

# DISC : A Dynamic Shape Compiler for Machine Learning Workloads

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

论文分享

wangzhen @ 2022.06.17

# 背景和动机

- 灵活多样的深度学习框架
  - TensorFlow, Pytorch, MxNet等
  - 可在多种算力强劲的设备上执行
  - 灵活性也带来了一些性能差距，尤其是对新模型而言
- 深度学习编译器可以弥补灵活性与性能之间的鸿沟
  - XLA
  - TVM
- 当前最先进的深度学习编译器均是面向静态shape的
  - shape在编译期是静态可知的
  - 静态shape可以带来如下好处：
    - 性能：图级别的优化、融合决策、代码生成以及scheduling等

# 背景和动机

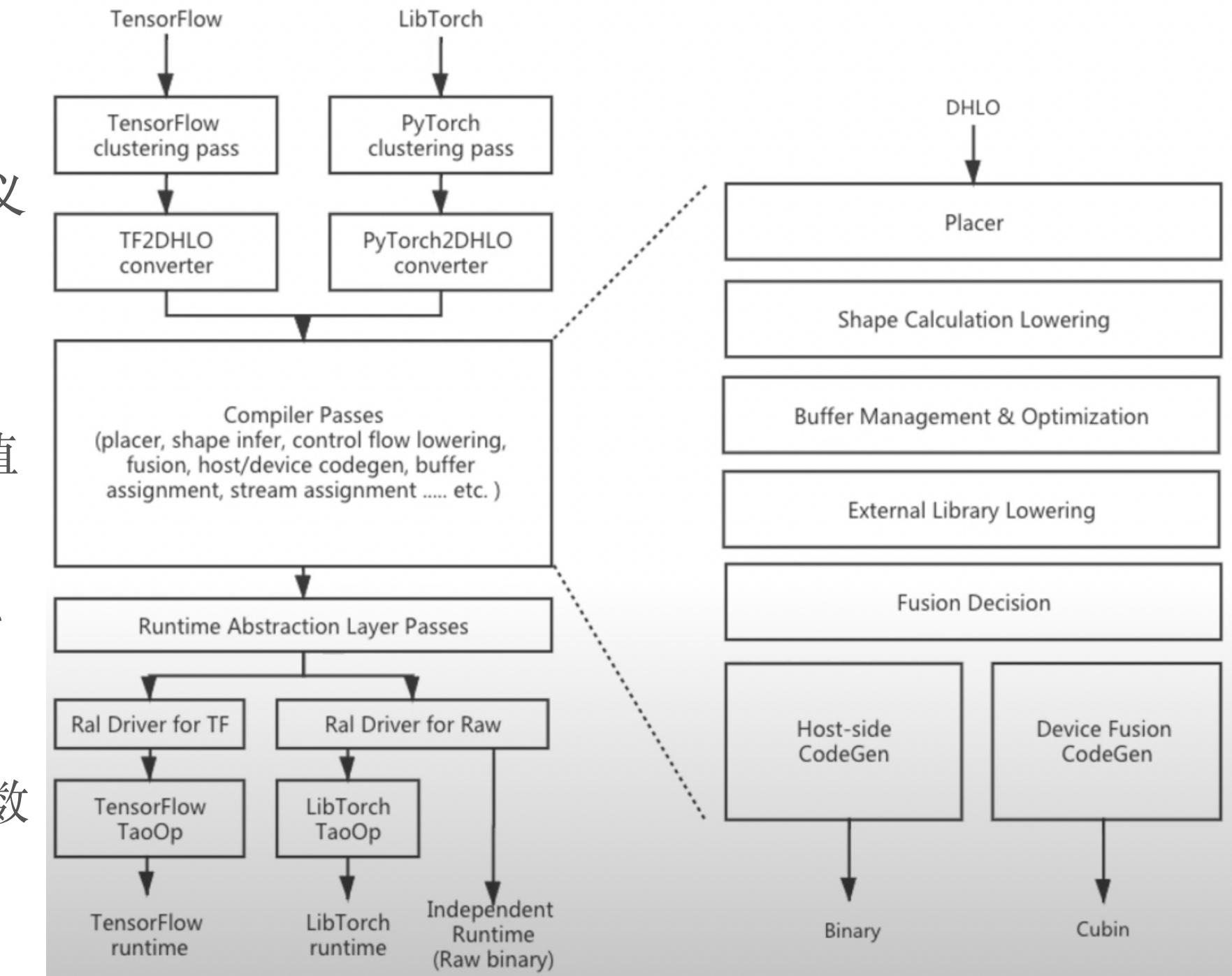
- 动态shape负载在基于编译器进行部署与应用时受到的阻碍
  - shape变化时带来的编译开销
  - 主机或者设备端的内存使用也不能得到良好的优化
  - 复杂的模型部署过程
  - 对于一些负载而言，shape的变化不受限制
- 动态shape负载样例
  - CV负载：处理不同的图片大小，如目标检测
  - 带有可变输入序列长度、可变输出序列长度以及可变batch size的Seq2seq模型
  - 稀疏负载在`Unique` Op的作用下生成可变的shape
    - `tf.feature\_column`
    - 分布式训练中的大规模embedding

