Accelerating Inference of 3D-CNN on ARM Many-core CPU via Hierarchical Model Partition

Jiazhi Jiang, Zijiang Huang, Dan Huang*, Jiangsu Du, and Yutong Lu*

School of Computer Science and Engineering, Sun Yat-sen University, China
Email: {jiangjzh6, huangzj53, huangd79, dujs, luyutong}@mail.sysu.edu.cn

***Abstract*—Many applications such as biomedical analysis and scientific data analysis involve analyzing volumetric data. This spawns huge demand for 3D CNN. Although accelerators such as GPU may provide higher throughput on deep learning applications, they may not be available in all scenarios. CPU, especially many-core CPU, remains an attractive choice for deep learning in many scenarios. In this paper, we propose a inference solution that targets on the emerging ARM many-core CPU platform. A hierarchical partition approach is claimed to accelerate 3D-CNN inference by exploiting characteristics of memory and cache on ARM many-core CPU.**

***Index Terms*—3D-CNN, NUMA, ARM, manycore CPUs**

I. Introduction

3D-CNN has grabbed huge attention due to its wide range of applications. Although accelerators such as GPU can provide higher throughput than CPUs on deep learning applications, they may not be optimal on all scenarios. For memory-hungry applications such as 3D-CNN that consume tens to hundreds of gigabytes of memory, fundamental operations like 3D convolution or even 2D convolution with large batch sizes can often easily exceeds the available memory capacity of GPU. Among various CPU architectures, the ARM ecosystem has gained considerable attention in both HPC and data centers. Due to the energy-efficient designs of ARM architecture, the manufacturers tend to integrate dozens or even hundreds of light-weight cores to boost the performance of CPU. What's more, several ARM CPU are linked on one server with NUMA style to share memory and bring higher aggregate computing power. Although many-core CPU integrates far more cores than traditional multi-core CPU and expect to bring higher aggregate computing power, overheads such as thread contention and synchronization are also more severe leading to neutralizing the performance gain obtained by more cores. NUMA architecture may result in frequent remote memory access.

To address these issues and exploit the aggregate computing power of many-core CPU on ARM servers, more efficient parallelism and strong scaling capability of 3D-CNN are required. We propose an effective 3D-CNN inference approach based on model parallelism to improve parallel efficiency and scalability of 3D-CNN on ARM many-core CPU. The backbone of our approach is to hierarchically partition 3D-CNN model and distribute the inference of 3D-CNN model with awareness of cache and memory architecture.

II. Decouple 3D-CNN model

To avoid frequent data exchange and movement introduced by model parallelism, channel partition including group convolution and channel shuffle techniques are exploited to turn 3D-CNN into loosely-couple structure at the channel dimension. Depth partition including depth sampling and depth inserting are exploited to decouple 3D-CNN at depth dimension.

A. Channel-based Decoupling

Group convolution is first proposed in Alexnet [1]. It is used to divide the convolution into multiple GPUs to compute separately due to the limitation of hardware resources at that time. As there is no data exchange between channel groups in group convolution, we use group convolution to replace the convolution of 3D-CNN model. Each convolution kernel only operates on the input within the corresponding group. With the decrease in the amount of computation, the accuracy of the trained model is also slightly degraded. The reason is that the convolution receptive fields are confined within the channel group, resulting in a lack of cross-channel feature perception. To enable cross-channel feature perception and mitigate the problem of accuracy of degradation, a channel shuffle stage is employed at the end of each layer in the 3D-CNN model.

B. Depth-based Decoupling

3D convolution has an extra depth dimension than 2D convolution. To further expose a high degree of parallelism, a second-level decoupling is employed in the depth dimension. Unlike the channel dimension, the depth dimension contains temporal information of the dataset. We stride the depth dimension and partition it into groups like $[0, 2, 4, 6.....N-2]$ and $[1, 3, 5.....N-1]$ instead of $[0, N/2), [N/2, N)$. In this way, 3D-convolution operation in each group can obtain global temporal information of the dataset.

III. Optimization of 3D-CNN inference

In this section, we a employ hierarchical partitioning approach including inter-node partition and intra-node partition for inference of 3D-CNN to fully exploit the cache and memory architecture of ARM many-core server as shown in Figure 1.

A. Scheme-1: Inter-node partition

Since Section II has retrofitted 3D-CNN structure to decouple the models in the channel, the input tensor, convolution kernel, and intermediate output within a group are completely independent from that of other groups. The decoupled 3D-CNN models

978-3-9819263-7-8/DATE23/© 2023 EDAA

are partitioned by channel group for model parallelism in the inference stage. In detail, input tensor and convolution kernels in different channel groups are partitioned into different NUMA nodes. All input tensors and parameters within a channel group are allocated into the same NUMA node. This is different from the implementations in traditional frameworks, which put all input tensors and parameters into one NUMA node. It makes threads of different NUMA nodes have to obtain all input tensors and parameters from remote memory resulting in the inefficiency of 3D-CNN inference. Intermediate results are interpolated and merged after the convolution is completed.

