



中国科学院软件研究所
Institute of Software Chinese Academy of Sciences



智能软件研究中心
Intelligent Software Research Center

MLIR 的简介、实践、思考

张洪滨

2021.09.28

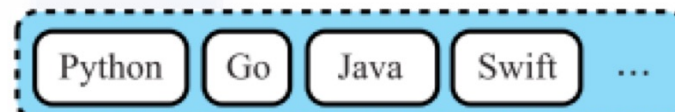
目录

1. MLIR 简介 – 深度学习框架的发展 | 什么是MLIR
2. MLIR 实践 – MLIR 社区工作 | buddy-mlir
3. MLIR 思考 – 基于 MLIR 的深度学习框架

1. MLIR 简介 – 深度学习框架的发展 | 什么是MLIR

MLIR 简介 – 深度学习框架的发展

多种语言绑定

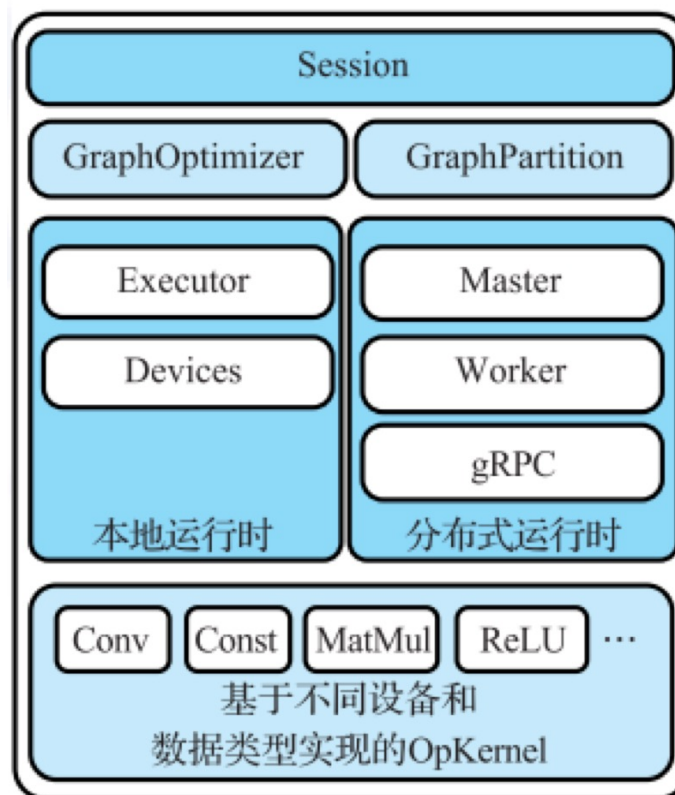


C/C++ 接口



TensorFlow后端部分

- 运行时部分
- 框架部分
- 计算图部分
- 图优化部分
- 计算核函数部分
- 计算节点部分



MLIR 简介 – 深度学习框架的发展

多种语言绑定

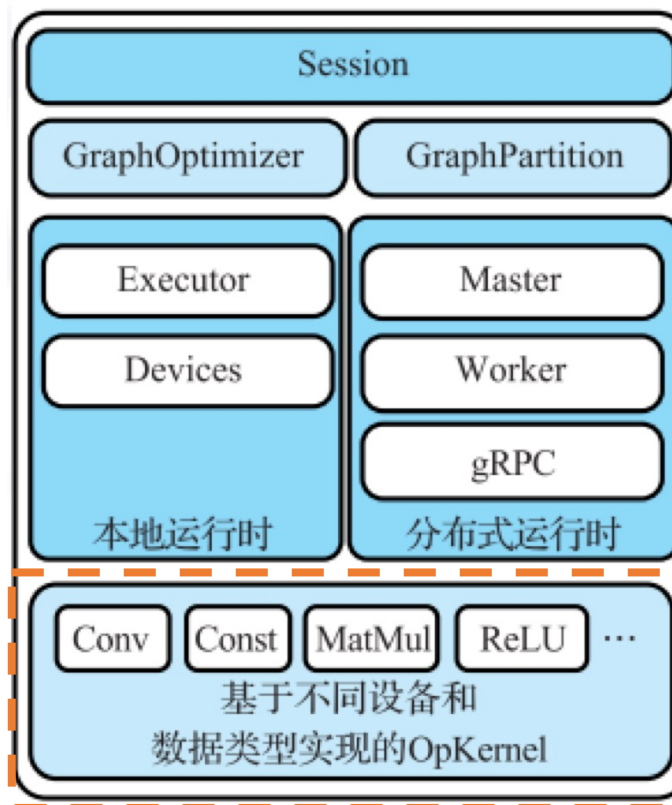


C/C++ 接口



TensorFlow后端部分

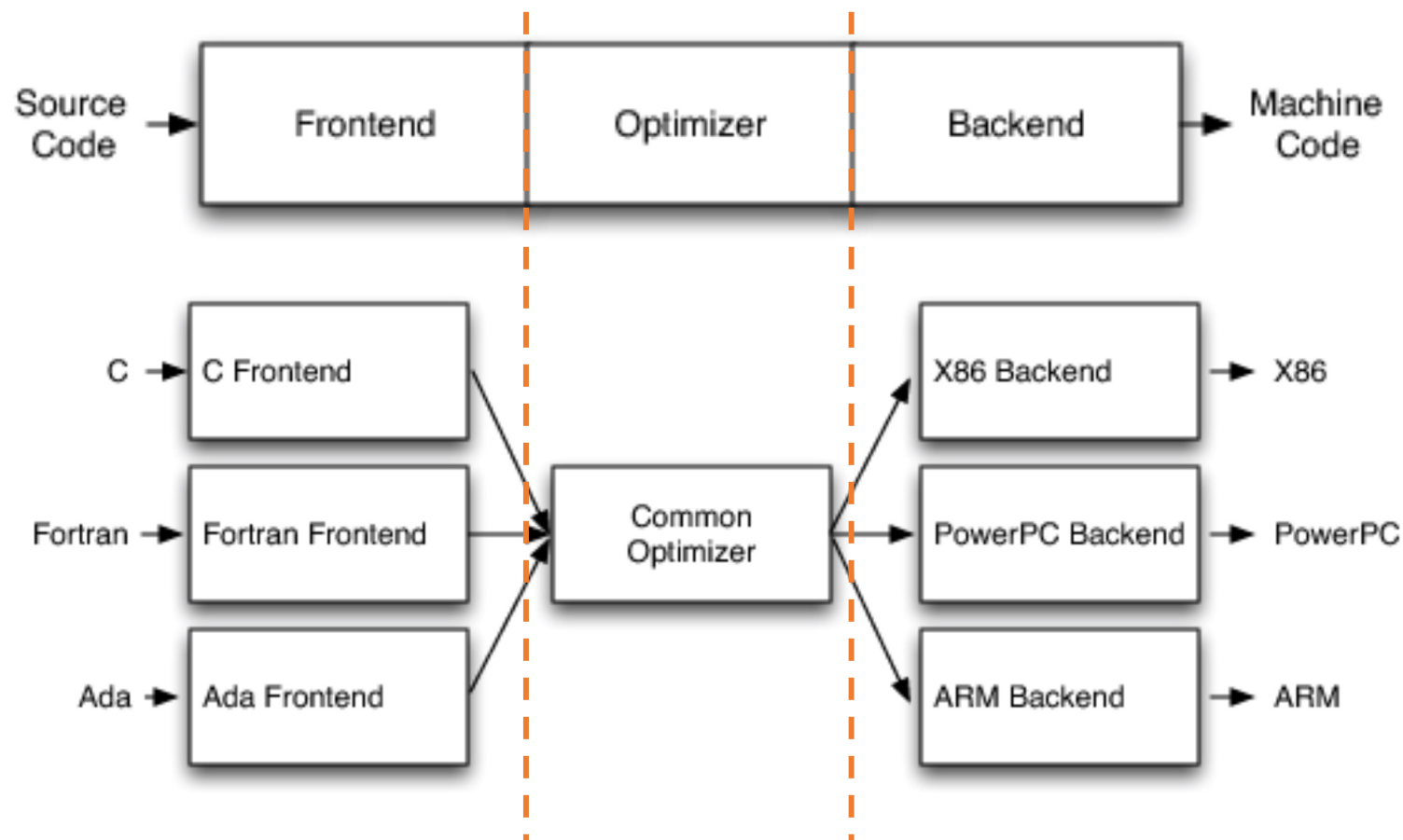
- 运行时部分
- 框架部分
- 计算图部分
- 图优化部分
- 计算核函数部分
- 计算节点部分



