Exploring the Performance Bound of Cambricon Accelerator in End-to-End Inference Scenario

Yifan Wang[1,2], Chundian Li[1,2], and Chen Zeng[1,2]

[1] SKL of Computer Architecture, Institute of Computing Technology, CAS, China
{wangyifan2014, lichundian, zengchen}@ict.ac.cn
[2] University of Chinese Academy of Sciences, China

Abstract. Deep learning algorithms have become pervasive in a broad range of industrial application scenarios. DianNao/Cambricon family is a set of energy-efficient hardware accelerators for machine learning, especially for deep learning, covering from edge embedded devices to cloud data centers. However, in the real application scenario, the complicated software stack and the extra overhead (memory copy) hinder the full exploitation of the accelerator performance. In this paper, we try to explore the performance bound of Cambricon accelerator MLU100 in end-to-end deep learning inference scenarios (from data/model load to inference results store). We leverage the offline model to bypass the general deep learning framework, use the multiple threads programming to fully exploit the parallelism of the multi-core accelerator and apply specific data structure to decrease the memory copy overhead. The evaluation results show that, for RetNet-50 on CIFAR-10 dataset, our optimization methods are 32.09× faster than the baseline of the optimized batch size (64), and achieve 85% of the performance upper-bound on the Cambricon MLU100 board.

Keywords: DianNao/Cambricon Accelerator · End-to-End Optimization · RetNet-50 on CIFAR-10.

1 Introduction

Deep learning algorithms have become pervasive in a broad range of industrial application scenarios, such as autonomous driving, natural language processing, and advertisement recommendation. To meet the explosive growth of deep learning application requirements, many machine learning accelerators have been designed both from the academic and the industry, such as DianNao/Cambricon accelerators[3, 4], Google TPU systolic architecture[15], and IBM TrueNorth neuromorphic processor[19].

These hardware accelerators achieved high performance to process machine learning workloads in ideal evaluation environments. However, in the real application environments, developers usually utilize the hardware accelerators via machine learning frameworks, which are too high-level to handle the specific operator-level optimization on hardware[2, 24], hindering the full exploitation of the accelerator performance. To address this problem, we focus on a specific end-to-end inference scenario, using a Cambricon MLU100 board[1] to process the image classification task over 10,000 images. This task is the Cambricon track on BenchCouncil AI Challenges, the other three tracks are AI system challenge of RISC-V[13] and X86 platform[7], and 3D face recognition algorithm challenge[23]. AIBench provides the workloads, datasets, and the baseline for the AI challenges[9, 10]. The source code of AIBench is publicly available from `http://www.benchcouncil.org/benchhub/AIBench/` (Sign up to get access).

To explore the performance bound of the Cambricon board over the specific end-to-end inference scenario, we adopt three optimization methods. Firstly, we use the serializing tool provided by Cambricon to generate a offline model, which bypasses the deep learning framework, accessing the hardware accelerator via runtime library. Secondly, we use the multiple threads programming to maximize the memory bandwidth of the external DRAM on the Cambricon board, fully exploiting the parallelism of the multi-core accelerator. Thirdly, we convert the image data to the expected input data structure of the Cambricon chip to decrease the memory copy overhead. According to evaluation results, our optimization methods are 32.09× faster than the baseline of the optimized batch size (64), and achieve 85% of performance upper-bound on the Cambricon MLU100 board.

The rest of this paper is organized as follows. In Section II, we introduce the background. Section III presents our optimization methods. Section IV illustrates the evaluation results. Finally, we conclude our work in Section IV.

2 Background

In this section, we briefly introduce the specific deep learning task, including the hardware accelerator, the neural network model and the input dataset.