# 概览

- DISC
  - 带有完全动态shape语义的端到端编译器架构
  - 一次编译，任意输入shape均可运行
  - 仅针对具有静态rank的动态shape（暂未发现较流行的动态rank应用）
- 主要贡献
  - 首款基于MLIR构建的支持动态shape的编译器
  - 提出一个支持完全动态shape语义的IR以及编译期生成的运行时流(`runtime flow`)
  - 在只有附加的shape约束信息而没有完整的shape信息条件下，仍可进行融合和代码生成操作
  - 多个前端框架支持，如TensorFlow和Pytorch
  - 支持静态和动态的混合优化

# 系统设计

- DHLO: mlir-hlo的扩展，可完整表示动态shape语义
- 编译期生成的运行时流
- 自适应shape推断
  - 在编译期标识shape约束，此时确定的shape值并不可知
  - 发射运行时代码，用于根据给定的输入tensor计算具体其shape值
  - `placer`: shape计算在主机端(CPU)完成，数据计算在设备端(GPU)完成
- 动态buffer管理
  - 发射`alloc`和`dealloc`指令
- 主机端控制
  - launch kernel/launch维度计算/供应商计算库调用/设备初始化/同步/cubins管理等



BladeDISC顶层模块设计图

# 系统设计

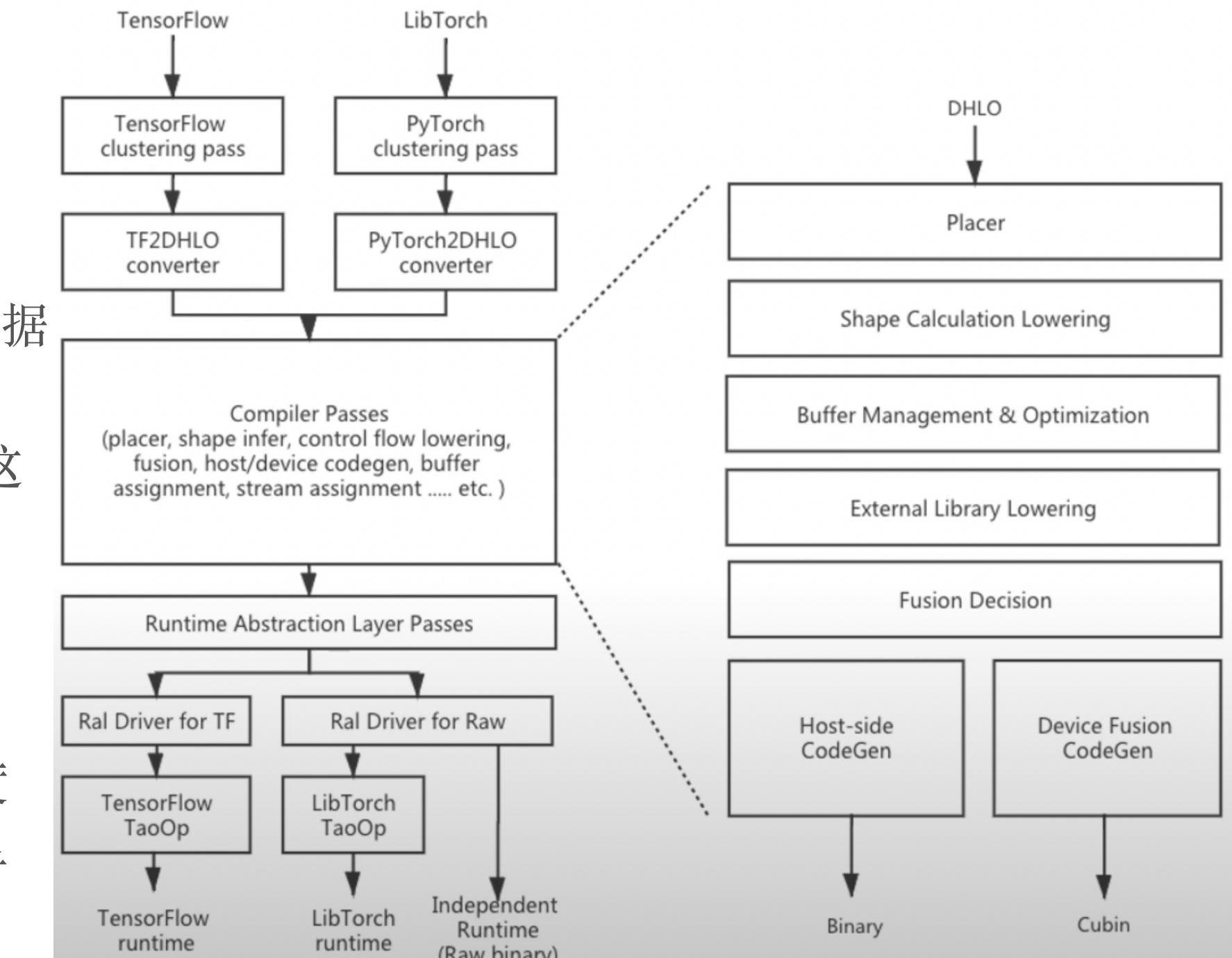
- 融合与代码生成

- Shape Hints: 根据图的拓扑序传递shape, 即根据shape传递确定未知shape, 如`Add op`的输出shape与输入shape是相同的。DISC内部将具有这种shape传递属性的op分类记录在一个表中

- Shape Constraints, 主要分为两类:

- 维度大小相等性限制: 一个tensor的某一维度大小是否等于该tensor的另一维度大小, 或者是否等于另一tensor的任意维度大小
- tensor大小等价性限制: 两个tensor是否拥有相同的元素个数

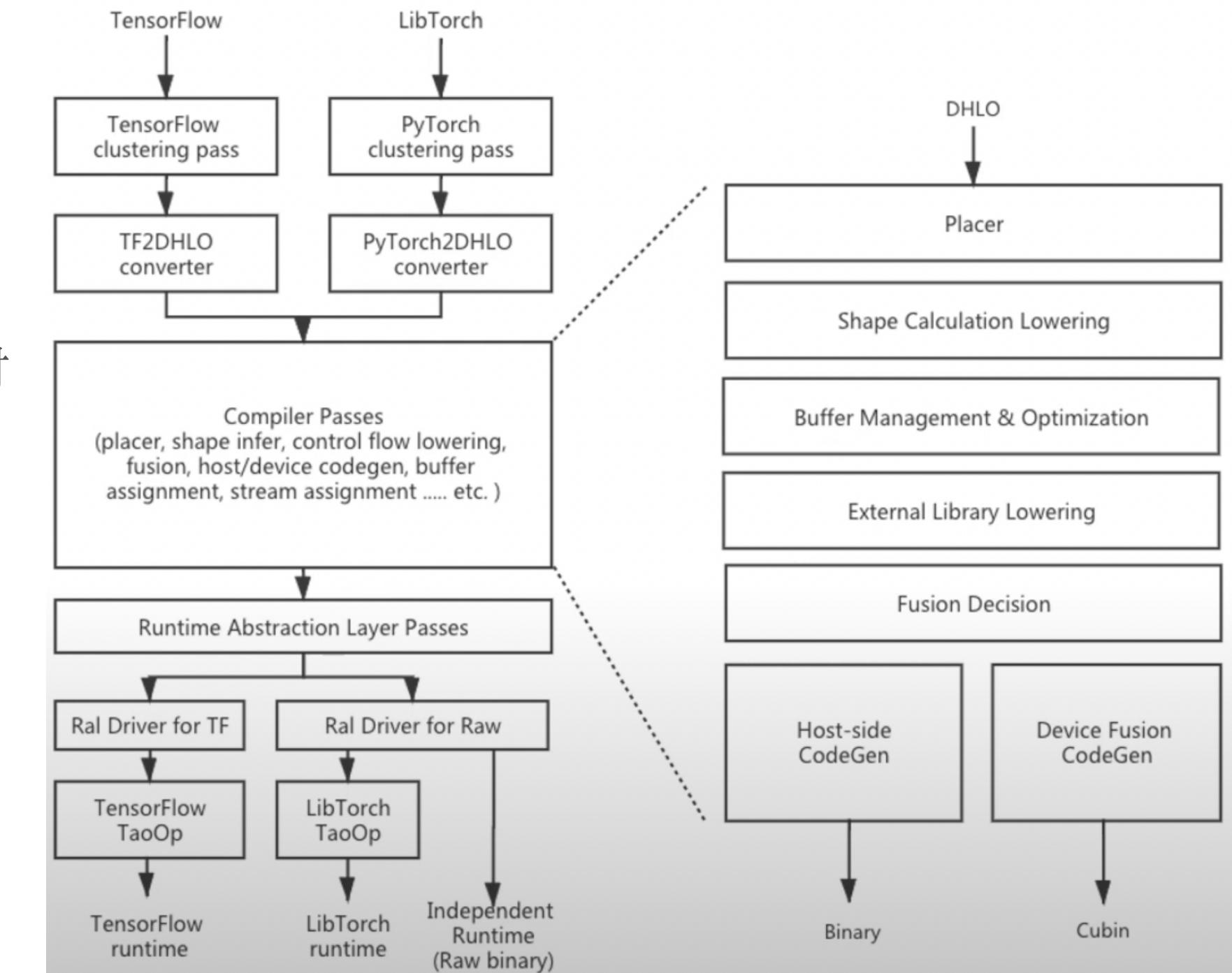
- 代码生成阶段, Shape Constraints可用于更激进的index计算化简



