those aggregated numbers; moreover, it is mentioned that different benchmarks may have their preferred labeling schemes, but the paper did not show or discuss how the other models performed on benchmarks using these preferred schemes versus the hybrid scheme. Other than the way certain results are presented, their methods do a good job of isolating problems in the area of neural data prefetching seen in previous works and systematically addressing them in specific components of their model. The authors' Voyager model is a new and exciting contribution to data prefetching neural networks, even if it remains impractical.

6 Notes

- Voyager (2021):
  - a new neural network for data prefetching. A practical neural prefetcher. Probabilistic model of data prefetching.
  - can learn **address correlations**/temporal prefetching (for prefetching irregular sequences of memory accesses). It can also accomodate **delta correlations**/patterns (strides).
  - has a hierarchical structure separating addresses into pages and offsets, which introduces mechanisms for learning relations among pages and offsets. The hierarchical treatment of data addresses helps accomodate address correlations.
  - SPEC 2006 and GAP benchmark suites (irregular SPEC and graph): 41.6% IPC improvement over system with no prefetching, 21.7% IPC improvement over Domino prefetcher, 28.2% IPC improvement over ISB prefetcher. It has 79.6% accuracy/coverage of the benchmarks.
  - lower overhead: 15-20x reduced computation (training and prediction) cost, 110-20x reduced storage overhead (model size, Voyager is also smaller than non-neural temporal prefetchers); normal neural models not in hardware due to slow training and prediction.
  - They also showed the prefetching results on Google search and Google ads, Voyager achieves 37.8% and 57.5% accuracy/coverage, respectively.
- Voyager Design
  - Learn temporal correlations: hierarchical neural structure with one part of the model predicting pages addresses and the other predicting offsets within a page. Number of unique addresses to predict range from hundreds of thousands to tens of millions, much more than number of categoris is traditional ML tasks. The number of pages (rather than address) is in the tens of thousands and is more manageable. Naively predicting pages and offsets independently leads to offset aliasing.

    * page-aware offset embedding: uses a mixture of expert and a core mechanism called attention-layer embedding. Given a query (page embedding), the layer computes the page's correlation with each key (offset embedding), and produces a probability vector that represents this correlation.
    * The embedding layer is the primary storage and computation bottleneck for networks with large nuber of classes. So Voyager's page-aware offset embedding layer reduces its size and the number of parameters to learn. This simplifies model and reduces training overhead.
  - To deal with compulsory misses, uses vocabulary (words as inputs in the model that can product output) that includes addresses and deltas.
    * addresses of low frquency are prepresented using deltas of their page and offset from the previous page and offset
    * model does not learn address correlations for infrequent addresses, and learns some delta correlations instead to prefetch compulsory misses.
  - it is trained to predict the most predictable addresses from multiple possible labels, using a multi-label training scheme.
    * different labeling schemes work well for different workloads.
    * each training sample is associated with a set of labels, instead of a single output label
    * Voyager candidate labels: global, PC, basic block, spatial, and co-occurrence. So at an address, multiple future addresses are correlated with the current address through these five different labeling schemes; each are considered potential outputs. Candidate with highest probability is chosen, and probability distribution calculated with a binary cross entropy (BCE) loss function.
  - It was trained online in order to emulate hardware prefetchers.

- Voyager Evaluation against practical prefetchers and neural prefetchers.
  -

- Neural Prefetchers:
  - they still have a lot of computational cost that makes them impractical for hardware
  - authors found long data address histories is a good feature to predict irregular accesses.
  - multiple localizers benefit some hard to predict benchmarks.
  - These insights are meant to to guide development of practical prefetchers.

- Data prefetching problems:
  - class explosion problem: data prefetching has enormous inputs and output spaces, i.e. for 64-bit address address space, a model needs to predict from among 2^{64} unique address values.

    * authors address Class explosion problem addressed by decomposing address prediction into 1) page prediction (space is 10s-100s thousands) and 2) offset prediction (space is 64).
    * offset aliasing problem: addresses with same offset will share same offset embedding (internal representation of input features in a neural network, learned during training such that features that behave similarly have same embeddings.), leading to poor performance in neural networks. The authors use a new attention-based embedding layer that allows page prediction to provide context for offset prediction.
  - labeling problem: data prefetchers have no known ground truth tables from which to learn. Its not clear which label to use to train the ML model.
    * branch predictors can be trained by the ground truth answers as revealed by program's execution.
    * cache replacement policies can be trained by learning from Belady's provably optimal MIN policy.
    * to address labeling problem, authors use a new form of localization built into Voyager, a multi-label training scheme, enabling model to learn from multiple possible labels. (no single ground truth table, but model learns the most predictable label)

- Temporal prefetching
  - exploits correlations between consecutive addresses to predict the next address.
  - classification problem where each address is a class, and learning task is to learn the probability that an address will be accessed given a history of past events.
  - Historical events are input features. Future events are output labels.
- Stride Prefetching
  - they are also like probabilistic framework by incorporating strides of deltas in the formulation.
  - They look at history of strides and make a prediction for the next stride for the prefetch.

References

[1] Zhan Shi, Akanksha Jain, Kevin Swersky, Milad Hashemi, Parthasarathy Ranganathan, and Calvin Lin. 2021. A hierarchical neural model of data prefetching. Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. Association for Computing Machinery, New York, NY, USA, 861-873. DOI:https://doi.org/10.1145/3445814.3446752