深度学习编译器架构综述

李煦阳　DZ21330015

2022

摘要

本文对深度学习编译器的架构设计进行整理与总结。

1　前言

深度学习为人类社会许多行业送来了强大的工具，用以完成一些曾被认为只有人类才能完成的“智能”任务。工程师会根据应用的具体背景与条件，设计、训练深度学习模型，这些模型将被部署在集群或边缘设备上。为了加速模型的设计过程，工业界与学术界共同设计了数款深度学习框架，如 TensorFlow，PyTorch，MXNet 等，它们设计不同、各自拥有优劣。同时，由于深度学习矩阵的运算特点，许多硬件架构也被提出用来加速深度学习模型的执行。若能统一且高效地实现深度学习模型到目标硬件计算的映射，可以加速深度学习应用与研究的迭代。

人们开始设计深度学习编译器。类似 LLVM，它也是一个经典的多前端多后端的架构，不同的深度学习模型将被翻译至统一的中间表示，完成检查、按需优化，再翻译至指定的目标硬件上。但由于深度学习自身特点，其编译器的具体设计细节与传统编译器相异，本文将对深度学习编译器架构的核心设计与组件进行介绍。

2　架构设计

图1展示了深度学习编译器的基本架构设计。总体上，它分为前端和后端，前者统一不同深度学习框架的输入格式并进行与目标硬件无关的优化与调试，后者根据目标硬件特点与编译模式进行优化、适配与代码生成。前端与后端有着较大差异，深度学习编译器于是采取多层中间表示，适应针对性不同的转换、优化算法。

高层中间表示　以图表示硬件无关的计算与控制流。良好设计算子抽象，使其可以表达不同的深度学习框架模型的计算与控制流是这一部分的难点。该层表示需要建立算子与数据间的控制流与依赖关系，并要易于优化算法的使用。有时也需要它具有一定拓展性，适配模型定制的算子。高层中间表示上的优化，尝试减少图上的冗余并提升效率。这些优化包括在传统编译器中的常见优化，也包括与深度学习特定相关的优化。根据优化范围，可分类为结点级、块级、数据流级优化。

低层中间表示　用于硬件相关优化与代码生成，编译器后端根据输入模型与目标硬件的先验知识，为它补充了足以表达硬件特征的细粒度信息。后端会针对性地进行指令集、内存、并行化等相关