BladeDISC顶层模块设计图

# 系统设计

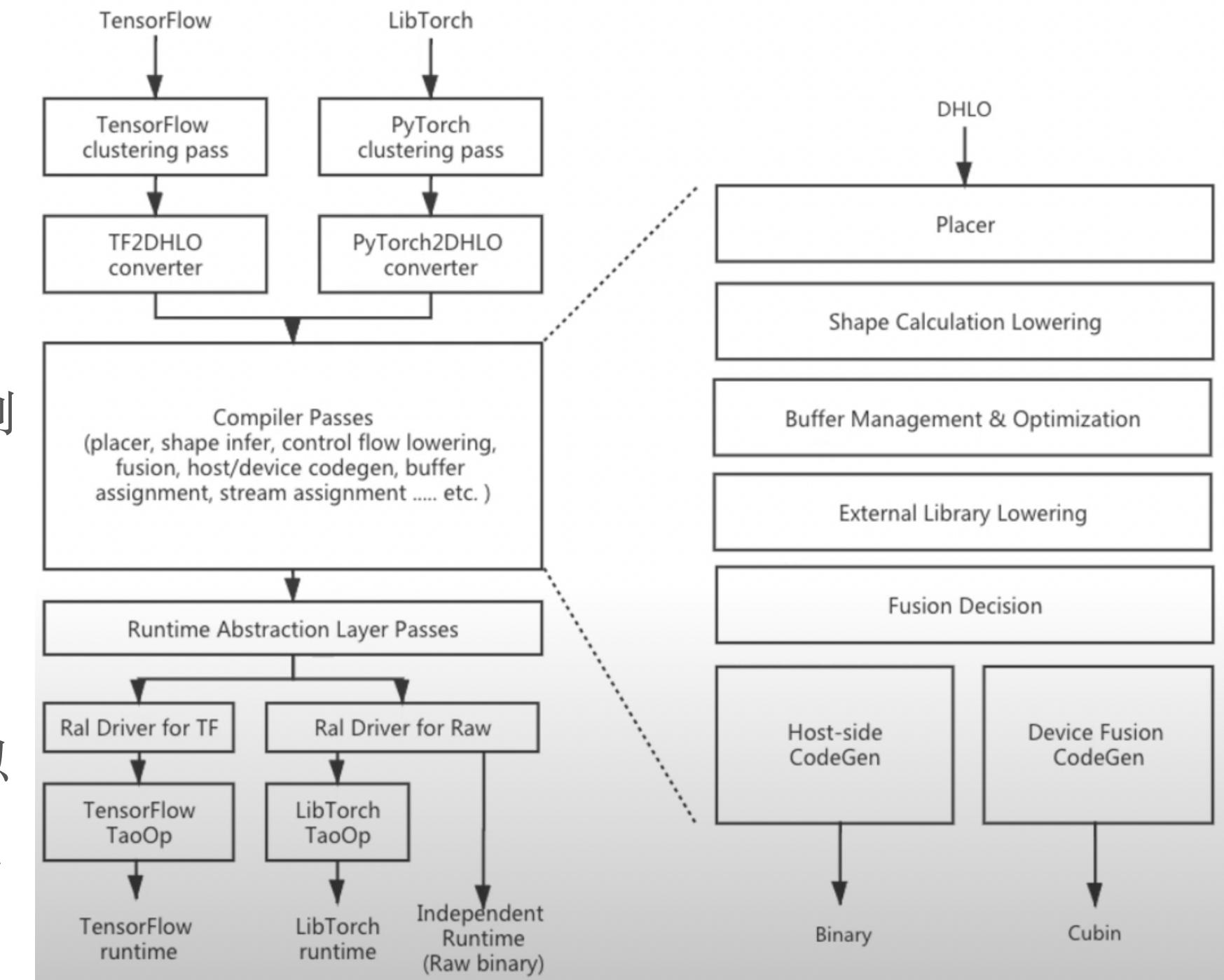
- 融合与代码生成
- 自适应shape融合配置
  - 内存受限模式的融合代码生成，倾向于选择对各种shape友好的模板，如经典的`loop fusion`和`input fusion`
  - 生成不同版本kernel以及主机端选择逻辑，用于针对不同的运行时shape做出不同响应：
    - launch维度的选择
    - 启用循环矢量化加载/存储与否
    - 是否需要隐式广播