不同的底层硬件
OpKernel有不同的实现

- CPU
- GPU
- 深度学习处理器

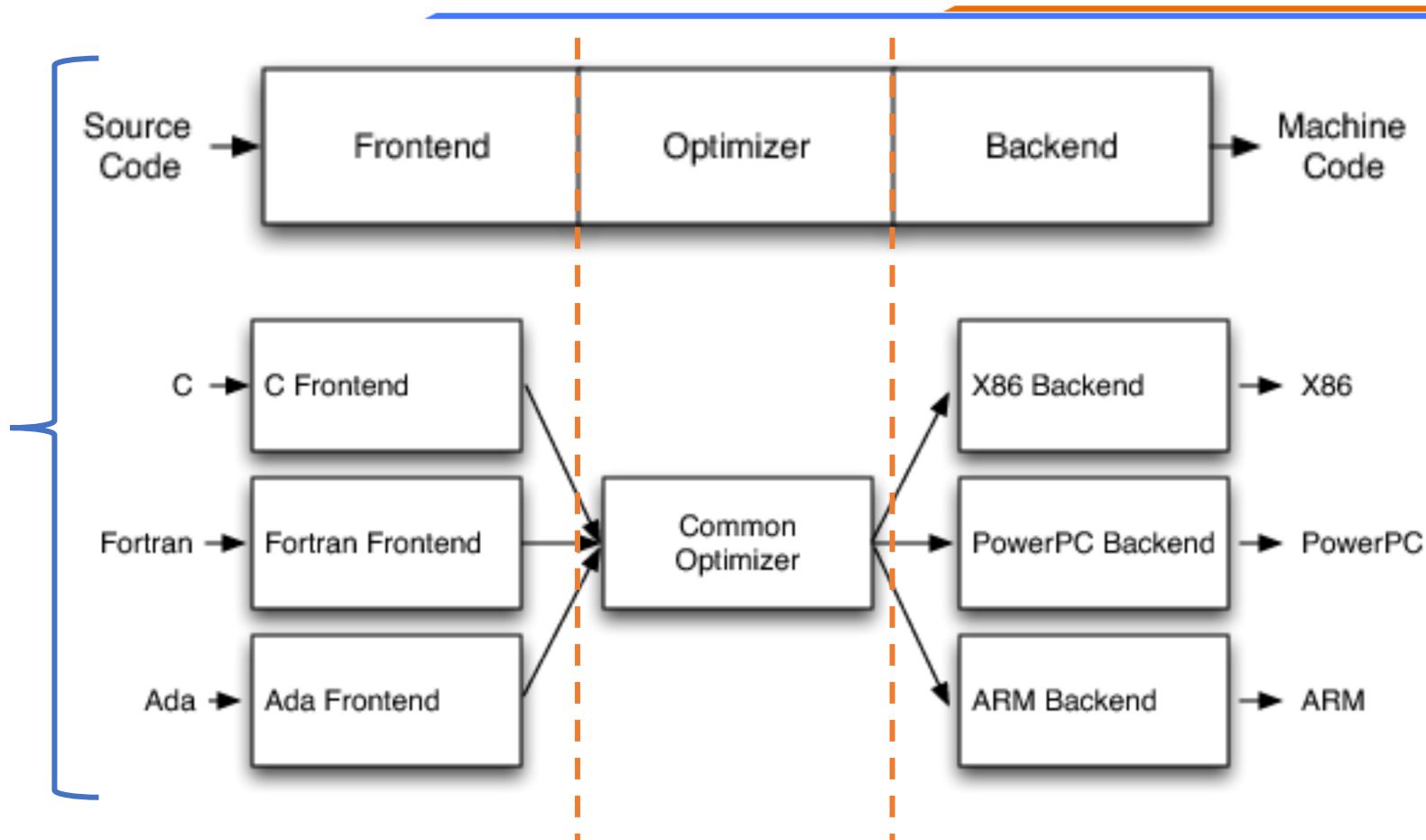
MLIR 简介 – 深度学习框架的发展



MLIR 简介 – 深度学习框架的发展

前端

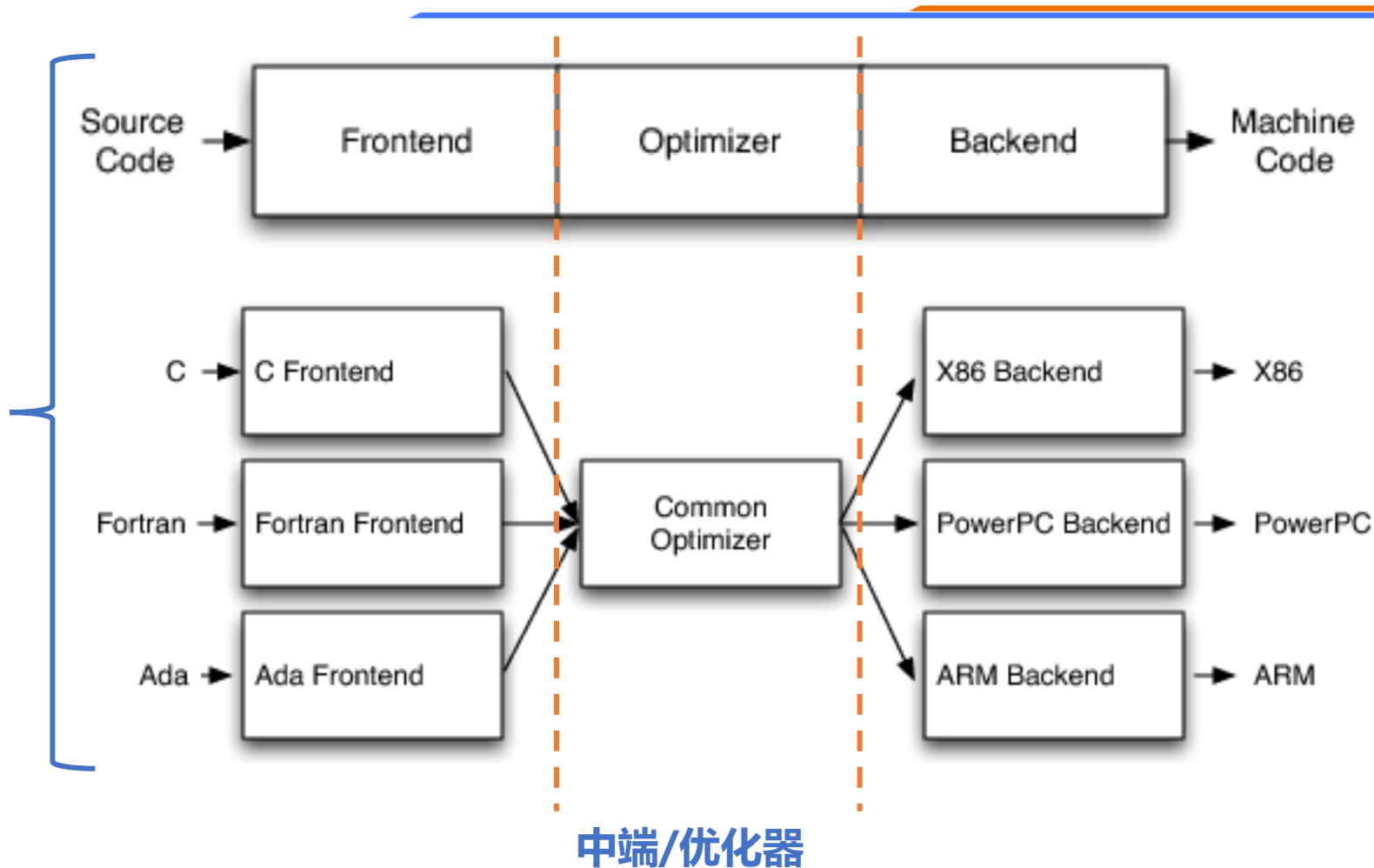
- 词法语法分析
- 构造抽象语法树
- 语义检查
- IR转换



MLIR 简介 – 深度学习框架的发展

前端

- 词法语法分析
- 构造抽象语法树
- 语义检查
- IR转换

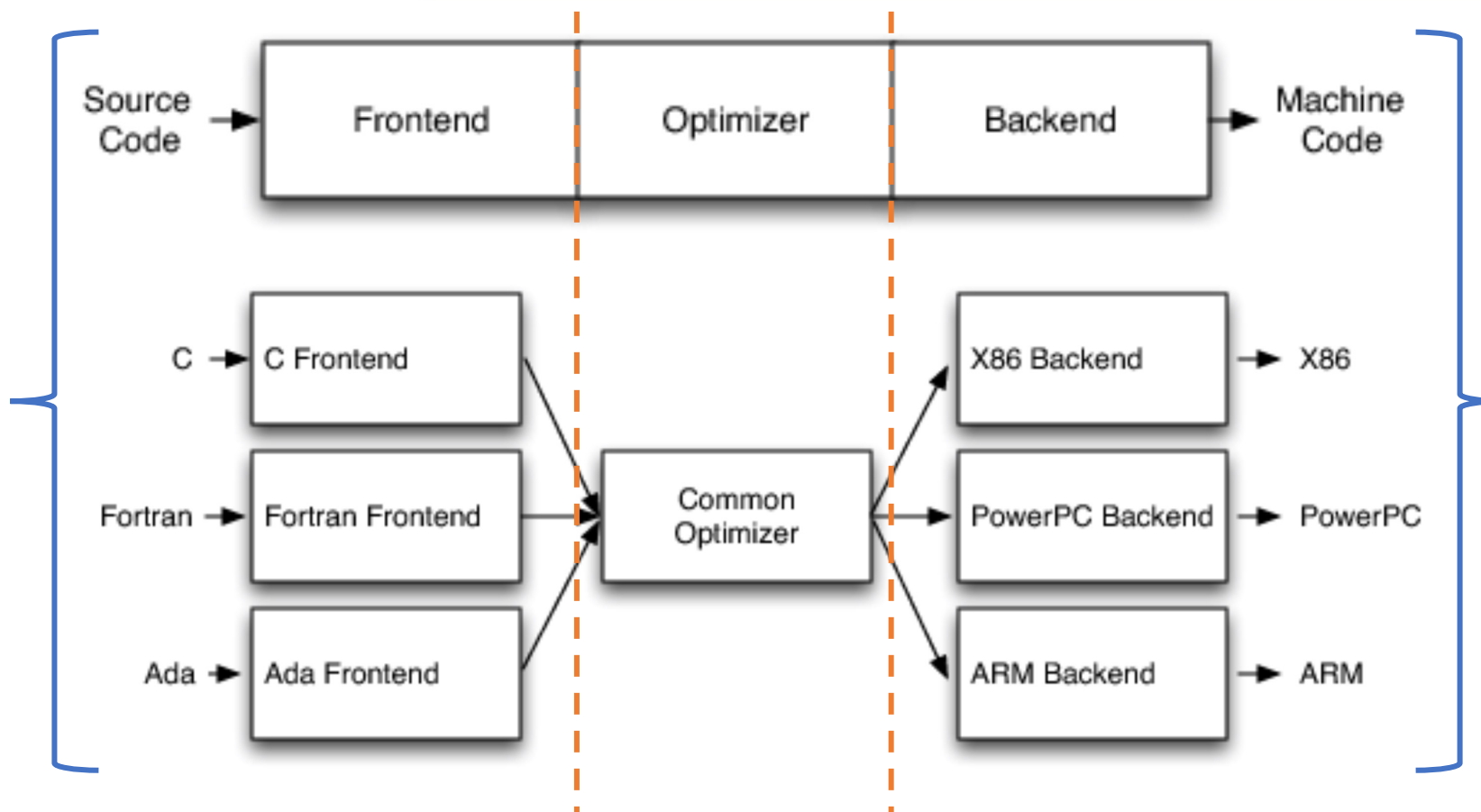


进行各种 IR 层级的优化

MLIR 简介 – 深度学习框架的发展

前端

- 词法语法分析
- 构造抽象语法树
- 语义检查
- IR转换



后端

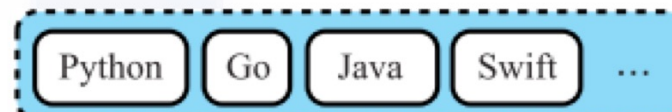
- 指令选择
- 寄存器分配
- 代码生成

中端/优化器

进行各种 IR 层级的优化

MLIR 简介 – 深度学习框架的发展

多种语言绑定

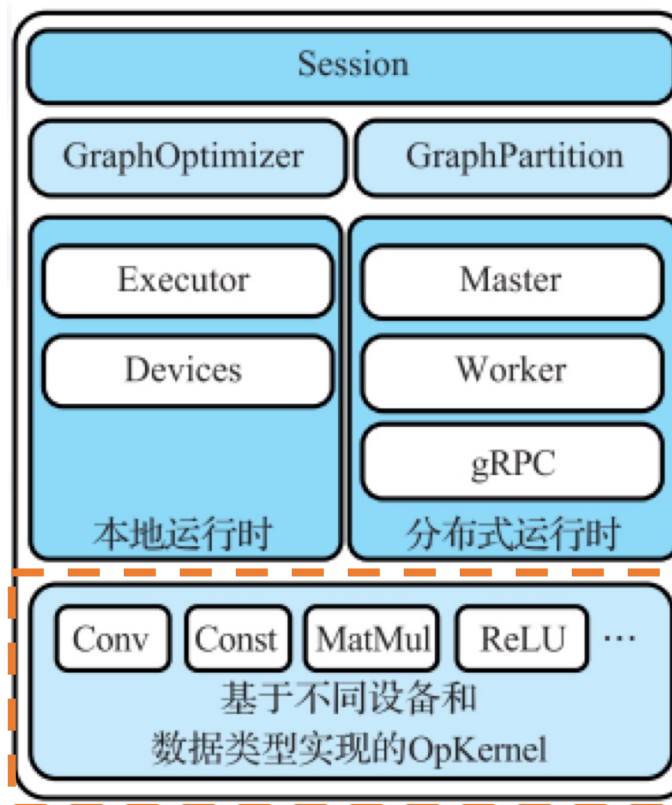


C/C++ 接口