DianNao Family and Cambricon MLU100 Accelerator. DianNao[3] is a high-throughput and energy-efficient accelerator for the deep neural network proposed in 2014. A large neural network can be split into small workloads to reuse the NFU (Neural Functional Unit) and SRAM buffers efficiently, so DianNao can execute neural networks on different scales. To support more machine learning algorithms and application scenarios, they have proposed various accelerator architectures, such as PuDianNao[18] for polyvalent machine learning algorithms and ShiDianNao[8] for computer vision algorithms on edge. Cambricon MLU100 is a commercial product based on DaDianNao[5] architecture, which is a multi-core supercomputer for machine learning. Cambricon MLU100 boards have been widely deployed in cloud servers to accelerate numerous machine learning applications.

Residual Neural Network and CIFAR-10 dataset. Residual Network (ResNet)[12] is a classic deep neural network for image classification, which has been adopted in a lot of deep learning benchmarks as the evaluation workload[6, 10, 11]. ResNet leverages the residual blocks to address the degradation problem and achieves the lowest error on the ImageNet test set in 2015, and wins the

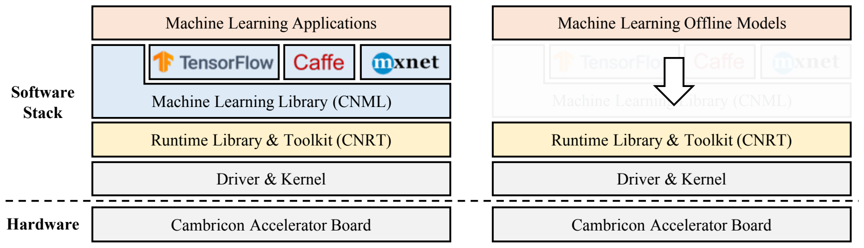

Software Stack
Hardware
Machine Learning Applications
TensorFlow
Caffe
mxnet
Machine Learning Library (CNML)
Runtime Library & Toolkit (CNRT)
Driver & Kernel
Cambricon Accelerator Board
Machine Learning Offline Models
Runtime Library & Toolkit (CNRT)
Driver & Kernel
Cambricon Accelerator Board

Fig. 1. Online (Left) and Offline (Right) Deployment.

ILSVRC classification competition[20] in 2015. And the deep residual learning framework has been one of the main trends in designing the deep neural networks[14, 21, 22]. CIFAR-10 dataset consists of 60, 000 32×32 colour and labeled images in 10 classes[16], which is a standard dataset to evaluate the algorithm performance[12, 17]. And CIFAR-10 also has been used in lots of deep learning benchmarks as the standard input[6, 10].

3 Design and Implementation

In this section, we present the detailed optimization methods of the end-to-end inference workload mentioned above. Meantime, we illustrate the optimization results for each step.

3.1 Offline Model

Cambricon accelerators provide two programming libraries for users, the CNML (Cambricon Neuware Machine Learning) library and the CNRT (Cambricon Neuware Runtime) library. These two libraries allow developers to leverage the machine learning acceleration engine in the Cambricon board with the user-friendly programming framework and interface.

The libraries support online and offline approaches to deploy the machine learning algorithms on Cambricon accelerators, as shown in Fig. 1. The online method is a traditional way to deploy the machine learning model. The model is managed by the CNML framework, or other general machine learning frameworks (e.g. Caffe and TensorFlow). The offline method usually serializes the compiled computing graph and operators to a new file, generating a new model file (offline model), which can be loaded directly by the CNRT library. The online or offline approach decides whether the system will load the machine learning framework when executing the machine learning applications. The offline method makes the machine learning model bypass the machine learning framework and achieves higher performance than the traditional way in the end-to-end inference scenario.

The performance of the offline deployment method is shown in Fig. 2. Compared with the online deployment method, the offline method is 1.58× faster on average and 1.52× faster on the optimal bath size (64).

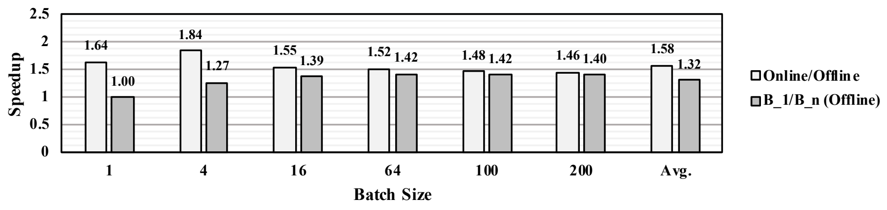


Fig. 2. Speedup of offline model optimization method over online, and $batch\_size = n$ over $batch\_size = 1$ in offline model method.

