Optimizing Loop Partitioning in TVM

Wuwei Lin*
wuweil@andrew.cmu.edu

Yingjing Lu*
yingjinl@andrew.cmu.edu

1 Project Website

https://yingjinglu.github.io/pages/compiler-proj.html

2 Project Description

To achieve good performance by AOT compilation using TVM, the input shape need to be known at compile time. This is impractical because we often want to change the batch size or the shape in runtime. Allowing variables in the input shape relaxes this constraint. However, naive approach may often result in excessive amount of conditionals to enumerate all possibilities of the variable, and it is difficult to achieve good performance.

We would like to enhance loop partition algorithm to tackle dynamic shape. Loop partition is helpful for dynamic shape if it can partition the shape into suitable tiles so that we can utilize external highly-optimized microkernels, which often has strict requirements for the input shape. We would like to explore heuristics for loop partition, such as utilizing performance of microkernels for different shapes, to partition the loop in an efficient way. In addition, partitioning the loop may also introduce more possibilities for cache reuse optimization, pipelining across different loop segments and more parallelization.

In this project, we intend to extend the existing TVM stack with a loop partitioning optimization module. The basic functionality of the system takes in a tensor computation loop with some given heuristics, and to partition the loop so that the loop runs faster / consumes less energy than the naive serial execution. Potential field of optimization includes parallel loop execution, tiling, scheduling, cache optimization and pipeline scheduling. A cost and exploration model for auto tuning performance. The optimization could be taken further if the target hardware environment is given where the algorithm can gear directly towards the configuration for full optimization.

3 Objectives

75% Objective: Construct appropriate test cases with dynamic batch size and Benchmark existing loop partition module in TVM, and analyze the bottleneck of the existing loop partition module. Extend the current loop partition module to support different loop partition strategies that leverage the information from the cost module. A cost module that takes in the procedure and analyze the number of execution cycle, cache, memory utilization and energy/time consumption given a general hardware environment such as a CPU environment or a x86 CPU environment.

100% Objective: Based on 75% goal. A loop partition analysis module that takes in the loop and conditionals in the loop, with variable input shape size, achieves better data reuse, parallelization than the naive loop pass for general x86 CPU and GPU environment. The pass should be able to partially automatically explore the optimization space given the feedback cost from the cost module.

* Denotes equal contribution

Preprint. Under review.

125% Objective: Based on 100% goal, A loop partition analysis module that can further explore data reuse on cache/memory, parellization for a given hardware configuration such as a design spec of an accelerator. It can further calculate the pipelining according to the config. The cost model should also be able to support cost calculation for energy consumption, execution cycle, calculations for cost feedback to guide optimization. Combine the loop partition pass and the scheduling part of the TVM IR such that given the information of loop partition and the cost module, we can automatically tensorize the loop body to utilize external microkernels.

4 Timeline

Mar 23 - 29
Literature review and existing work exploration **(Lin, Lu)**

Mar 30 - Apr 5
Construct test cases and benchmark existing loop partition module in TVM, and analyze its bottleneck. **(Lin)**
Implement the cost model **(Lu)**

Apr 6 - 12
Extend the loop partition module to support different strategies **(Lin)**
Implement and test the cost model with the loop analysis module **(Lu)**

Apr 13 - 19 **Milestone**
Expect to nearly achieve the 75% goal **(Lu, Lin)**
Expecting to achieve a basic working version of the loop analysis module that can export the loop execution procedure.
Expect the cost model to export the cost for the given procedure analysis.

Apr 20 - 26
Implement the cost exploration procedure. **(Lu)**
Implement the loop procedure optimization to leverage microkernels. **(Lin)**

Apr 27 - May 3
Extend the analysis to arbitrary input hardware configuration such as an FPGA **(Lu, Lin)**
Testing and summarize the results, write out the report. **(Lu, Lin)**

5 Literature Search

The TVM paper that introduces the architecture, cost modules and existing features Chen et al. [2018a].

FlexTensor, an automatic scheduling exploration module for tensor computation based on TVM Zheng et al. [2020].

Learning to optimize tensor program: introduces a ML based tensor computation cost exploration model that utilizes boosting tree and treeGRU to efficiently explore the optimization search space without manual tuning Chen et al. [2018b].

6 Resources Needed

Some example loop components in the deep learning tensor computation workloads.

TVM build and testing environment such as consumer x86 CPU, consumer grade GPU and enterprise GPU, FPGA, ARM processors. Some of which could be accessed through AWS.

References

Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, Carlos Guestrin, and Arvind Krishnamurthy. TVM: An automated end-to-end optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 578–594, Carlsbad, CA,

October 2018a. USENIX Association. ISBN 978-1-939133-08-3. URL https://www.usenix.org/conference/osdi18/presentation/chen.

Tianqi Chen, Lianmin Zheng, Eddie Yan, Ziheng Jiang, Thierry Moreau, Luis Ceze, Carlos Guestrin, and Arvind Krishnamurthy. Learning to optimize tensor programs. In *Advances in Neural Information Processing Systems*, pages 3389–3400, 2018b.

Size Zheng, Yun Liang, Shuo Wang, Renze Chen, and Kaiwen Sheng. Flextensor: An automatic schedule exploration and optimization framework for tensor computation on heterogeneous system. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 859–873, 2020.