TensorFlow后端部分

- 运行时部分
- 框架部分
- 计算图部分
- 图优化部分
- 计算核函数部分
- 计算节点部分

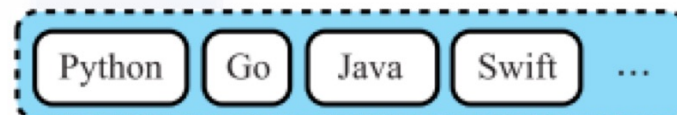


不同的底层硬件
OpKernel有不同的实现

- CPU
- GPU
- 深度学习处理器

MLIR 简介 – 深度学习框架的发展

多种语言绑定

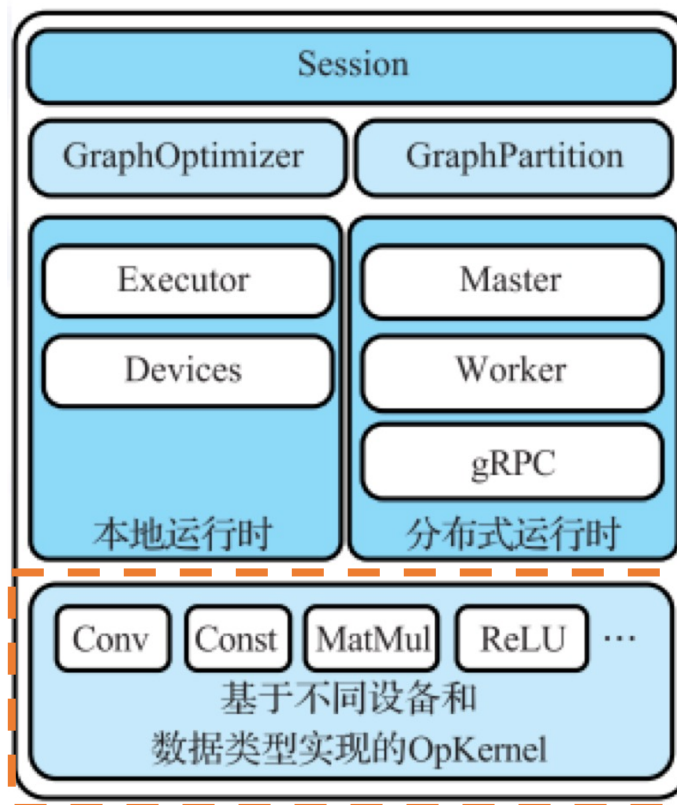


C/C++ 接口



TensorFlow后端部分

- 运行时部分
- 框架部分
- 计算图部分
- 图优化部分
- 计算核函数部分
- 计算节点部分



不同的底层硬件
OpKernel有不同的实现

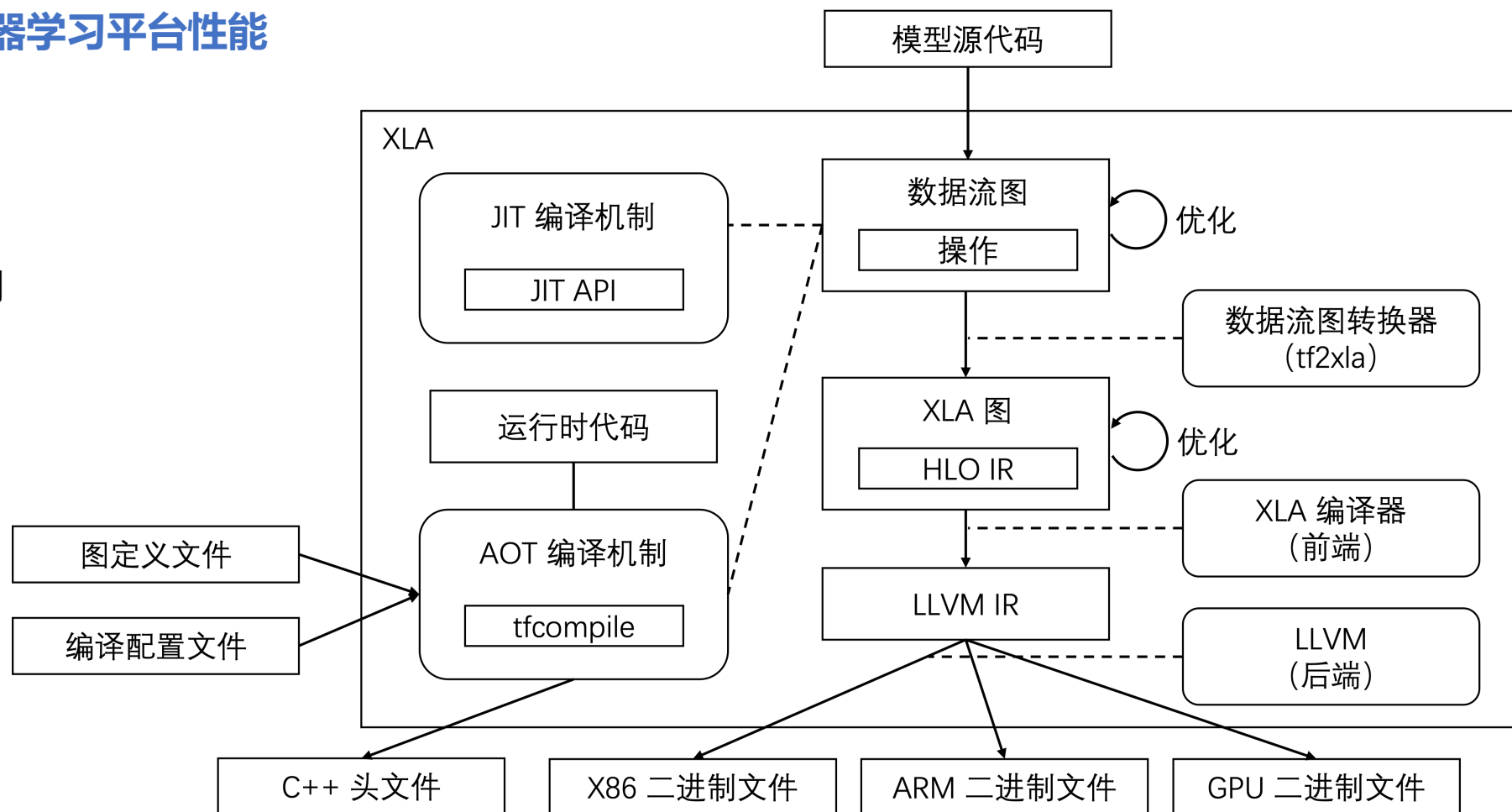
- CPU
- GPU
- 深度学习处理器

XLA 线性代数计算优化机制
引入中间表示层 (IR) 和编译器

MLIR 简介 – 深度学习框架的发展

通过编译技术优化机器学习平台性能

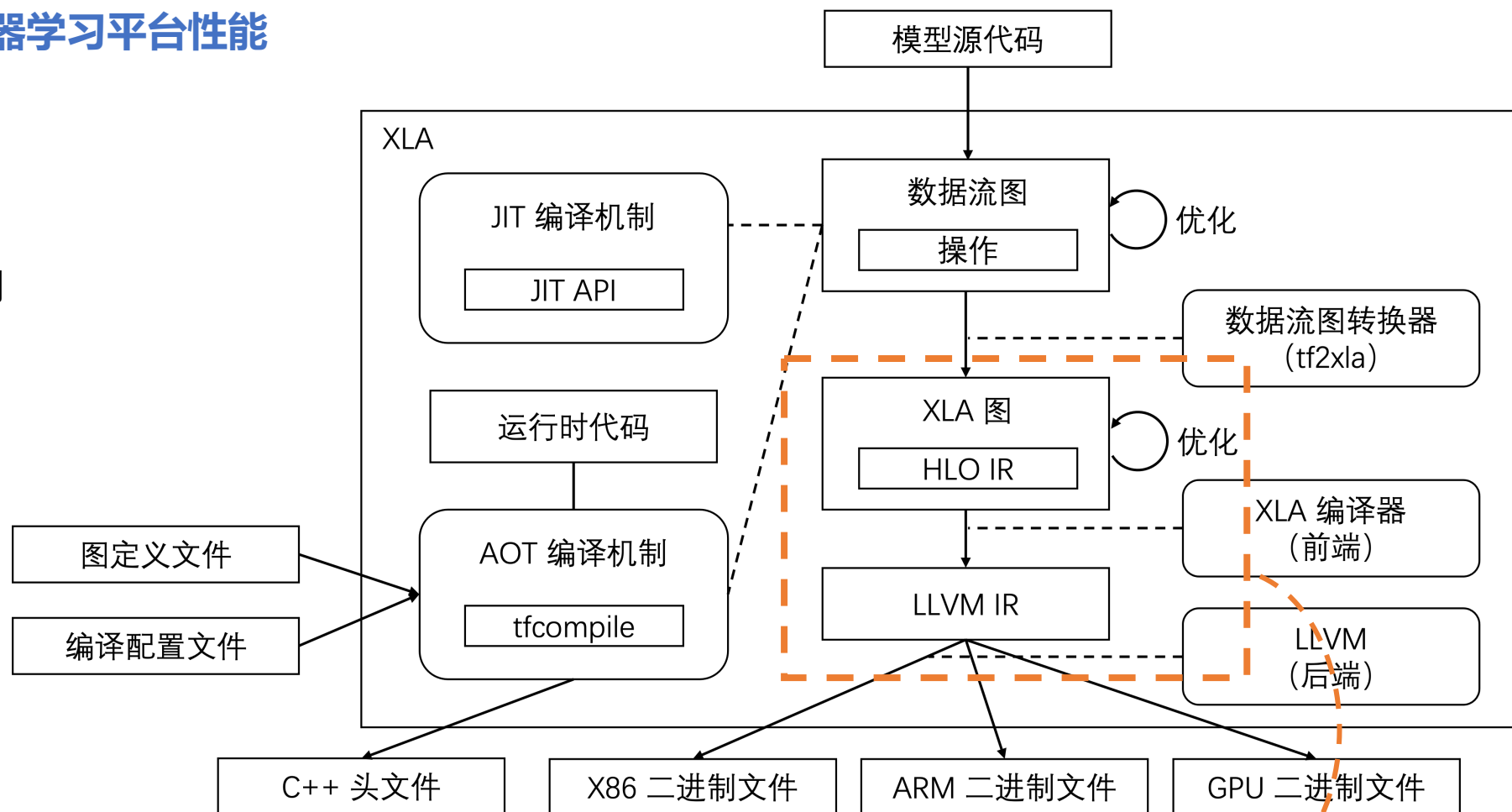
- 加速数据流图执行
- 提升内存使用效率
- 降低自定义操作依赖
- 减小移动应用内存占用
- 增加平台可移植性