BladeDISC顶层模块设计图

# 系统设计

- 多框架支持
  - 支持TensorFlow和PyTorch
  - 如果shape可知，DISC会优先将计算图lowering到静态shape编译器
- 静态shape库支持(主要针对计算密集型算子)
  - 静态shape库包括供应商库（cuBLAS/cuDNN）以及针对每种shape进行手工调优过的预生成kernel
  - DISC实现了一个接口，用于根据不同的运行时shape从库中选择最佳kernel



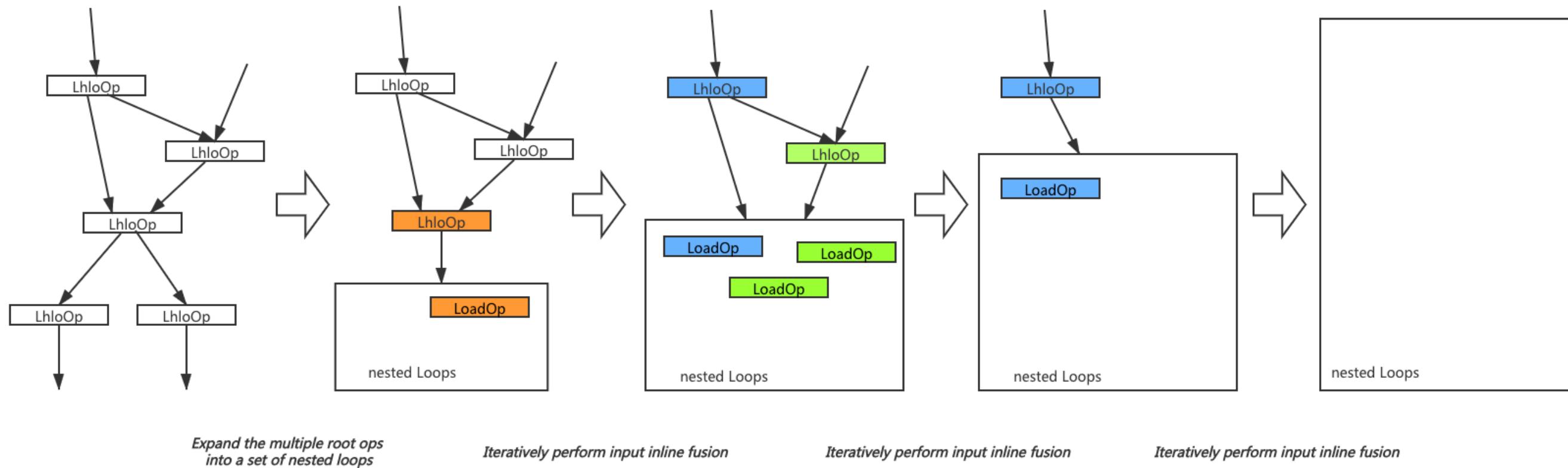
BladeDISC顶层模块设计图

# 示例讲解

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

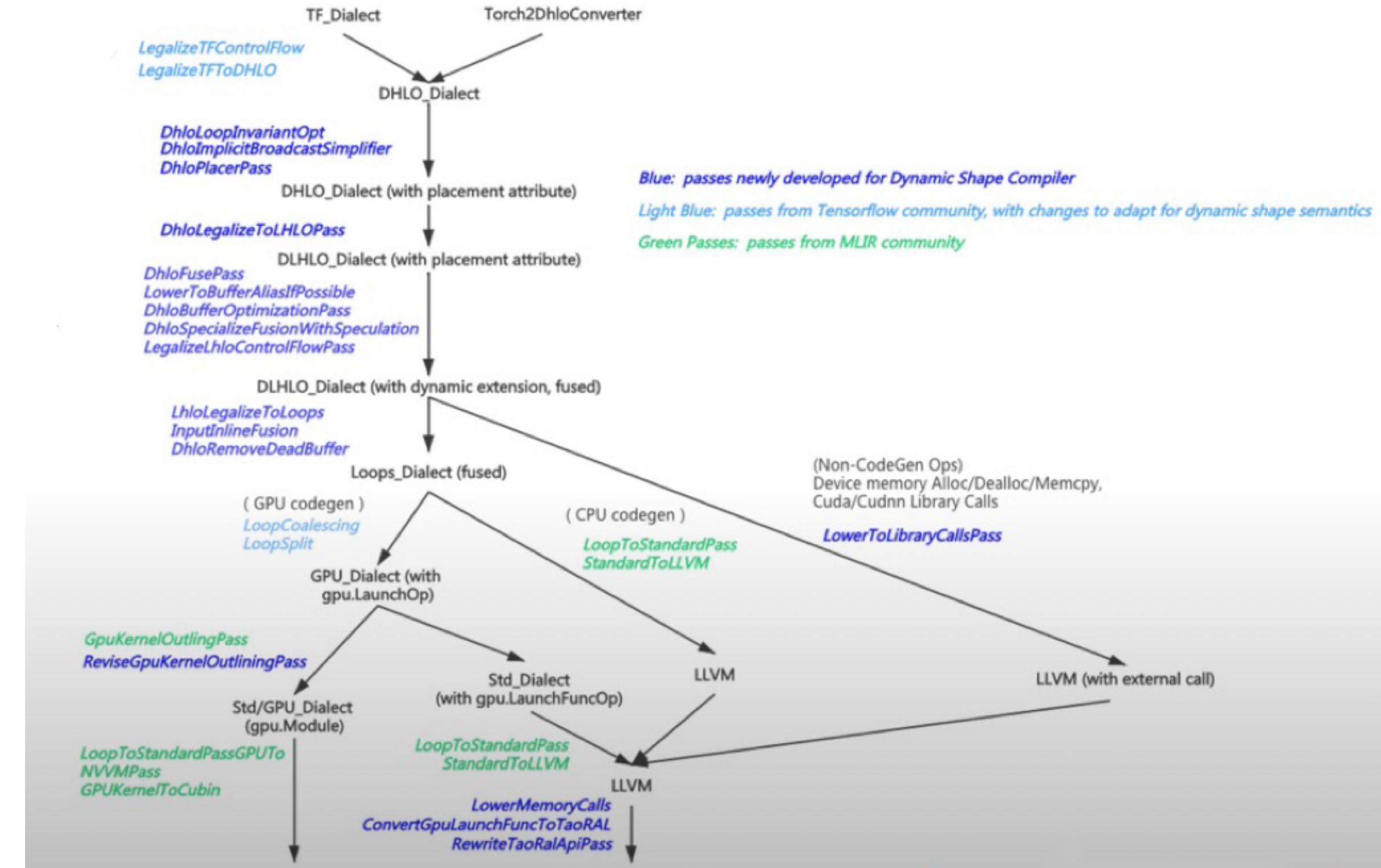
```
def HLO_DSliceOp:  
    ....  
    let arguments = (ins  
        Hlo_Tensor:$operand,  
        HLO_Tensor:$start_indices,  
        HLO_Tensor:$limit_indices,  
        HLO_Tensor:$strides  
    );
```

```
// An example of Shape Constraints on IR:  
// dim num element equality  
d0 = shape.get_dim(%0, 0)  
d1 = shape.get_dim(%0, 1)  
d2 = shape.get_dim(%1, 0)  
d3 = shape.get_dim(%2, 1)  
s0 = mul d0, d1  
s1 = mul d2, d3  
// tie_eq不会被lower到任何计算，仅仅是在IR表示中给予一些提示  
shape.tie_eq(s0, s1)
```



