EasyChair Preprint

№ 13975

UCMoH: a Unified Learning-Based Cost Model for Tensorized Program Tuning in Heterogeneous Acceleration Clusters

Zihan Wang, Lei Gong, Wenqi Lou, Chao Wang and Xuehai Zhou

July 15, 2024

UCMoH: A Unified Learning-Based Cost Model for Tensorized Program Tuning in Heterogeneous Acceleration Clusters

Zihan Wang, Lei Gong✉, Wenqi Lou, Chao Wang, and Xuehai Zhou

University of Science and Technology of China
leigong0203@ustc.edu.cn

Abstract. Tensorized programs use various hardware intrinsics on heterogeneous accelerators to improve tensor computation performance. The wave of hardware customization introduces massive hardware accelerators and intrinsics, prompting deep learning compilers(DLCs) to explore tensorized program tuning to effectively leverage these hardware intrinsics. At the core of program tuning relies the design of the cost model, but currently there is still a lack of cost models specifically designed for tensorized programs, which severely hampers the co-optimization of DLCs and heterogeneous accelerators. To the best of our knowledge, we propose the first unified cost model for tensorized program tuning by introducing a unified feature representation and unified transfer prediction strategy. To meet training and testing requirements, we constructed a dataset dedicated to tensorized program tuning. UCMoH significantly improves adaptability to diverse execution environments and enables flexible transfer prediction while ensuring high accuracy. We will apply UCMoH to the tuning framework in the heterogeneous acceleration cluster.

Keywords: Cost Model· Tensorized Program Tuning· Lifelong Learning · Heterogeneous Acceleration Cluster.

1 Introduction

With the rise of AI algorithms such as deep learning and large models, customized hardware accelerators like *x86 AVX512/VNNI*, *ARM NEON/SDOT*, *NVIDIA Tensor Core*, *Cambricon MLU*, *Ascend Cube*, and *TPU* have proven to be essential means to address performance issues in related applications[3, 4]. Correspondingly, these hardware backends offer intrinsics at the software programming level to invoke underlying customized units for accelerating tensor computation. **The process of transforming and scheduling the original tensor program using these hardware intrinsics is called tensorization**.

To effectively leverage these hardware intrinsics, DLCs begin to explore tensorized program tuning[1]. The tuning process typically takes place in heterogeneous acceleration clusters. As shown in Fig. 1, DLC adopts a tensorization-aware scheduling method to generate the tensorized program space. First, the partition step divides the complete workload onto the most suitable acceleration platform. And then, the tensorization step matches the sub-workload to

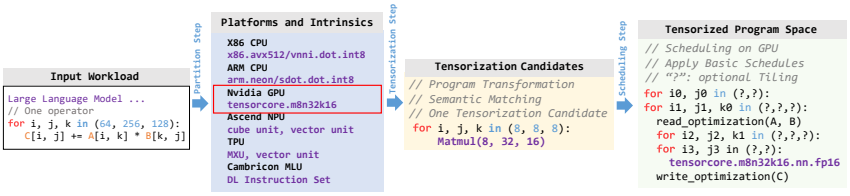

Fig. 1. Tensorized program space generation.

various intrinsic semantics on the acceleration platform. Finally, the scheduling step varies how intrinsics are invoked by applying basic schedule, e.g., loop tiling, reorder. **The above three steps generate a tensorized program space with hardware platform diversity, intrinsic diversity, and rich schedules.** DLCs need to search for the best performing program in this space. However, obtaining the actual latency of programs involves unacceptable time costs. **To address this issue, many DLCs resort to the cost model – using the predicted latency from the cost model as the criterion**[7]. Therefore, as the core of program tuning, the design of the cost model is crucial for searching for the optimal program.