MLIR 简介 – 深度学习框架的发展

通过编译技术优化机器学习平台性能

- 加速数据流图执行
- 提升内存使用效率
- 降低自定义操作依赖
- 减小移动应用内存占用
- 增加平台可移植性



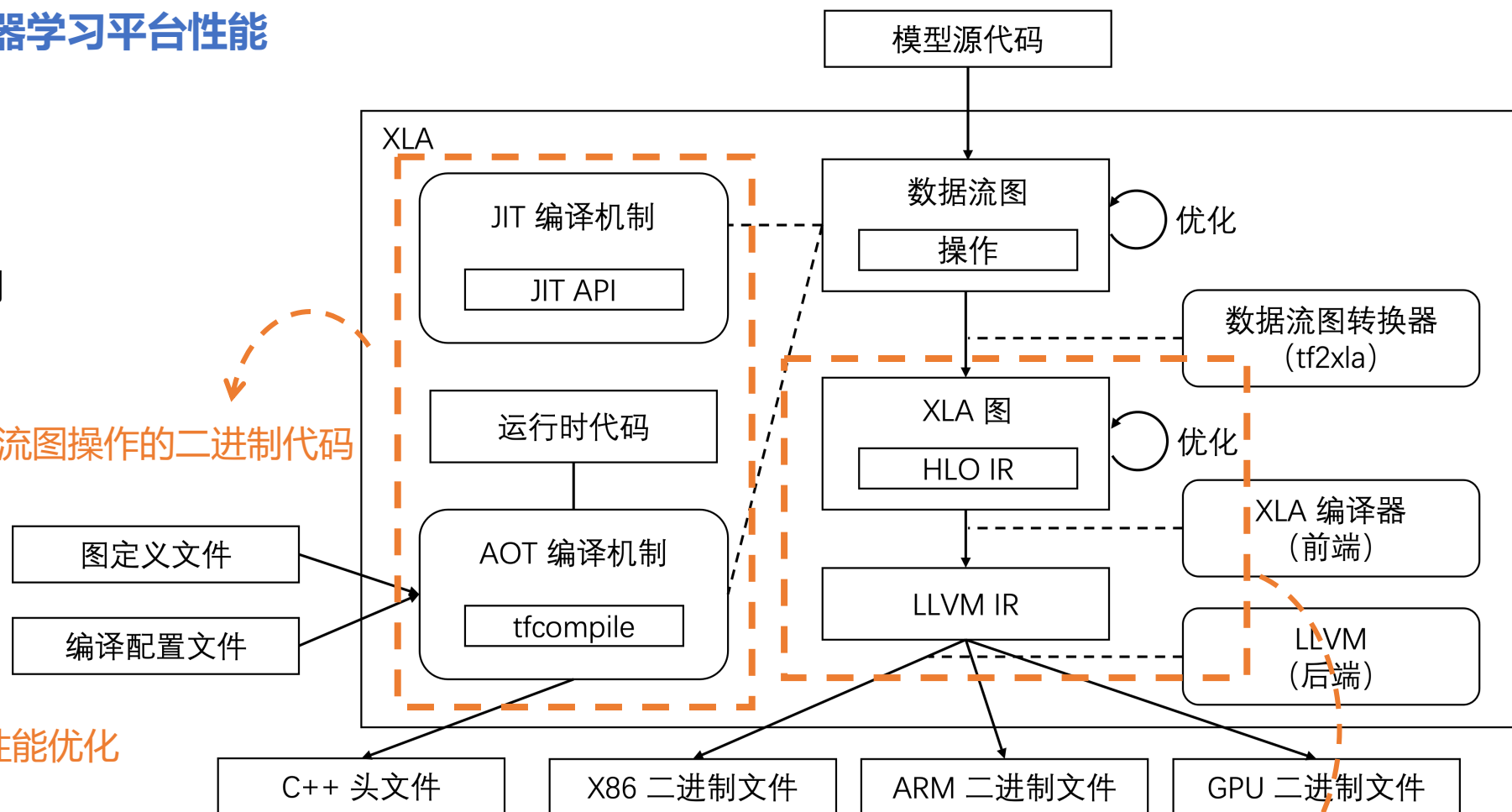
MLIR 简介 – 深度学习框架的发展

通过编译技术优化机器学习平台性能

- 加速数据流图执行
- 提升内存使用效率
- 降低自定义操作依赖
- 减小移动应用内存占用
- 增加平台可移植性

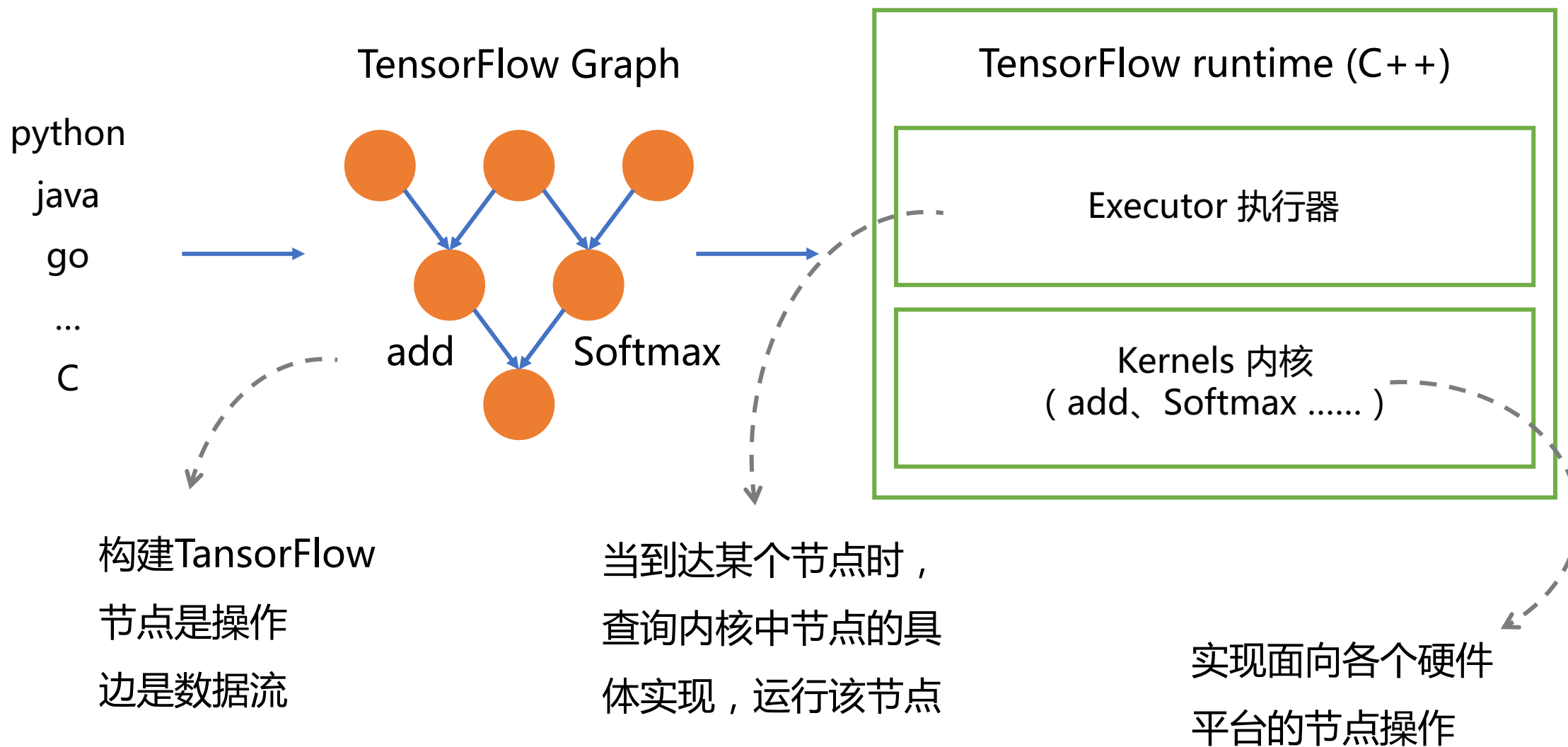
JIT 用于运行时创建执行数据流图操作的二进制代码
面向异构设备进行性能优化

AOT 在运行前创建并集成了
模型和运行时的二进制代码
适用于移动端的推理过程的性能优化



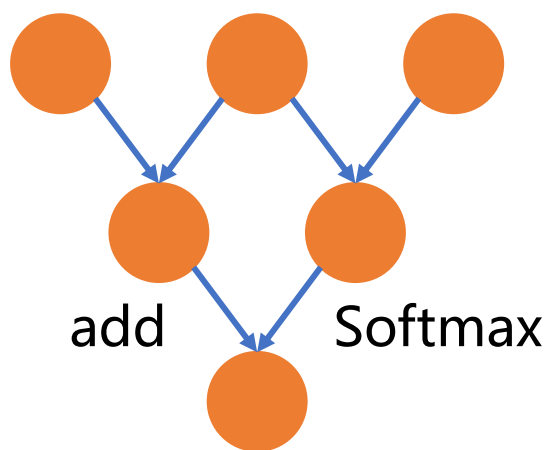
HLO的引入将核函数开发时的前后端代码解耦，增强可移植性