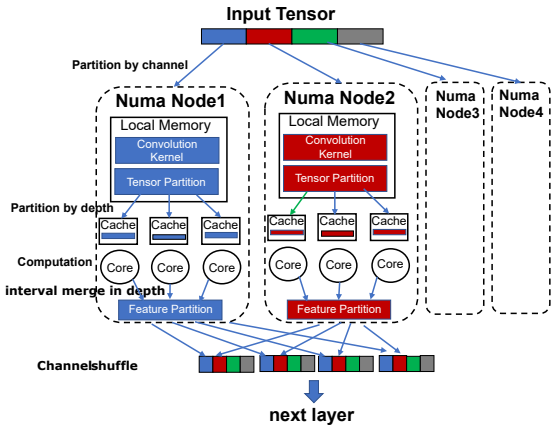

Fig. 1. Hierarchical model partition for inference of 3D-CNN.

B. Scheme-2: Intra-node partition

Within each NUMA node, the partitioning method is further refined. Since Section II retrofitted the 3D-CNN structure to further decouple the models in the depth dimension. The decoupled 3D-CNN model is further partitioned by stride sampling for depth in the inference stage. With multi-threading parallelism, the threads tend to handle arbitrarily access to any data partition. To alleviate the potential thread contention on the tensor partition of each NUMA node. The threads on the NUMA node are also divided into groups, and each group of threads is responsible for computing the fine-grained data blocks from depth sampling.

C. Scheme-3: Thread management

To alleviate the thread contention and remote memory access, we propose a thread-binding to be jointly employed with scheme-1 and scheme-2 for the co-location between input data and threads on many-core processors. To co-locate computation with the required data on the same NUMA node, a primary thread will be created for each NUMA node. These primary threads are bound to the respective NUMA nodes. Each primary thread will spawn a local thread pool on each NUMA node, which contains worker threads responsible for actual computation of 3D-CNN. The worker threads in the thread pool are also bound to the local node. Since data partitions have been allocated on local node with scheme-1, there is no need to invoke thread migration or access data from remote memory. Threads in the local thread pool are pinned to more fine-grained data blocks sliced by scheme-3 on local node. The cache-able partitions are computed with a particular group of threads.

IV. Experiments and Evaluation

We decouple and train the three 3D-CNN models on a DGX server with four V100 GPU. Our experiments for 3D-CNN inferences are set up on the Kunpeng920 processor. It has 128 ARM cores and 4 NUMA nodes. Each NUMA node is equipped with 192GB of 8-channel DDR4 memory.

TABLE I
Impacts of model decoupling on accuracy.

Dataset	CNN model	Decoupled	Top-1 Acc	Top-5 Acc
Jester	3D-resnet34	None	93.9	99.6
		Decouple	92.7	99.5
	3D-resnet50	None	94.1	99.7
		Decouple	92.8	99.5
	C3D	None	93.8	99.6
		Decouple	92.8	99.5

A. Model Accuracy Analysis

In this subsection, comparisons of accuracy between decoupled 3D-CNN models and the original CNN models on Jester dataset are provided to show the effectiveness of our approach. The results are shown in Table I. The baseline is the original 3D-CNN models. It can be seen that the impact of our network structure decoupling approach on accuracy is negligible. Although the decoupling operation may result in a loss of accuracy, the accuracy compensation measures such as channel shuffle and stride sampling mitigate this problem.

B. Overall performance analysis

Figure 2 presents the inference time of 3D-CNN on ARM many-core server following different implementing methodologies. ACL-opt is an implementation based on the ACL library combined with all optimization schemes. With our optimizations, ACL achieves significant speedup against the naive implementation. When applying our solution on ACL, 128 threads are divided into groups for independent computation. Compared with the naive implementation with ACL, it significantly alleviates the overheads brought by concentrating too many threads on a single operator. In addition, most of the remote memory accesses are also eliminated. Applying our optimized solution on NCNN also consistently outperforms its naive implementation.

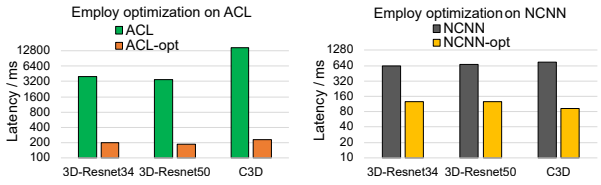

Fig. 2. Performance comparison of 3D-CNN inference with 128 threads for different implementations.

V. Conclusion

In this paper, we present a hierarchical model partitioning approach to boost the performance of 3D-CNN inference on ARM many-core systems. With our optimizations, it outperforms the naive version significantly.

References

[1] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," *NeurIPS*, vol. 25, 2012.