Intelligent Hybrid Memory Scheduling Based on Page Pattern Recognition

Yanjie Zhen*, Weining Chen*, Wei Gao*, Ju Ren*, Kang Chen* and Yu Chen†*‡
*Department of Computer Science and Technology, Tsinghua University, Beijing, China
† Quan Cheng Laboratory, Jinan, China
Email: {zhenyj17, cwn20, gaow17}@mails.tsinghua.edu.cn, {renju, chenkang, yuchen}@tsinghua.edu.cn

***Abstract*—Hybrid memory systems exhibit disparities in their heterogeneous memory components' access speeds. Dynamic page scheduling to ensure memory access predominantly occurs in the faster memory components is essential for optimizing the performance of hybrid memory systems. Recent works attempt to optimize page scheduling by predicting their hotness using neural network models. However, they face two crucial challenges: the page explosion problem and the new pages problem. We propose an intelligent hybrid memory scheduler driven by page pattern recognition to address these two challenges. Experimental results demonstrate that our approach outperforms state-of-the-art intelligent schedulers regarding effectiveness and cost.**

I. Introduction

Many recent studies [1]–[3], have explored hybrid memory architectures that integrate memory components of varying access speeds and capacities by introducing new memory hardware or employing memory disaggregation techniques. Dynamic page scheduling across diverse memory components is crucial for optimizing the performance of hybrid memory systems. Traditional schedulers, such as the history scheduler, are designed based on fixed rules and thus perform poorly with applications exhibiting irregular memory access patterns [1]. To enhance the effectiveness of schedulers, some works [2], [3] leverage neural models to learn irregular memory access patterns and predict page hotness. However, they face two significant challenges arising from the complexity of real-world applications.

One challenge is the page explosion problem. Applications running on hybrid memory systems often possess large memory footprints, with the number of accessed pages potentially reaching several hundred thousand. The cost of learning and inference of such a vast number of pages is unacceptable, particularly in the time-sensitive and resource-limited hybrid memory management context. To address this problem, some existing works [2], [3] employ neural models to predict the hotness of a selected critical pages subset, which inevitably leads to a sacrifice in effectiveness.

The other challenge is the new page problem. The training and test sets are divided based on time when employing neural models for prediction. Models trained on data from an earlier time interval are used to predict the hotness of pages in a subsequent time interval. However, applications access different memory regions during execution, resulting in the test set encountering new pages not present in the training set. The model does not learn the access patterns of these new pages. Whether these new pages can be predicted using the trained model and how to make such predictions remain unexplored.

This paper proposes an intelligent hybrid memory scheduler based on page pattern recognition to address these challenges.

II. Solution

Memory access analysis. We analyze application's memory access patterns to address the above two challenges.

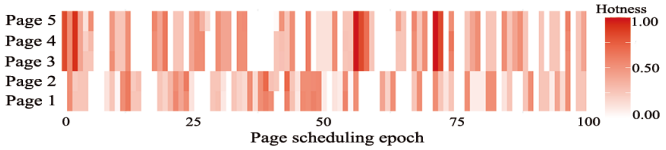

Fig. 1. Hotness heatmap of five sample pages from benchmark *pennant* [4].

Observation 1: Many pages have similar access patterns. Fig 1 visualizes the hotness heatmap of five sample pages from benchmark *Pennant*. Specifically, *Page1* and *Page2* exhibit a similar access pattern with consistent hotness throughout most epochs. *Page3*, *Page4* and *Page5* also exhibit a similar access pattern. Meanwhile, the access patterns of the page groups {*Page1, Page2*} and {*Page3, Page4, Page5*} are significantly different. Pages with similar access patterns tend to exhibit similar migration behavior, meaning they are either migrated simultaneously or not at all. This observation offers valuable insights for addressing the page explosion problem, suggesting that the focus could be learning the common patterns among pages.

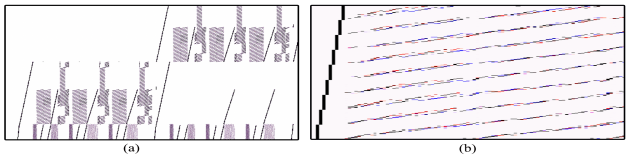

Fig. 2. The heatmap of benchmark *cactuBBSN* and *pennant*'s memory accesses.

Observation 2: The access patterns of most new pages generally exhibit high similarity to pages already included in the training set. Existing neural models in hybrid memory schedulers typically employ online or offline training methods. When using online training, they partition the program's execution into multiple time intervals, where the model is continually being trained in a time interval for use in the next time interval. As shown in Fig 2(a), the memory access region of the benchmark *cactuBBSN* changes over time, resulting in the emergence of new pages. For offline training, data

‡Corresponding author.
This work was supported by the research project funded by Quan Cheng Laboratory (Grant No. QCLZD202305).