There are currently many cost models for tensor programs[7], but it is difficult to effectively apply a cost model for tensor programs to accurately predict the performance of tensorized programs. First, while normal tensor programs run on general arithmetic processing units (ALU on CPUs, and CUDA Core on GPUs), tensorized programs mainly rely on customized acceleration components. Thus, the configuration and invocation of hardware intrinsics become the main factor affecting tensorized program performance. Second, tensorized programs are affected by the diversity of hardware platforms and hardware intrinsics in heterogeneous acceleration clusters, posing great challenges for their cost model's generalization across different hardware. This challenge will be increasingly acute due to the continuous iteration of various NPUs. In contrast, tensor programs are more stable due to mature general-purpose components. **In summary, it is necessary to design a new cost model for tensorized programs**.

According to the requirements of an efficient cost model specialized for tensorized program tuning in heterogeneous acceleration clusters, we posit that the efficient cost model involves two key points: ❶ feature representation, i.e., how to represent programs as features learnable by the model; ❷ transfer prediction, i.e., how the model can support predictions on untrained hardware at low cost and with flexibility. To the best of our knowledge, we propose the first unified cost model for tensorized program tuning by introducing a unified feature representation and unified transfer prediction strategy.

2 Key Ideas and Methods

2.1 The Unified Feature Representation

Considering the acceleration platform diversity (**Req 1**) and intrinsic diversity (**Req 2**) in the heterogeneous cluster, the ideal feature representation for tensorized programs should pursue generality while ensuring prediction accuracy

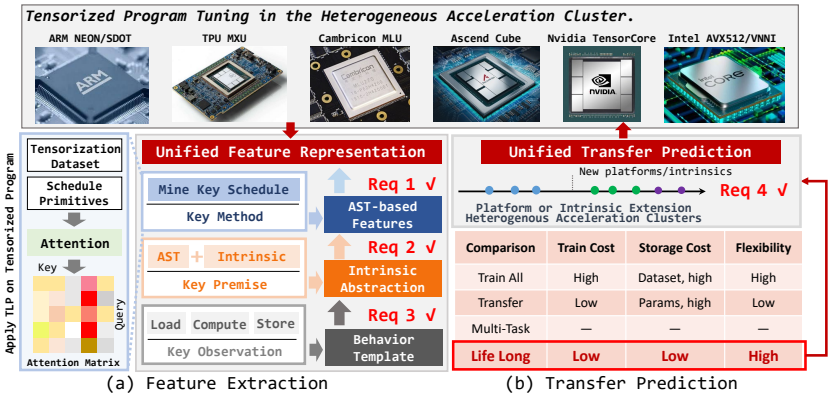

Fig. 2. Tensorized program space generation.

(**Req 3**). As shown in Fig. 2 (a), to meet three requirements above, we propose the unified feature representation for tensorized programs.

Tensorized programs exhibit a unified program behavior, which can be abstracted into a three-stage execution model: load, compute, and store. This unified behavior remains consistent across platforms. Therefore, the behavior template allows tensorized programs running on different platforms to be unifiedly represented (**meeting Req 3**). Considering abstract syntax tree (AST) as a comprehensive program abstraction, it not only intuitively demonstrates program behavior but also incorporates hardware intrinsic information. This can further cope with intrinsic diversity (**meeting Req 2**). The target AST features should effectively reflect the impact of schedule differences on program performance to ensure accuracy. Although TLP's feature representation method lacks generality, it achieves SOTA accuracy[7]. We can utilize the attention matrix of TLP to mine the key schedules affecting program performance, and then construct target AST features based on these key schedules (**meeting Req 1**).

2.2 The Unified Transfer Prediction Strategy

Considering acceleration platform extension and hardware intrinsic extension in heterogeneous acceleration clusters, the cost model must support transfer prediction at low cost and with flexibility (**Req 4**). Multiple learning methods are compared in Fig. 2 (b). Lifelong learning as an incremental learning method is particularly suitable for heterogeneous acceleration cluster scenarios (**meeting Req 4**). Models do not forget old intrinsics when learning new ones, significantly reducing storage and training costs and allowing flexible adaptation. We adopt the method of selective synaptic plasticity[5] to address the transfer prediction. Specifically, UCMoH is trained on new intrinsics using the following loss function,

$$L'(\theta) = L(\theta) + \lambda \sum_i b_i(\theta_i - \theta_i^b)^2 \tag{1}$$

Where $L(\theta)$ is the lambda rank loss[6]. b_i is the importance of parameters. θ_i is the parameter to be learned. θ_i^b is the parameter converged on old tasks.