3.2 Multiple Threads

The Cambricon MLU100 provides n machine learning acceleration cores and m memory controllers (channels), $n \bmod m = 0$. Each channel manages an external DRAM and creates a hardware-queue (HQ) in the OS kernel. For a single thread program, the n acceleration cores access the data from 1 external DRAM, which will cause the bandwidth contention and degrade the accelerator performance. To exploit the parallelism of the memory controller, we create a m-thread program and bind each thread to a different channel (hardware-queue in OS), to maximize the data access bandwidth between the acceleration cores and external DRAMs.

There are two methods to leverage multiple acceleration cores on MLU100, model parallel and data parallel. The model is partitioned into n cores in the model parallel method, and the n cores will process 1 image collaboratively. In the data parallel method, each core is allowed to access the whole model, and the n cores will process n images concurrently and independently. The model parallelism introduces extra overhead because of the intermediate result copy and fusion. To exploit the parallelism of the multi-core, we set model parallelism to 1 and data parallelism to n (number of cores).

As shown in Fig. 3 after exploiting the parallelism of the MLU100, compared with the single-thread method, the parallel programming method is 11.83× faster on average and 12.17× faster on the optimal bath size (64).

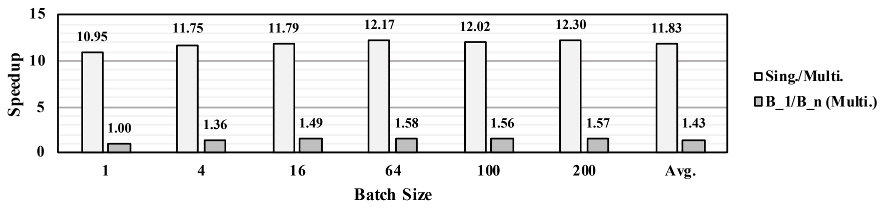
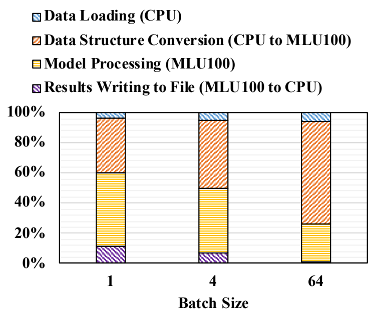
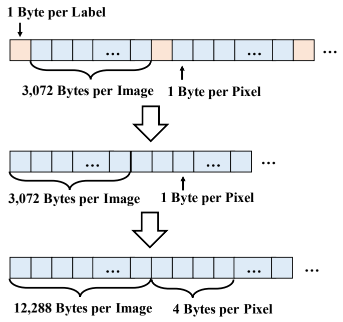


Fig. 3. Speedup of multiple-thread optimization method over single-thread, and $batch\_size = n$ over $batch\_size = 1$ in multiple-thread method.



(a) Running time breakdown



(b) Data structure conversion

Fig. 4. Running time breakdown and data structure conversion.

3.3 Data Structure Conversion

To fully understand the bottleneck of the program, we collect the running time of each step in the program, and the running time breakdown is shown in Fig. 4(a). While leveraging the two optimization methods mentioned above, the data loading, model processing, and results writing to file only cost 6.06%, 25.74%, and 0.81% of the total processing time, respectively, while the image data type conversion costs 67.39% of the total time. The data structure conversion overhead is caused by the mismatch problem. The loaded images are formatted according to the data structure in the CIFAR-10 dataset, each image has 1 byte label data and $3,072$ bytes pixel data. However, the MLU100 takes contiguous address space of the image data as the input, and each pixel is represented by a `float32` value. The data type conversion process is illustrated in Fig. 4(b).