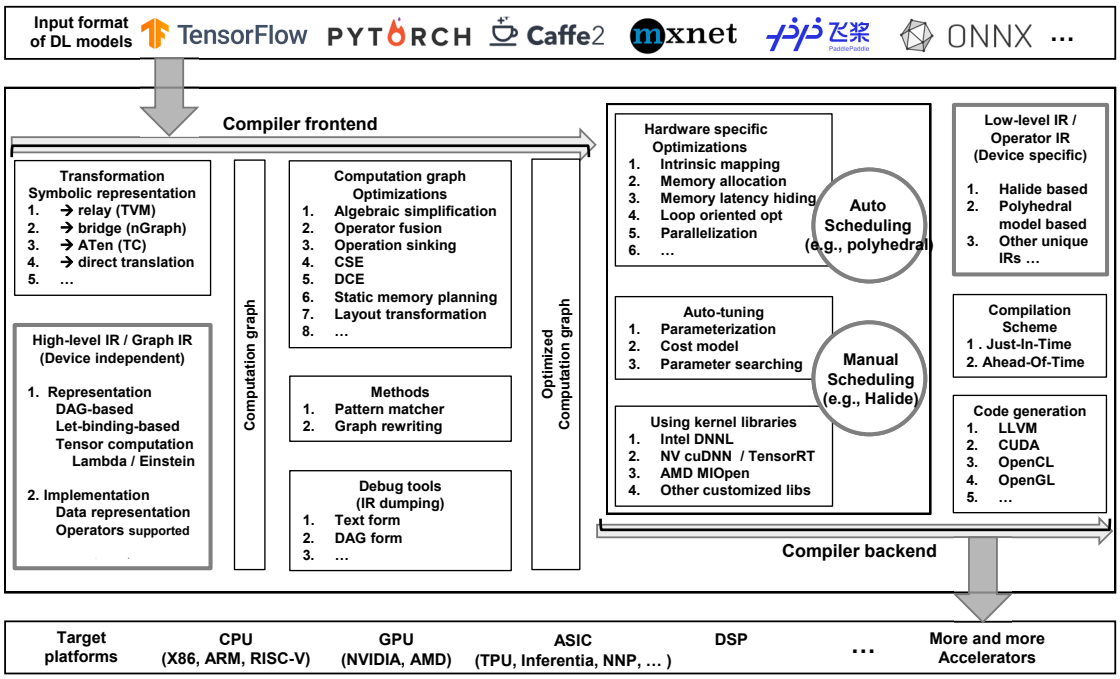

图 1: 深度学习编译器架构总览

硬件优化。在选择具体优化参数时，编译器会采取自动重排（auto-scheduling）或自动调谐（auto-tuning）两个策略；根据模型运行环境，编译器也会采取 JIT 或 AOT 作为具体的编译策略。目标代码生成时一般借助成熟的第三方工具链，如 LLVM。

3 架构组件细节

3.1 高层中间表示

深度学习模型表达的计算，相比传统程序更复杂，因而其编译器的中间表示需要有表达更复杂语义的能力。

3.1.1 基本表示

以有向图表示计算图是一种基本手段，图中节点代表原子的深度学习算子，如卷积、池化等，图中的边表示张量（tensor）。不考虑循环的存在，图是自然无环的。传统编译器也会用图（数据依赖图）作为表示，许多优化可以直接迁移使用，如公共子表达式删除、无用代码删除等。而外地，结合深度学习的领域知识，可以再做进一步优化。图表示是简单的，但是它并不能明确地表达计算的作用范围（scope），具有一定的语义模糊性。为了处理这种语义的模糊性，可以引入 *let*-绑定对表达式命名、约束作用范围，形成 A-normal form（ANF）的中间表达形式。ANF 常被用做函数式语言的中间表示。

深度学习的特点是其算子的复杂性，其编译器中要支持的算子可以分为 5 类，一般代数算子，神经网络算子，张量操作算子（`resize` 等），广播、规约算子（*min*），控制流算子。控制流算子可以函数式地用递归实现。有三类表达算子的方式。

1. 函数形式表达，提供基本函数集合，用以组合成更复杂的算子，
2. λ-表达式形式，可由用户自行定义计算函数，
3. 爱因斯坦标记，用于表达累加相关算子（需要支持结合律与交换律），这种表达形式可以方便并行化优化。

具体的实现中，编译器以占位符表示具体的张量，包含变量名与各维度的大小信息。对于动态输入无法直接确定的维度，标记以 *Any*. 编译器也要明确张量的内存布局（layout），对逻辑索引和物理索引准确映射。

3.1.2 前端优化

前端优化在作用范围上可分为三类，节点优化，窥孔优化，数据流优化。

1. 节点优化：可以删去输入张量有 0 维度的计算，
2. 窥孔优化：根据语义对局部的图进行变换，将图更规整化（算子下沉，operator sinking），连续的若干运算约减成更简单的（算子融合，operator funsion）等，
3. 数据流优化：如公共子表达式删除，无用代码删除，内存使用设计，布局转换。

3.2 低层中间表示

后端一般都会 LLVM 工具链生成目标代码，但是在之前，会在次低层中间表示上进行及其有关的优化。Halide-based IR 将计算与调度进行分离，编译器会尝试寻找最优调度；Polyhedral-based IR 可以支持嵌套循环。也有一些编译器自己设计了低层中间表示。

后端会对低层中间表示进行硬件内部指令映射，内存分配与预取、延迟隐藏，和循环相关优化（循环整合、重排、展开），并行化。

深度学习模型的运行性能与许多设置相关，进行机器相关优化时需要选择参数，这需要编译器进行搜索，以最佳适配模型与目标硬件。这种技术被称为自动调谐。

4 总结

深度学习编译器的架构，主体上完成了多前端、多后端的连系，并针对深度学习特殊的领域知识，对中间层、尤其是中间层模型表示进行了特殊设计。相信在未来，更多组件会被适配入架构之中，以使其支持更全面（如考虑训练阶段）、更安全（如边缘系统上的隐私保护）、更多元化（如子图分割，单模型多后端）的需求。

本文参考综述 [LLL+21]。

参考文献

[LLL+21] Mingzhen Li, Yi Liu, Xiaoyan Liu, Qingxiao Sun, Xin You, Hailong Yang, Zhongzhi Luan, Lin Gan, Guangwen Yang, and Depei Qian. The deep learning compiler: A comprehensive survey. *IEEE Transactions on Parallel and Distributed Systems*, 32:708–727, 03 2021.