先进编译实验室
Advanced Compiler

动态shape深度学习编译器论文分享：DISC

嘉宾：王磊

先进编译实验室
Advanced Compiler

DISC : A Dynamic Shape Compiler for Machine Learning Workloads

Kai Zhu, Wenyi Zhao, Zhen Zheng, Tianyou Guo, Pengzhan Zhao, Feiwen Zhu
Junjie Bai, Jun Yang, Xiaoyong Liu, Lansong Diao, Wei Lin
Alibaba Group
{tashuang.zk, kevin.zwy, james.zz, tianyou.gty, pengzhan.zpz, feiwen.zfw
j.bai, muzhuo.yj, xiaoyong.liu, lansong.dls, weilin.lw}@alibaba-inc.com

Abstract

Many recent machine learning models show dynamic shape characteristics. However, existing AI compiler optimization systems suffer a lot from problems brought by dynamic shape models, including compilation overhead, memory usage, optimization pipeline and deployment complexity. This paper provides a compiler system to natively support optimization for dynamic shape workloads, named *DISC* . *DISC* enriches a set of IR to form a fully dynamic shape representation. It generates the runtime flow at compile time to support processing dynamic shape based logic, which avoids the interpretation overhead at runtime and enlarges the opportunity of host-device co-optimization. It addresses the kernel fusion problem of dynamic shapes with shape propagation and constraints collecting methods. This is the first work to demonstrate how to build an end-to-end dynamic shape compiler based on MLIR infrastructure. Experiments show that *DISC* achieves up to 3.3× speedup than TensorFlow/PyTorch, and 1.8× than Nimble.

Keywords: machine learning, AI compiler, dynamic shape

1 Introduction

Machine learning technique evolves fast in recent years. It is applied in a wide range of areas, such as image/speech recognition, translation, recommendation to serve people's life. One reason of the boost of machine learning is the growth of computing power. With the help of machine learning frameworks (TensorFlow[8], PyTorch[13], MXNet[10]), machine learning algorithm representation can be mapped to powerful devices for convenient execution. However, this mapping is non-trivial and there is still a performance gap, especially for new models and scenarios.

The recent developed models expose dynamic shape problem, which is less optimized with current techniques. The operations suffering most from dynamic shape problems are those with small sized computations, like element-wise and reduction operations. Traditional techniques, like XLA[5], usually apply fusion approach to reduce the off-chip memory access and frequent kernel launch overhead for such operations. However, the existing kernel fusion engines could only generate kernels with static shape information inferred during compilation time. This results in a problem that, these fusion engines will compile and generate kernel for every emerging shape, even though some of them share the same computation pattern. It leads to severe compilation overhead when the number of shapes is large. Due to this reason, XLA is usually closed for dynamic shape workloads to prevent negative optimization.

Note that large ops, like GEMM/Conv, do not suffer much from dynamic shapes as they usually go through library calls (cuDNN, cuBLAS, oneDNN) rather than compilation optimizations. We focus on small sized ops optimization targets in this paper.

There are some workaround solutions for dynamic shape problem based on XLA. Developers can only cluster ops that have static shape for XLA to optimize, and leave ops with dynamic shape features run without fusion. This loses optimization opportunities to a extent. Furthermore, some workloads only have dynamic shaped ops in practice. Another workaround is to form tensors into a specific shape with padding and slicing, which introduces redundant computations and may lead to negative optimizations. None of the workarounds solves this problem fundamentally.

MLIR[12] provides the infrastructure towards a new machine learning compiler. It brings high extensibility to new functions and compatibility to existing optimization buildings. Meanwhile, it naturally supports dynamic shape optimization with its design philosophy. However, what it brings is the infrastructure, but not the solution to dynamic shape problem itself. Nimble[15] is a compiling framework based on TVM to address dynamic shape problem, which is a concurrent work with *DISC* and *DISC* has an earlier RFC release[7]. It provides a compiler framework capable

EuroMLSys '21, April 26, 2021, Online, United Kingdom

ACM ISBN 978-1-4503-8298-4/21/04…$15.00
https://doi.org/10.1145/3437984.3458838

DISC：A Dynamic Shape Compiler for Machine Learning Workloads

BladeDISC 是一个端到端的动态形状编译器项目，用于机器学习工作负载，这是阿里巴巴机器学习平台推出的通用推理优化工具PAI-Blade 的关键组件之一。

论文发表在 EuroMLSys 2021，EuroMLSys主要聚焦于支持AI/ML应用的构建框架、编程模型、优化算法和软件工程工具的新兴趋势，以及使用ML构建此类框架或优化工具。