To reduce the extra processing time caused by the data type mismatch, we reformat the image data according to the MLU100 input data structure, and the performance is shown in Fig. 5. After eliminating the data type mismatch problem, the performance is 1.60× faster on average and 1.73× faster on the optimal bath size (64).

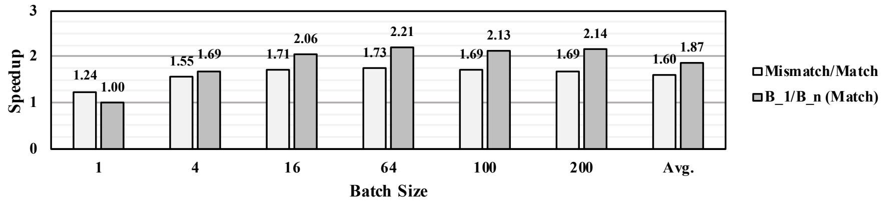


Fig. 5. Speedup of data structure matching optimization over mismatch, and $batch\_size = n$ over $batch\_size = 1$ in matching method.

4 Evaluation

As we presented above, our optimization methods are 32.09× faster than the baseline. The peak performance of Cambricon MLU100 board is 64 TeraFlops (half-precision, 16-bit)[1], and the I/O bandwidth of the 4 channels PCIe is 8 GB/s, the Roofline Model of Cambricon MLU100 board is illustrated in Fig 6. The Opt.3 is the final performance after we adopt three optimization methods, which achieve 85% of performance upper-bound on the Cambricon MLU100 board.

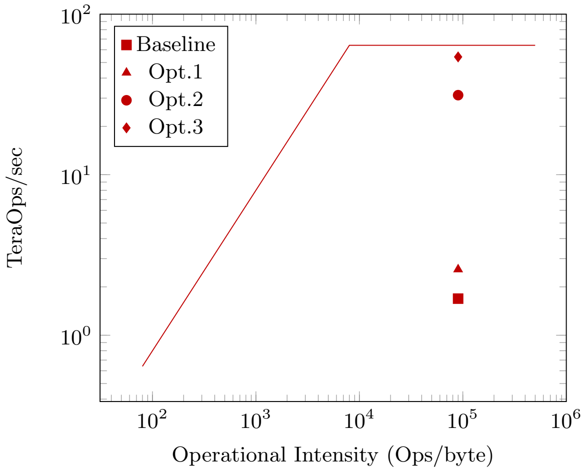


Fig. 6. Roofline Model of Cambricon MLU100 Board

5 Conclusion

Even though some hardware accelerators are energy-efficient in micro-benchmarks, a lot of potential is under the performance upper-bound while the end-to-end applications are executed on accelerators. To explore the performance bound of Cambricon board over the specific end-to-end inference scenario, we adopt three optimization approaches, while reducing the memory copy overhead and speeding up by utilizing better parallelism. The evaluation results show our optimization methods reveal 32.09× faster than the baseline on the optimal batch size (64). In addition, we achieve 85% of performance upper-bound of the Cambricon MLU100 board.