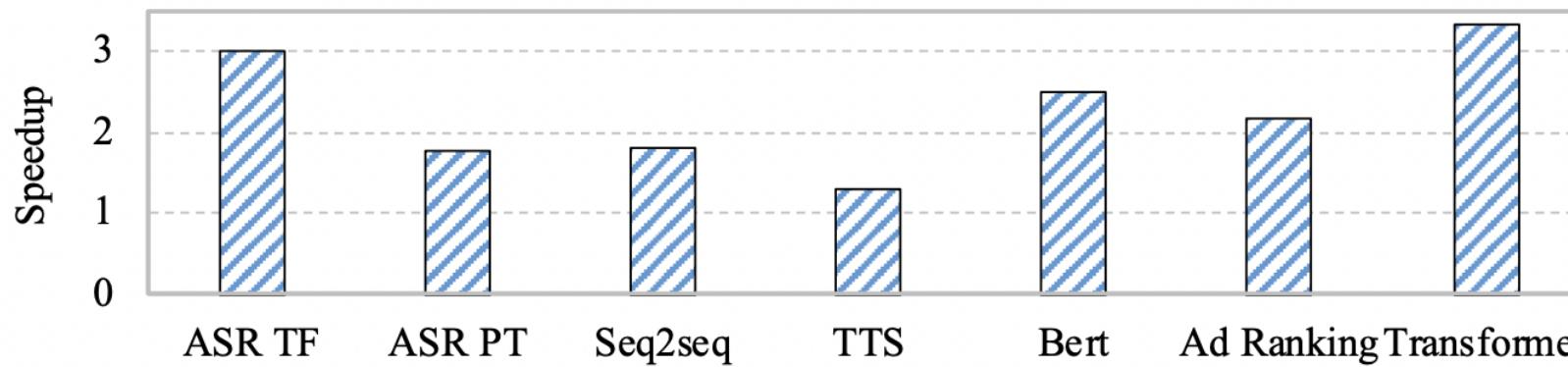
# 主要Pass Pipeline介绍

- 使用MLIR基础架构的好处
  - 模块化&灵活性
  - 可复用性&可扩展性
- 涉及的主要Dialects
  - DHLO Dialect
  - LDHLO Dialect
  - SCF Dialect
  - GPU Dialect



# 实验评估

- 与TensorFlow/PyTorch对比
  - 加速比最高可达3.35x，平均可达2.27x
  - 收益主要来源于自动Kernel融合
- 与Nimble对比，收益主要源于如下两点
  - DISC在访存密集型算子优化方面相较于Nimble具有2.61x的加速比
  - 更有效的Op融合
  - 编译期生成的运行时流相较于Nimble使用的VM解释执行，具有更低的运行时开销



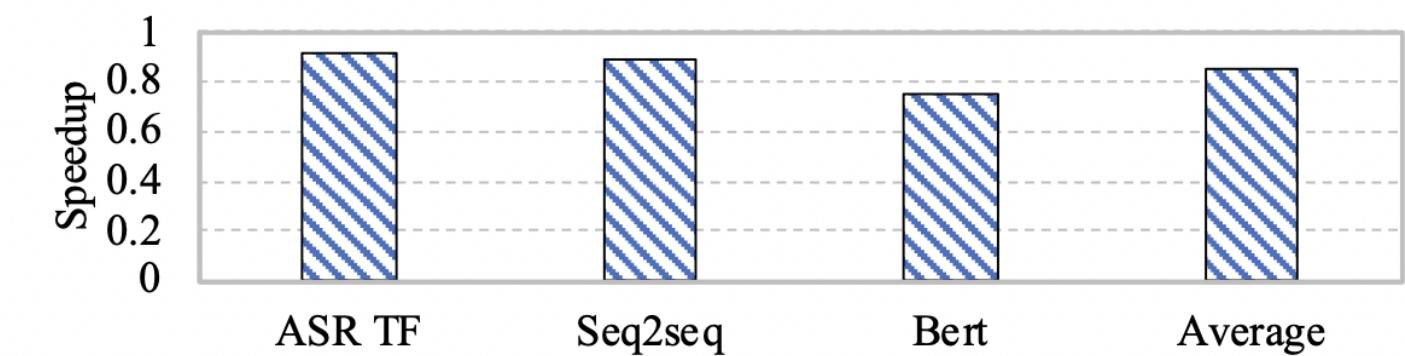
**Figure 3.** Speedup comparing with TensorFlow/PyTorch.