MLIR 简介 – TensorFlow执行方式



MLIR 简介 – 使用XLA加速TensorFlow

TensorFlow Graph

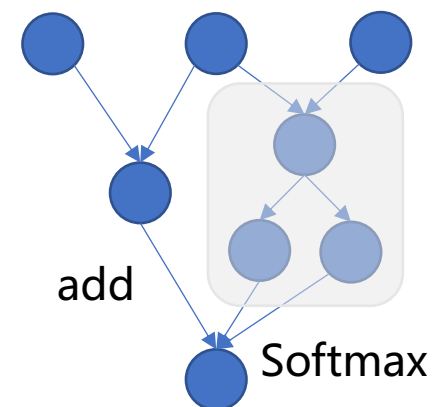


TensorFlow runtime (C++)

Executor 执行器

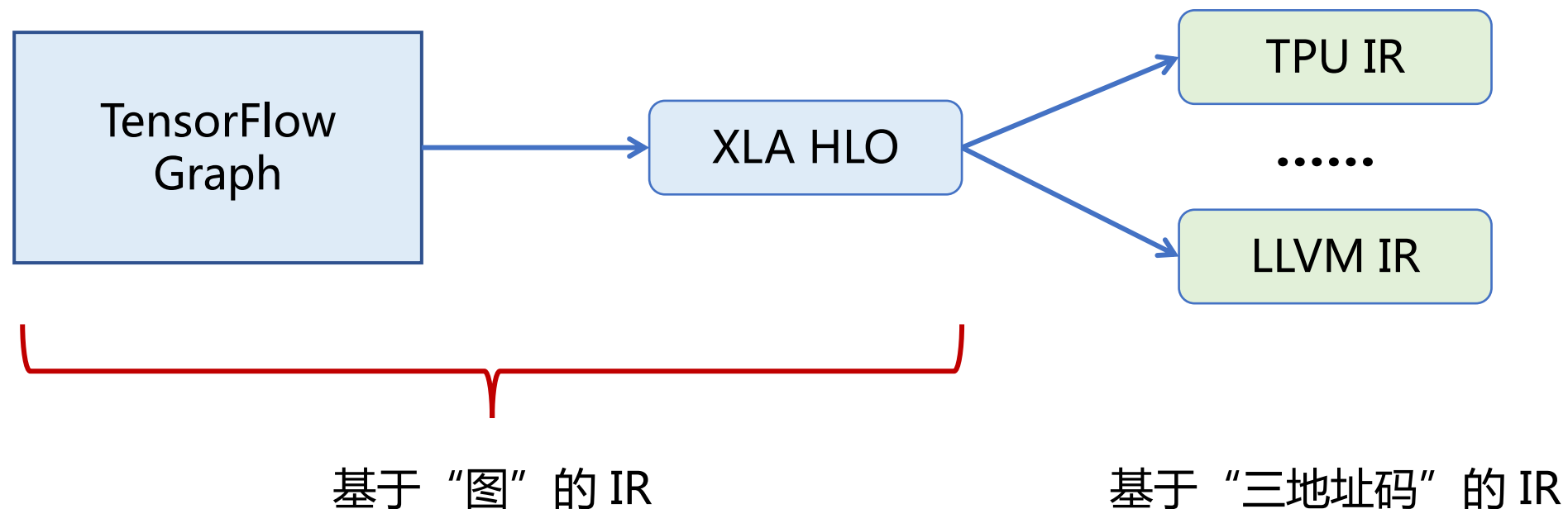
tf2xla 内核
(add、
Softmax)

XLA Graph



LLVM IR

MLIR 简介 – 存在什么问题



- 组合爆炸，很多组件无法重用
- 各个层次内部优化无法迁移
- 从XLA HLO到LLVM IR跨度太大，实现开销大

不是Machine Learning , 但为Machine Learning而生

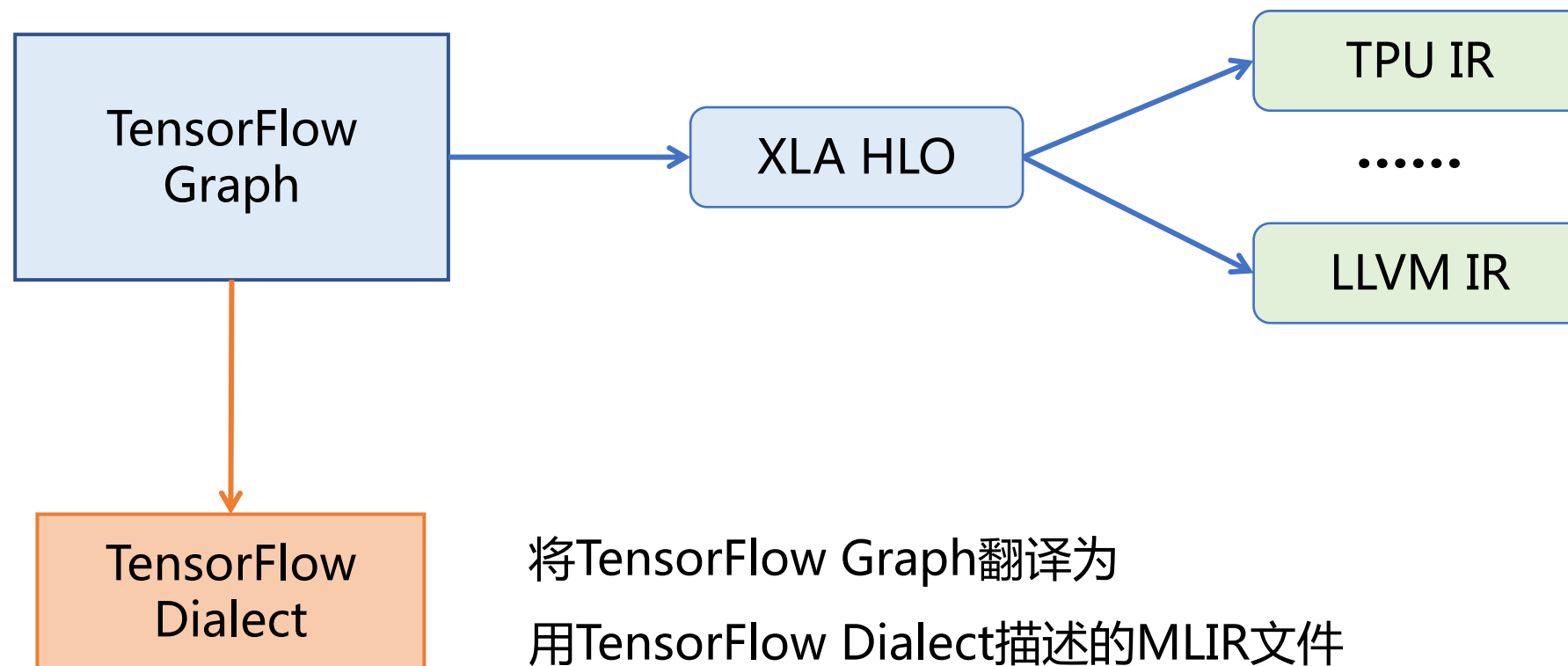
MLIR (Multi-Level Intermediate Representation)