References

1. Cambricon Technologies: Cambricon MLU100. (2019), `https://en.wikichip.org/wiki/cambricon/mlu/mlu100/`

2. Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E., Shen, H., Cowan, M., Wang, L., Hu, Y., Ceze, L., et al.: TVM: An automated end-to-end optimizing compiler for deep learning. In: 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI). pp. 578–594 (2018)
3. Chen, T., Du, Z., Sun, N., Wang, J., Wu, C., Chen, Y., Temam, O.: DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In: Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). pp. 269–284. ACM (2014)
4. Chen, Y., Chen, T., Xu, Z., Sun, N., Temam, O.: DianNao family: energy-efficient hardware accelerators for machine learning. Communications of the ACM pp. 105–112 (2016)
5. Chen, Y., Luo, T., Liu, S., Zhang, S., He, L., Wang, J., Li, L., Chen, T., Xu, Z., Sun, N., et al.: DaDianNao: A machine-learning supercomputer. In: Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). pp. 609–622. IEEE (2014)
6. Coleman, C., Narayanan, D., Kang, D., Zhao, T., Zhang, J., Nardi, L., Bailis, P., Olukotun, K., Ré, C., Zaharia, M.: DAWNBench: An end-to-end deep learning benchmark and competition. Training **100**(101), 102 (2017)
7. Deng, W., Wang, P., Wang, J., Li, C., Guo, M.: PSL: Exploiting parallelism, sparsity and locality to accelerate matrix factorization on x86 platforms. In: International Symposium on Benchmarking, Measuring and Optimization (Bench). Springer (2019)
8. Du, Z., Fasthuber, R., Chen, T., Ienne, P., Li, L., Luo, T., Feng, X., Chen, Y., Temam, O.: ShiDianNao: Shifting vision processing closer to the sensor. In: Proceedings of the 42nd International Symposium on Computer Architecture (ISCA). pp. 92–104. ACM (2015)
9. Gao, W., Luo, C., Wang, L., Xiong, X., Chen, J., Hao, T., Jiang, Z., Fan, F., Du, M., Huang, Y., et al.: AIBench: towards scalable and comprehensive datacenter AI benchmarking. In: International Symposium on Benchmarking, Measuring and Optimization. pp. 3–9. Springer (2018)
10. Gao, W., Tang, F., Wang, L., Zhan, J., Lan, C., Luo, C., Huang, Y., Zheng, C., Dai, J., Cao, Z., et al.: AIBench: an industry standard internet service AI benchmark suite. arXiv preprint arXiv:1908.08998 (2019)
11. Gao, W., Zhan, J., Wang, L., Luo, C., Zheng, D., Tang, F., Xie, B., Zheng, C., Wen, X., He, X., Ye, H., Ren, R.: Data motifs: A lens towards fully understanding big data and ai workloads. The 27th International Conference on Parallel Architectures and Compilation Techniques (PACT) (2018)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). pp. 770–778. IEEE (2016)
13. Hou, P., Yu, J., Miao, Y., Tai, Y., Wu, Y., Zhao, C.: RVTensor: A light-weight neural network inference framework based on the RISC-V architecture. In: International Symposium on Benchmarking, Measuring and Optimization (Bench). Springer (2019)
14. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4700–4708 (2017)
15. Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al.: In-datacenter performance analy-

sis of a tensor processing unit. In: Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA). pp. 1–12. IEEE (2017)
16. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
17. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems (NeurIPS). pp. 1097–1105 (2012)
18. Liu, S., Du, Z., Tao, J., Han, D., Luo, T., Xie, Y., Chen, Y., Chen, T.: Cambricon: An instruction set architecture for neural networks. In: Proceedings of the 43rd International Symposium on Computer Architecture (ISCA). pp. 393–405. IEEE (2016)
19. Merolla, P.A., Arthur, J.V., Alvarez-Icaza, R., Cassidy, A.S., Sawada, J., Akopyan, F., Jackson, B.L., Imam, N., Guo, C., Nakamura, Y., et al.: A million spiking-neuron integrated circuit with a scalable communication network and interface. Science **345**(6197), 668–673 (2014)
20. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Alexander, C.B., Li, F.F.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**(3), 211–252 (2015)
21. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4510–4520 (2018)
22. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1492–1500 (2017)
23. Xiong, X., Wen, X., Huang, C.: Improving RGB-D face recognition via transfer learning from a pretrained 2D network. In: International Symposium on Benchmarking, Measuring and Optimization (Bench). Springer (2019)
24. Zhao, Y., Du, Z., Guo, Q., Liu, S., Li, L., Xu, Z., Chen, T., Chen, Y.: Cambricon-F: machine learning computers with fractal von Neumann architecture. In: Proceedings of the 46th International Symposium on Computer Architecture (ISCA). pp. 788–801. ACM (2019)