背景

- 机器学习模型呈现出动态shape特征
  - 输入张量的shape变化
  - 模型计算过程中shape变化
- 现有深度学习编译器缺少对动态shape特征的优化的支持
  - 静态shape：编译时完全已知shape信息，可以帮助生成更优的优化决策进而生成高效代码
  - 动态shape：编译时无法已知全部shape信息，需要为每个shape编译，从而会增加编译开销、内存使用、优化和部署的复杂性
- 深度学习编译器支持动态shape相关工作
  - 区分动态shape和静态shape的算子后，仅对静态shape算子进行优化
  - 通过padding、slicing的方式对动态shape张量处理至特定shape
  - Nimble 将动态shape模型的运行时控制逻辑预构建为VM解释执行

DISC解决方案

问题一：现有IR缺少对动态shape计算的完整表示

- DISC基于MLIR，对XLA编译器中的HLO的IR进行复用扩充为DHLO，以支持动态shape表示

问题二：在运行时利用VM解释生成运行时流，带来解释开销且损失主机-设备协同优化机会

- DISC在编译时生成运行时流的代码，将主机端逻辑和设备计算一起编译

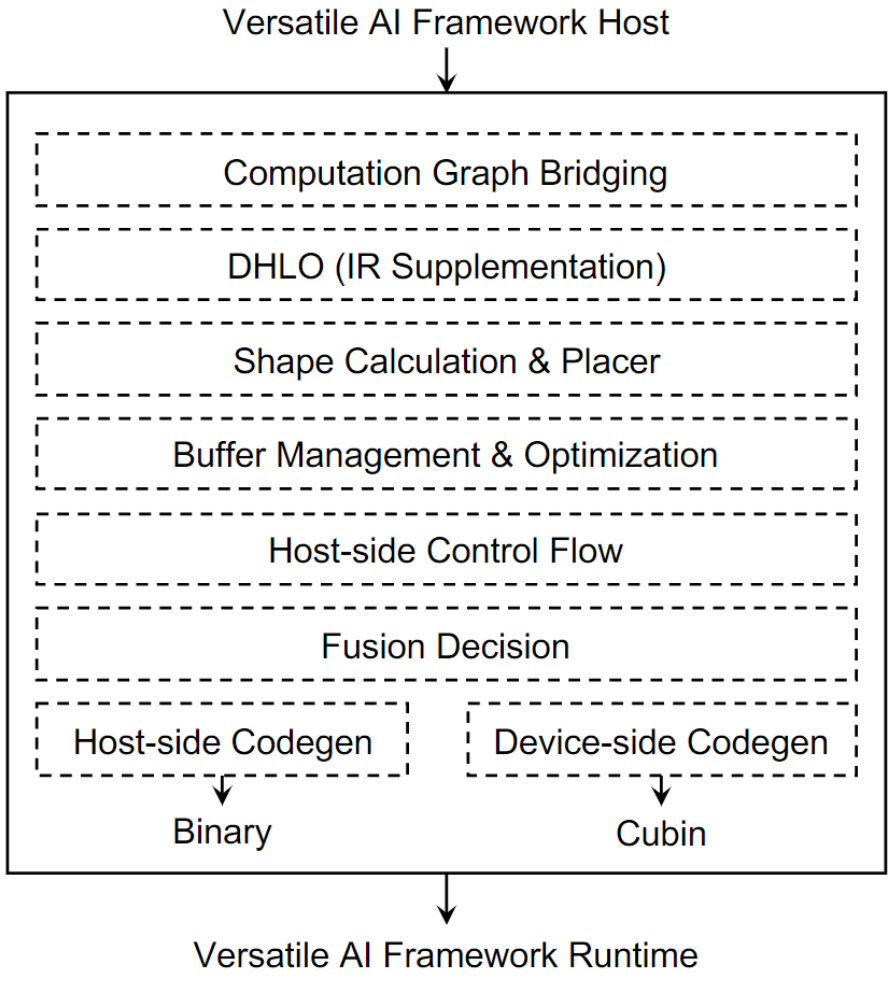
问题三：由于动态shape而无法进行内核融合等优化

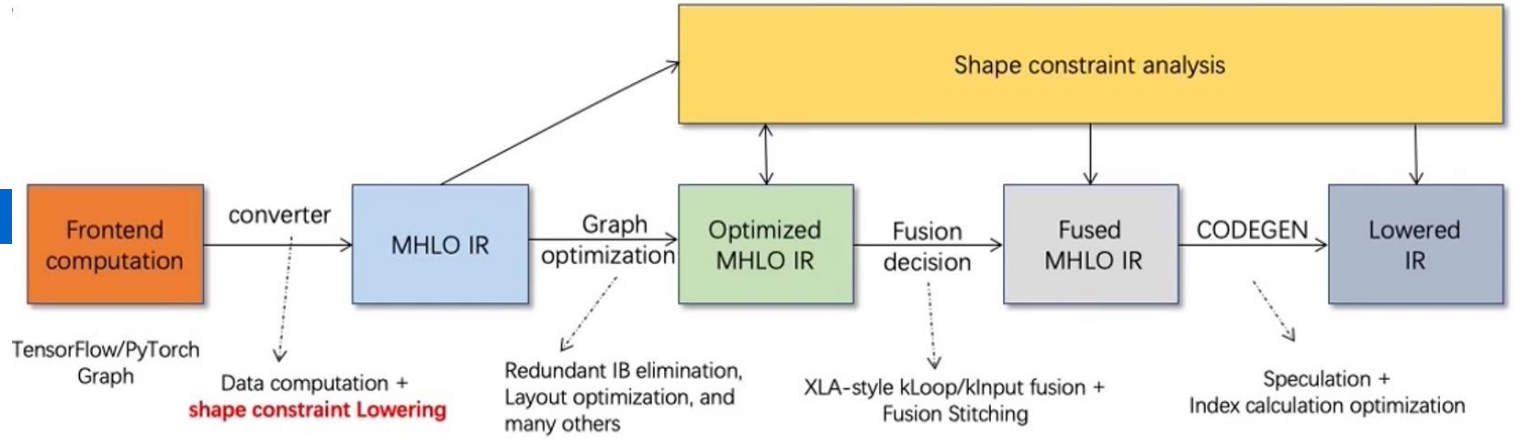
- DISC在基于算子间的shape传播特性和编译时捕获的shape约束信息，进行内核融合优化

问题四：深度学习编译器的灵活性

- DISC支持TensorFlow和PyTorch，且同时支持动态shape和静态shape混合情况

DISC架构概述





- AI框架的计算图Lower到DLHO、收集shape约束信息
- DLHO支持动态shape的表示
- 编译同时生成shape计算逻辑、主机-设备的计算逻辑放置
- 缓冲区管理与优化
- 外部库Lower、内核启动管理、设备管理以及与框架交互
- 根据算子调度兼容性分析执行融合决策
- 主机-设备端代码生成

DISC架构设计——DHLO

基于MLIR，对XLA编译器中的HLO的IR进行复用扩充为DHLO

```
def HLO_SliceOp:
    ......
    let arguments = (ins
        Hlo_Tensor:$operand,
        I64ElementsAttr:$start_indices,
        I64ElementsAttr:$limit_indices,
        I64ElementsAttr:$strides
    );
```

```
def HLO_DSliceOp:
    ......
    let arguments = (ins
        Hlo_Tensor:$operand,
        HLO_Tensor:$start_indices,
        HLO_Tensor:$limit_indices,
        HLO_Tensor:$strides
    );
```