The MLIR project is a novel approach to building **reusable** and **extensible** compiler infrastructure. MLIR aims to **address software fragmentation**, **improve compilation for heterogeneous hardware**, significantly **reduce the cost of building domain specific compilers**, and **aid in connecting existing compilers together**.

一个可重用、可扩展的
开源**编译基础框架**

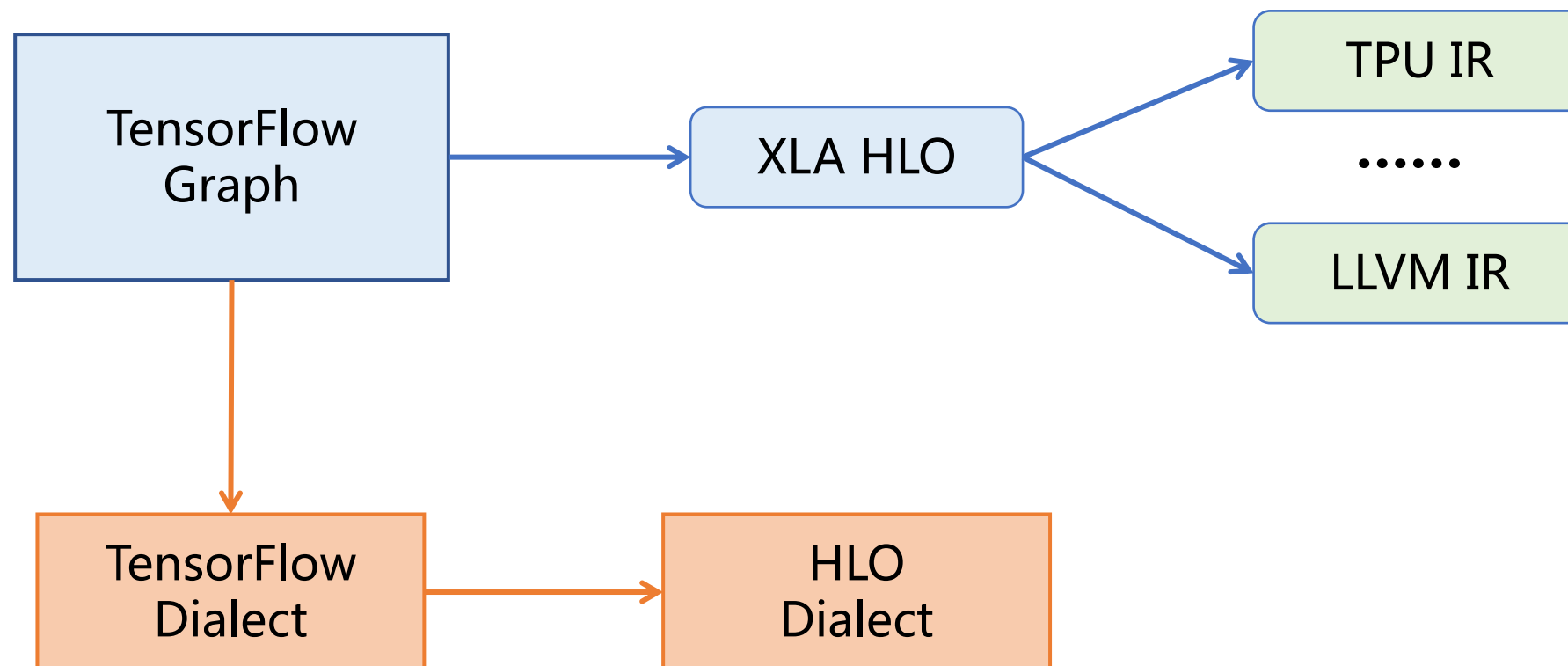
- 处理软件的碎片化
- 为面向异构硬件的编译提供支持
- 为领域专用编译器的开发减少开销
- 连接已有的各种编译器

MLIR 简介 – MLIR如何解决问题



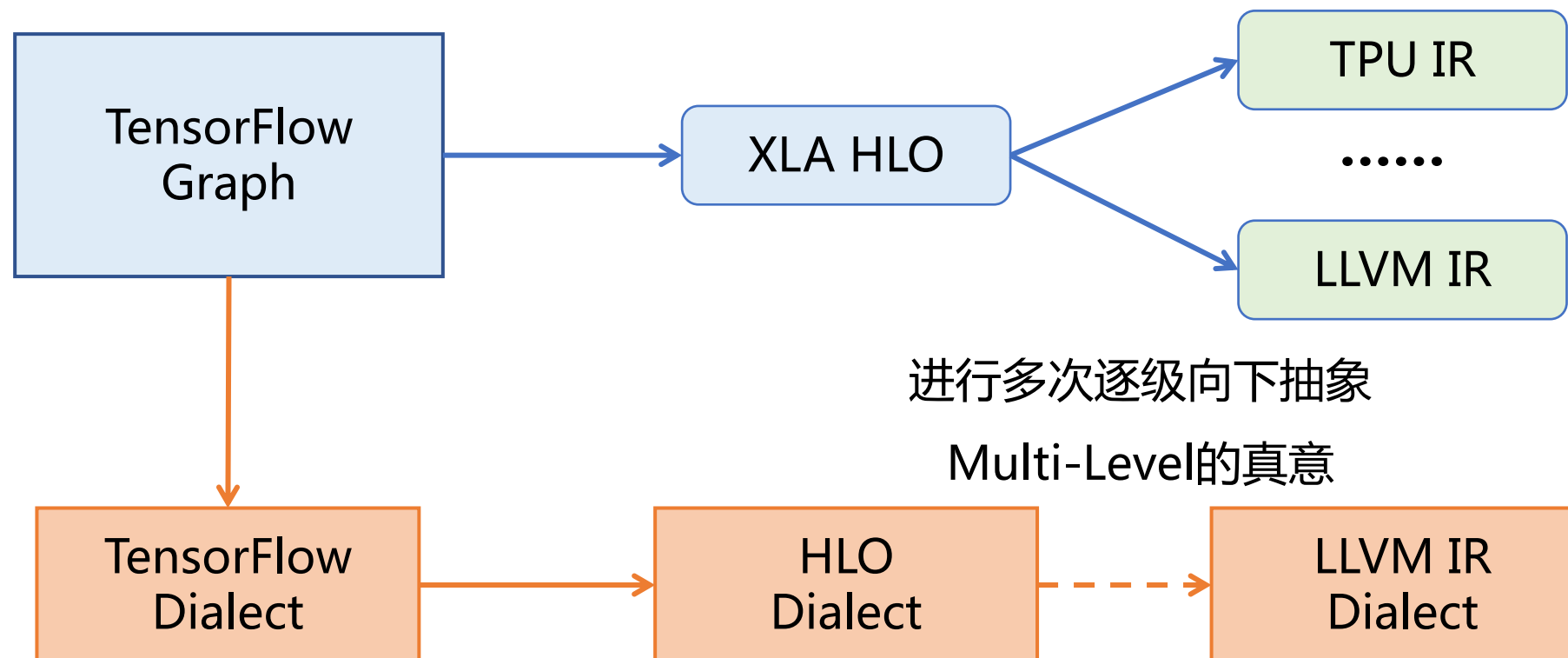
- 组合爆炸，很多组件无法重用
- 各个层次内部优化无法迁移
- 从XLA HLO到LLVM IR跨度太大，实现开销大