**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.

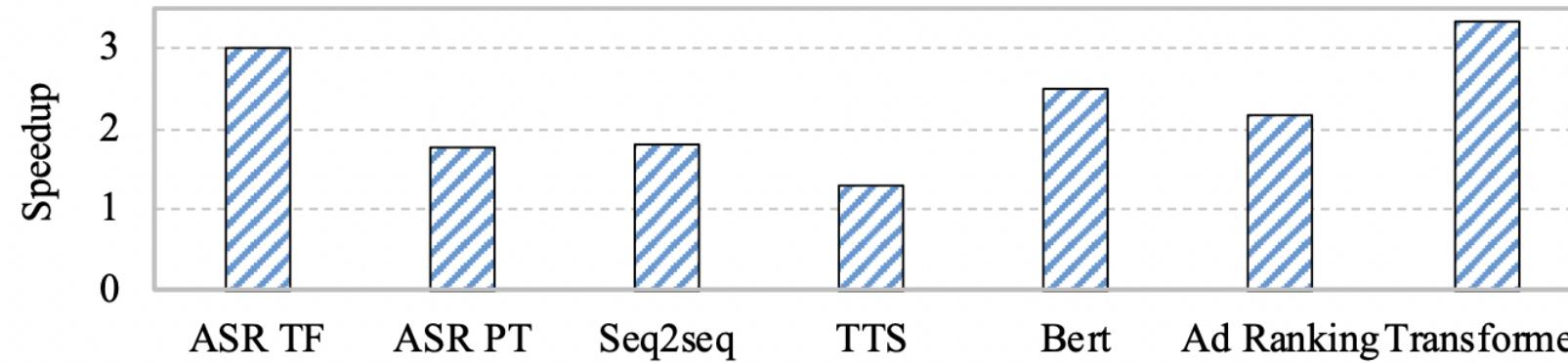
| Backend | Comp. bound | Mem. bound | Total |
|---------|-------------|------------|-------|
| Nimble  | 5232        | 8632       | 13924 |
| DISC    | 4476        | 6186       | 10734 |



**Figure 4.** Performance gap to static optimization.

# 实验评估

- 与静态shape优化之间的差距
  - 与静态shape优化相比，动态shape优化平均性能可达其85% (74.5% ~ 91.4%)
  - 静态shape可进行更加激进的图优化
  - 因为没有完整的shape信息，动态shape编译器将会丧失一些融合机会，这反应在融合策略、代码生成策略以及下标计算所需的指令数等方面



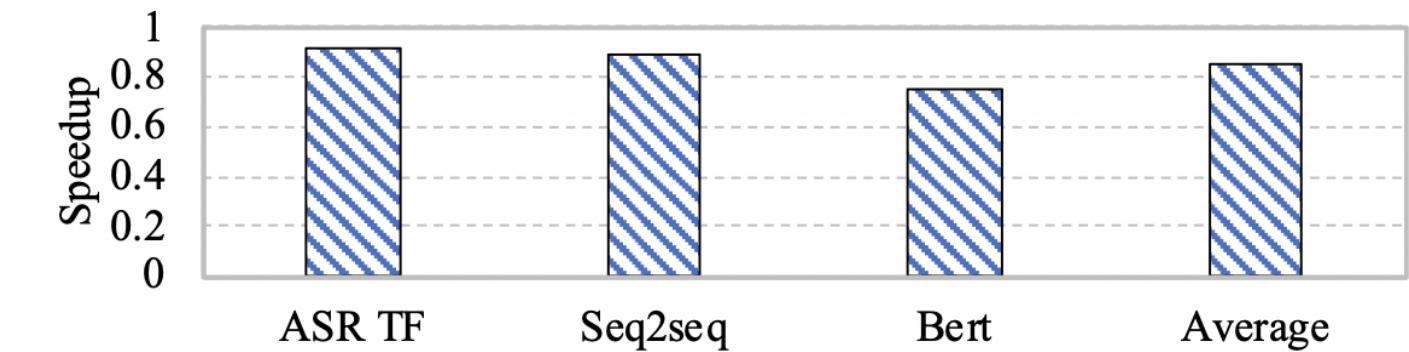
**Figure 3.** Speedup comparing with TensorFlow/PyTorch.

**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 |



**Figure 4.** Performance gap to static optimization.

# Thanks

[Learn More](#)

---