支持主流框架：Lower框架中具有动态shape特征子图 到 DHLO

DISC架构设计——生成运行时流

■ Adaptive Shape Inference

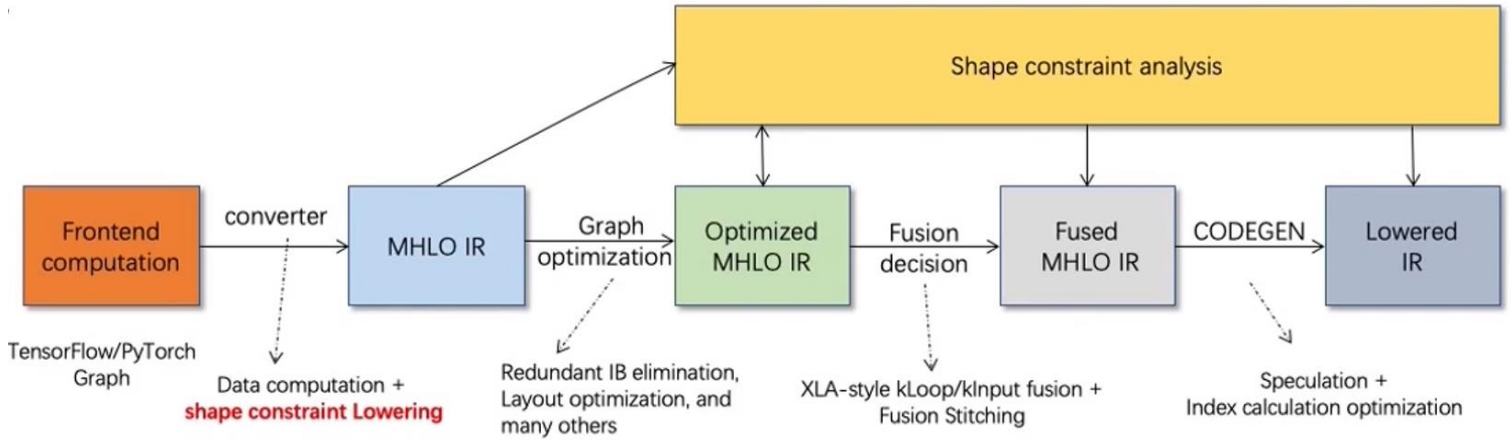
- ◆ Shape constraints：编译阶段，用于代码生成优化

  类型：dimension size equality、tensor size equality等

  来源：用户定义、前端输入、算子定义、IR分析、JIT捕获

- ◆ Shape calculation：运行时阶段

  编译生成shape inference代码置于主机端、内核计算代码置于设备端



DISC架构设计——内核融合及代码生成

■ 计算密集型算子

通常通过库调用的方式进行内核生成

支持在运行时根据shape选择合适的算子库实现

■ 访存密集型算子

内核融合策略：将具有相同数量元素的访存密集型算子融合在一起

Shape hints collection：shape propagation 和 shape constraints

Shape-adaptive fusion configuration：shape通用模板、生成多个shape的内核在运行时选择

DISC性能

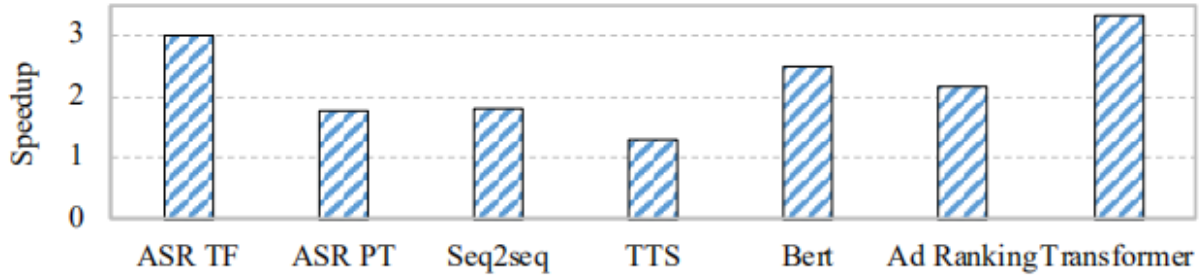


Figure 3. Speedup comparing with TensorFlow/PyTorch.

Table 2. Performance breakdown for Transformer.

Backend	Comp. bound	Mem. bound	CPU	E2E
Nimble	66.58	56.09	65.83	188.5
DISC	59.68	21.52	24.08	105.28

Table 3. Kernel number breakdown for Transformer.

Backend	Comp. bound	Mem. bound	Total
Nimble	5232	8632	13924
DISC	4476	6186	10734

参考资料

DISC : A Dynamic Shape Compiler for Machine Learning Workloads

https://alibaba.github.io/BladeDISC/index.html

相关工作

Nimble: Efficiently compiling dynamic neural networks for model inference

DietCode: Automatic Optimization for Dynamic Tensor Programs