MLIR 简介 – MLIR如何解决问题



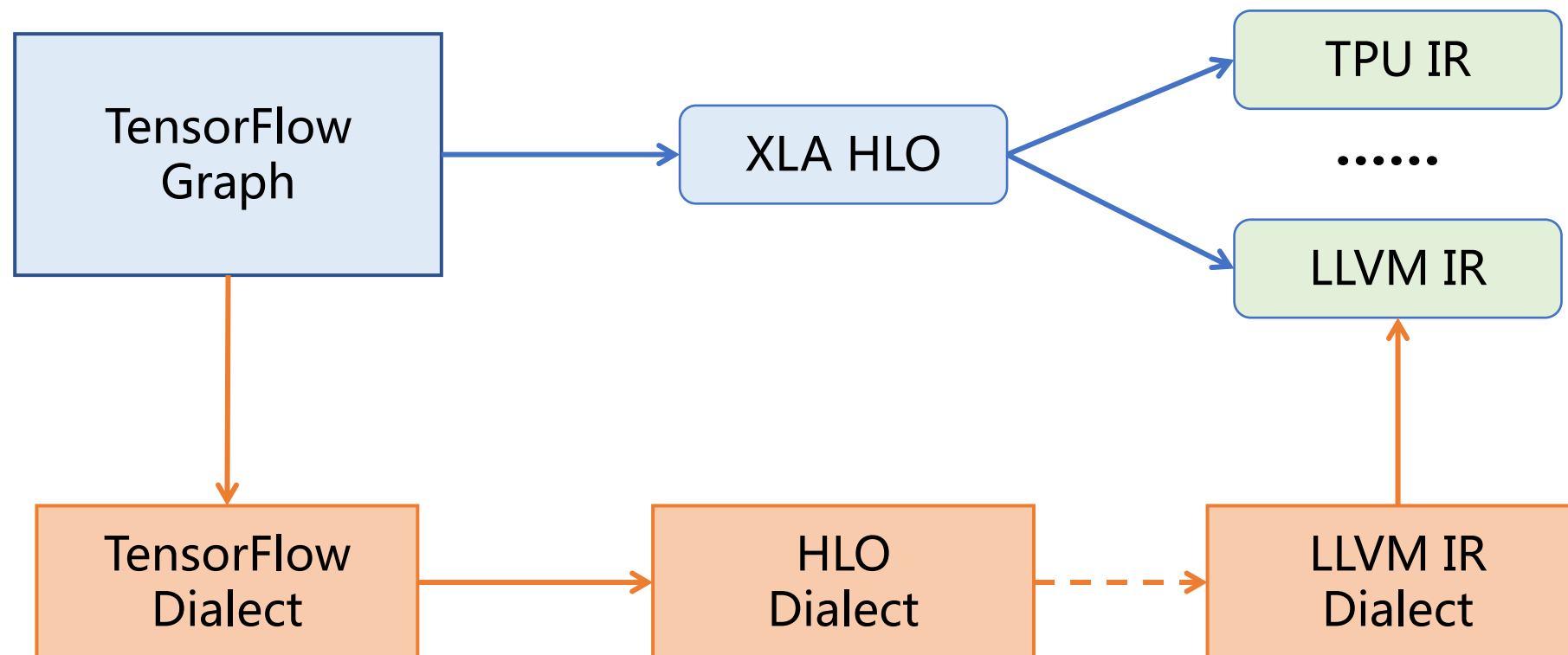
- 组合爆炸，很多组件无法重用
- 各个层次内部优化无法迁移
- 从XLA HLO到LLVM IR跨度太大，实现开销大

MLIR 简介 – MLIR如何解决问题



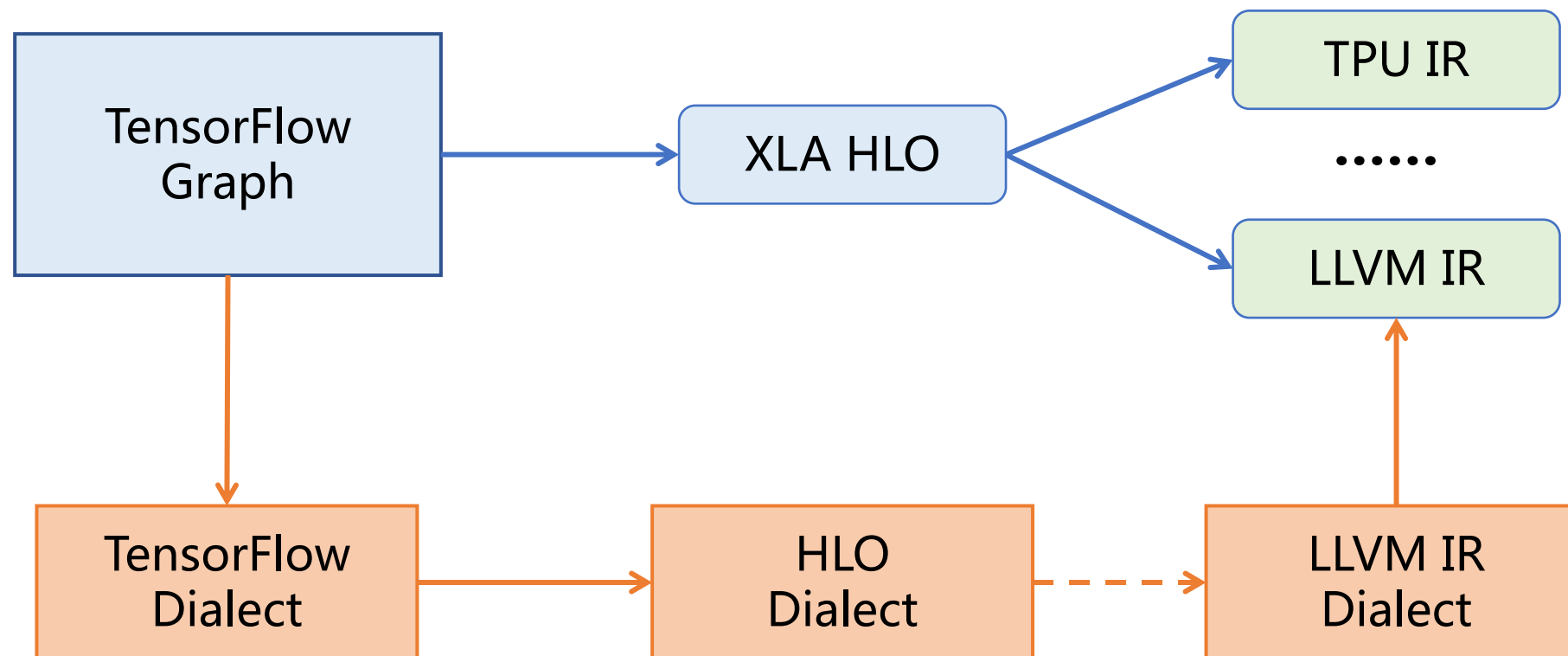
- 组合爆炸，很多组件无法重用
- 各个层次内部优化无法迁移
- 从XLA HLO到LLVM IR跨度太大，实现开销大