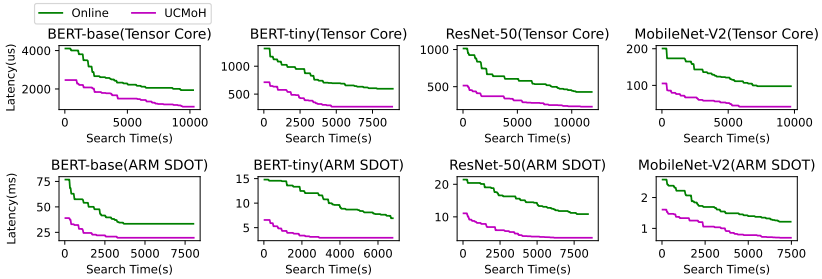

Fig. 3. Tuning curves of UCMoH and Online on tensor core and arm.sdot.

3 Implementation and Evaluation

To meet the training and testing requirements, we first construct TensorizeSet, a tensorized program dataset, which includes 10 hardware intrinsics (6 on *tensor core*, 2 each on *x86.avx512/vnni* and *arm.neon/sdot*) from 3 hardware platforms (NVIDIA A100 GPU, Xeon Platinum 8369B, ARM Yitian710) and 84 common neural network structures (CV and NLP). This dataset will be open-sourced and updated with intrinsics of *Ascend* and *Cambricon* in the future. Based on lifelong learning method, UCMoH is trained on TensorizeSet. It sequentially learns the 6 matmul intrinsics of *tensor core* and the dot product intrinsic of *arm.sdot*. The model architecture of UCMoH includes attention and fully connected layers.

We incorporate UCMoH into TVM MetaSchedule[2], the SOTA tensorized program tuning framework for evaluation. Each network is tensorized to 6 matmul intrinsics of *tensor core* and dot product intrinsic of *arm.sdot*. After 2000 rounds of tuning, UCMoH is compared with the default online cost model. The tuning curves are shown in Figure 3. These curves indicate that UCMoH can converge to lower latency faster. UCMoH can improve the inference speed by 1.9× on average. UCMoH can speed up the search time by 9.3× on average (the time required for UCMoH to achieve the latency of Online tuning 2,000 times).

Acknowledgments. This work was supported in part by the National Key R&D Program of China under Grants 2022YFB4501600 and 2022YFB4501603.

References

1. Feng, S., Hou, B., Jin, H., Lin, W., et al.: Tensorir: An abstraction for automatic tensorized program optimization. In: Proc. of ASPLOS (2023)
2. Shao, J., Zhou, X., Feng, S., Hou, B., Lai, R., Jin, H., et al.: Tensor program optimization with probabilistic programs. Proc. of NIPS (2022)
3. Wang, C., et al.: A ubiquitous machine learning accelerator with automatic parallelization on fpga. TPDS (2020)
4. Wang, C., et al.: Wookong: A ubiquitous accelerator for recommendation algorithms with custom instruction sets on fpga. TC (2020)
5. Wang, L., et al.: A comprehensive survey of continual learning: Theory, method and application. Proc. of PAMI (2024)
6. Wang, X., Li, C., Golbandi, N., Bendersky, M., Najork, M.: The lambdaloss framework for ranking metric optimization. In: Proc. of CIKM (2018)
7. Zhai, Y., Zhang, Y., Liu, S., Chu, X., Peng, J., et al.: Tlp: A deep learning-based cost model for tensor program tuning. In: Proc. of ASPLOS (2023)