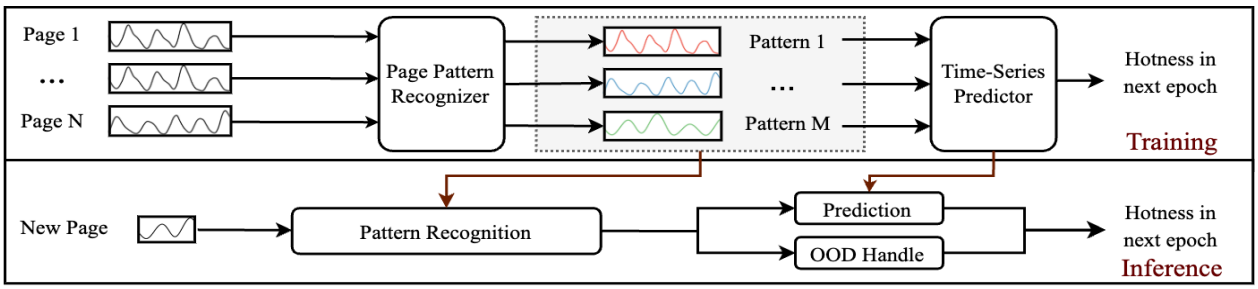

Fig. 3. The neural architecture.

generated during an application's offline execution is utilized for training, and the trained model is used to make predictions when the application is re-run online. However, due to the dynamic mapping of virtual memory and address space layout randomization, the address area accessed by an application is different each time it is executed. Fig 2(b) shows the heatmap of memory access patterns during two separate runs of the benchmark *pennant*, employed for simulating offline training. To better compare the patterns, we align them by shifting, ignoring discrepancies in page numbers. Both Fig 2(a) and Fig 2(b) demonstrate that the access patterns of new pages exhibit a high degree of similarity to those of pages already present in the training set. This observation provides valuable insights into addressing the new page problem, suggesting that new pages can be categorized into previously learned patterns during training.

A Pattern-Driven Scheduler. Based on our analysis of memory access patterns across various applications, we present a page pattern-driven hybrid memory scheduler using neural models. The neural architecture is shown in Fig 3.

During the training stage, we employ clustering based on the Gaussian Mixture Model(GMM) to identify pages with similar access patterns and extract their common patterns. GMM employs multiple Gaussian distributions to generate the joint probability of data samples. In our work, each distribution in GMM represents one access pattern. Given that these similar pages exhibit similar migration behaviors, we manage them as a page group. We employ an LSTM-based time series predictor to learn the access patterns of page groups. During the inference stage, we utilize the page pattern recognizer derived from the training set to identify the patterns of new pages and categorize them into previously learned patterns. The historical hotness of the corresponding page group is then fed into a time series predictor to predict the group's hotness for the upcoming epoch.

However, we face the challenge of inconsistent feature vector lengths when identifying patterns for new pages. The training dataset comprises long hotness sequences that span the epochs of the training set. Obtaining such long hotness sequences during the testing phase is unfeasible. This limitation is inherent to the real-time demands of our prediction task, restricting us to only short-term hotness data for new pages. Fortunately, our experiments found that pages with similar access patterns during a certain period are likely to exhibit similar patterns in other periods. This observation suggests effective pattern recognition can be conducted using only the first 50 epochs. In light of this, we introduce a warm-up phase of 50 epochs for each page during inference. During this phase, new pages are not used for hotness prediction but solely for pattern recognition.

III. Evaluation

We evaluate our work using an open-source lightweight hybrid memory simulator from [2]. We focus on 10 benchmarks from Rodinia [4], CORAL-2 [5], and SPECCPU2017 [6]. Our work is compared against both the state-of-the-art rule-based scheduler(the history scheduler) [1] and advanced neural-model-based schedulers(Kleio [2], Coeus [3]). We use fast memory hit rate to evaluate the effectiveness of different schedulers. Our experiments show that our work significantly outperforms previous approaches. On average, the fast memory hit rate of the history scheduler is 33.75%. Kleio and Coeus improve it to 38.00% and 41.74%, respectively, while our work enhance it to 62.15%(online training) and 61.89%(offline training). We also validate the capability of the page pattern recognizer in identifying new pages. Our experiments show that when using online training, the proportion of new pages can reach up to 27.54%. For offline training, all pages are considered new. Regardless of training mode, 99.50% of new pages can be assigned to existing patterns.

IV. Conclusion

This paper proposes an intelligent hybrid memory scheduler driven by page pattern recognition, addressing the page explosion and new page challenges. Experimental results show that it significantly outperforms previous approaches.

References

[1] M. R. Meswani, S. Blagodurov, D. Roberts, J. Slice, M. Ignatowski, and G. H. Loh, "Heterogeneous memory architectures: A hw/sw approach for mixing die-stacked and off-package memories," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2015, pp. 126–136.

[2] T. D. Doudali, S. Blagodurov, A. Vishnu, S. Gurumurthi, and A. Gavrilovska, "Kleio: A hybrid memory page scheduler with machine intelligence," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, 2019, pp. 37–48.

[3] T. D. Doudali and A. Gavrilovska, "Coeus: Clustering (a) like patterns for practical machine intelligent hybrid memory management," in *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 2022, pp. 615–624.

[4] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *2009 IEEE international symposium on workload characterization (IISWC)*. Ieee, 2009, pp. 44–54.

[5] "Coral benchmark codes," https://asc.llnl.gov/CORAL-benchmarks/, 2018.

[6] J. Bucek, K.-D. Lange, and J. v. Kistowski, "Spec cpu2017: Next-generation compute benchmark," in *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, 2018, pp. 41–42.