MLIR 简介 – MLIR如何解决问题



- 组合爆炸，很多组件无法重用
- 各个层次内部优化无法迁移
- 从XLA HLO到LLVM IR跨度太大，实现开销大

MLIR 简介 – MLIR如何解决问题



- 组合爆炸，很多组件无法重用
- 各个层次内部优化无法迁移
- 从XLA HLO到LLVM IR跨度太大，实现开销大

语法统一，组件可以重用
共享生态，各个层次可以协调优化
层级之间跨度小，方便实现

2. MLIR 实践 – MLIR 社区工作 | buddy-mlir

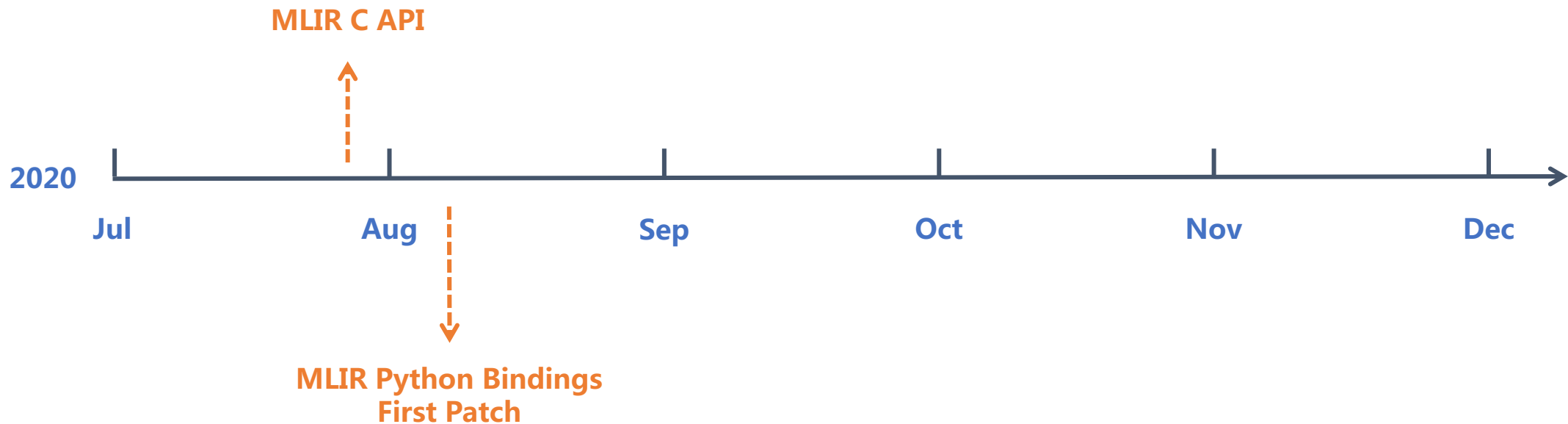
MLIR 实践 – MLIR Python Bindings



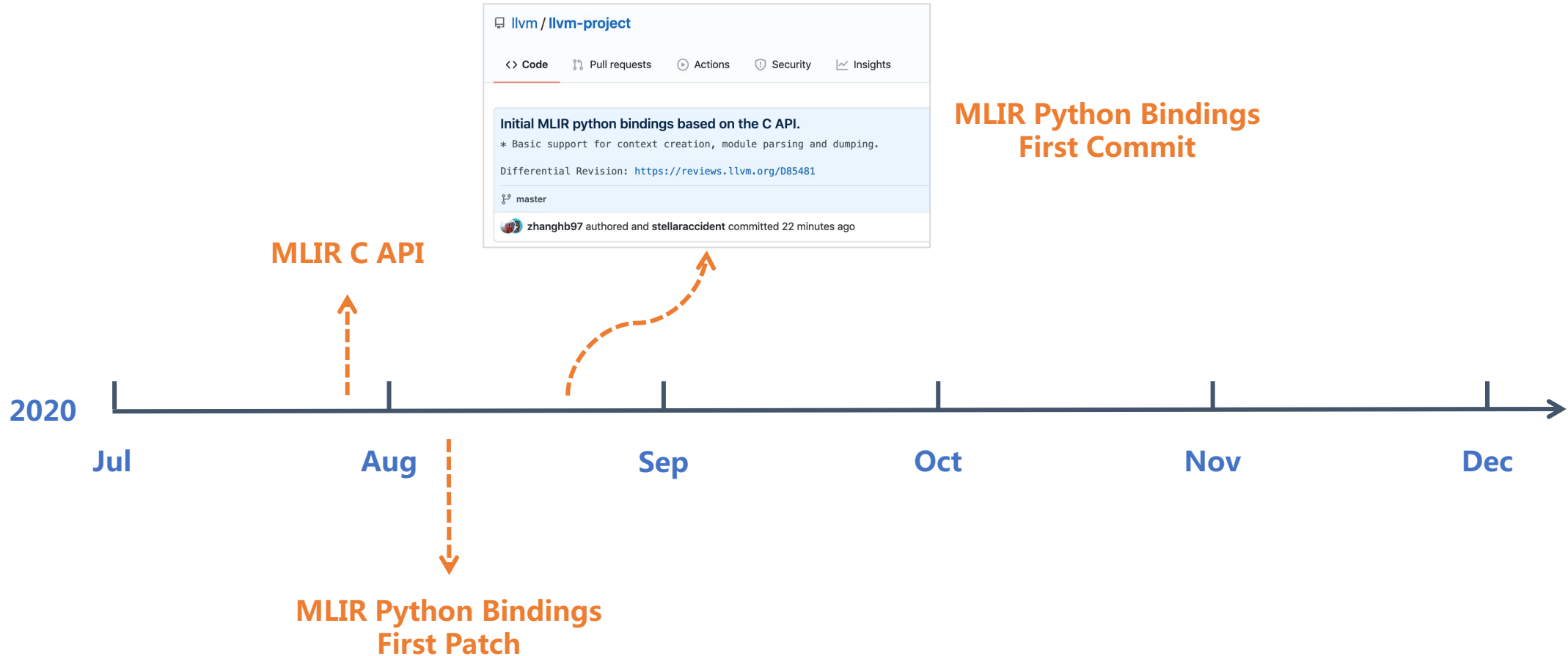
MLIR 实践 – MLIR Python Bindings



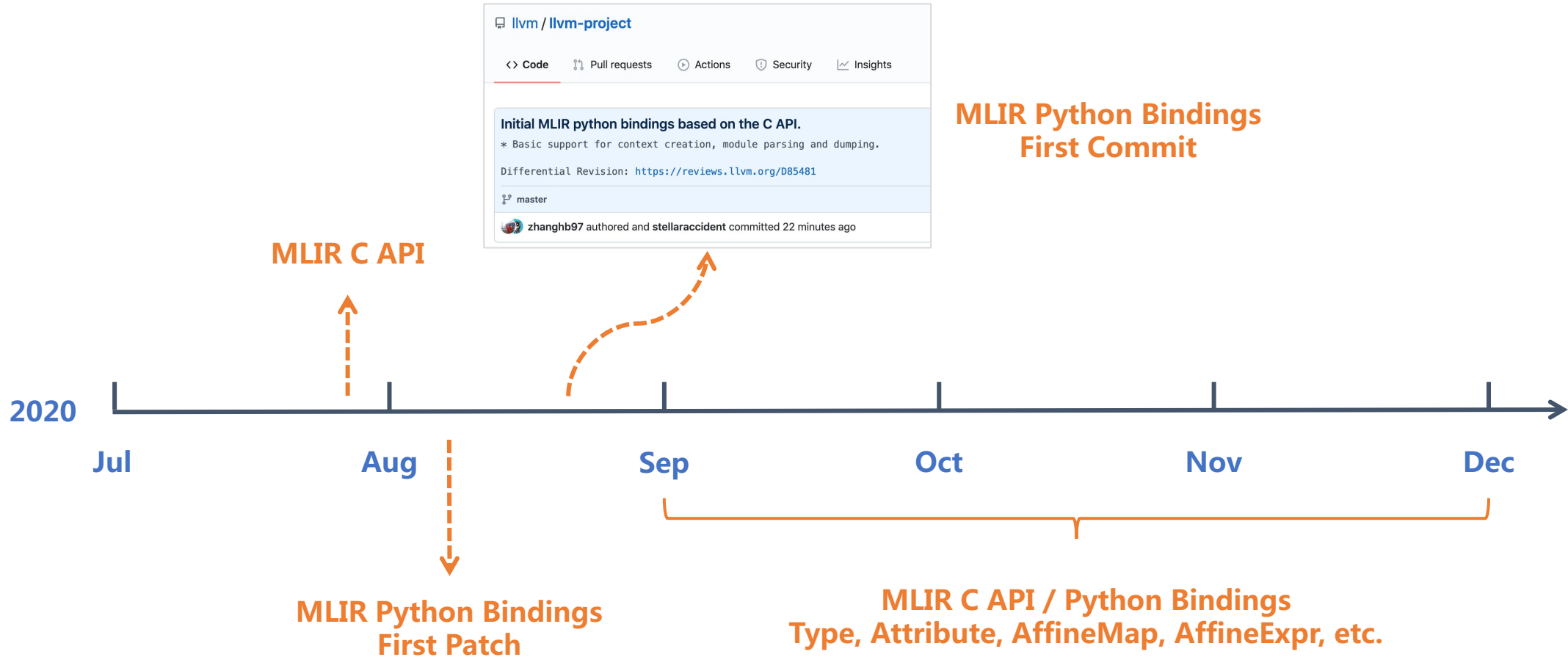
MLIR 实践 – MLIR Python Bindings



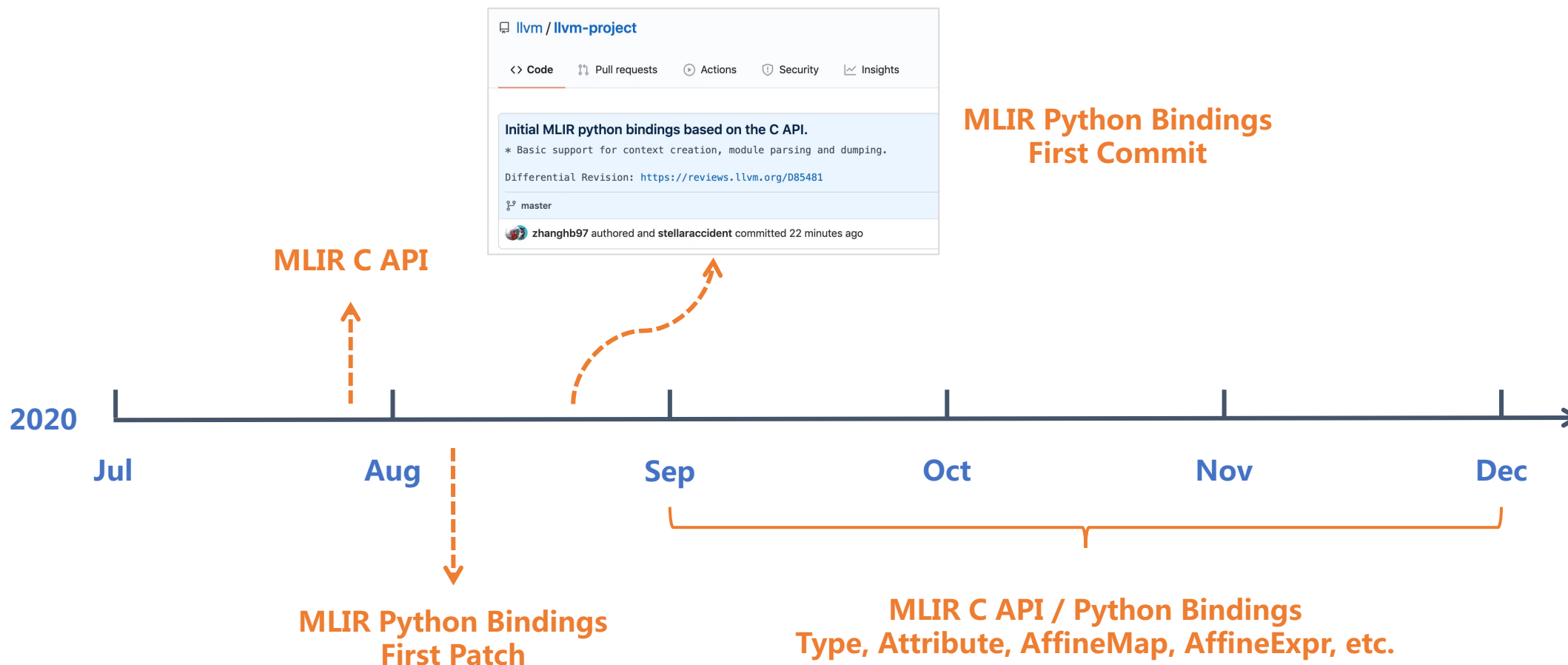
MLIR 实践 – MLIR Python Bindings



MLIR 实践 – MLIR Python Bindings



MLIR 实践 – MLIR Python Bindings

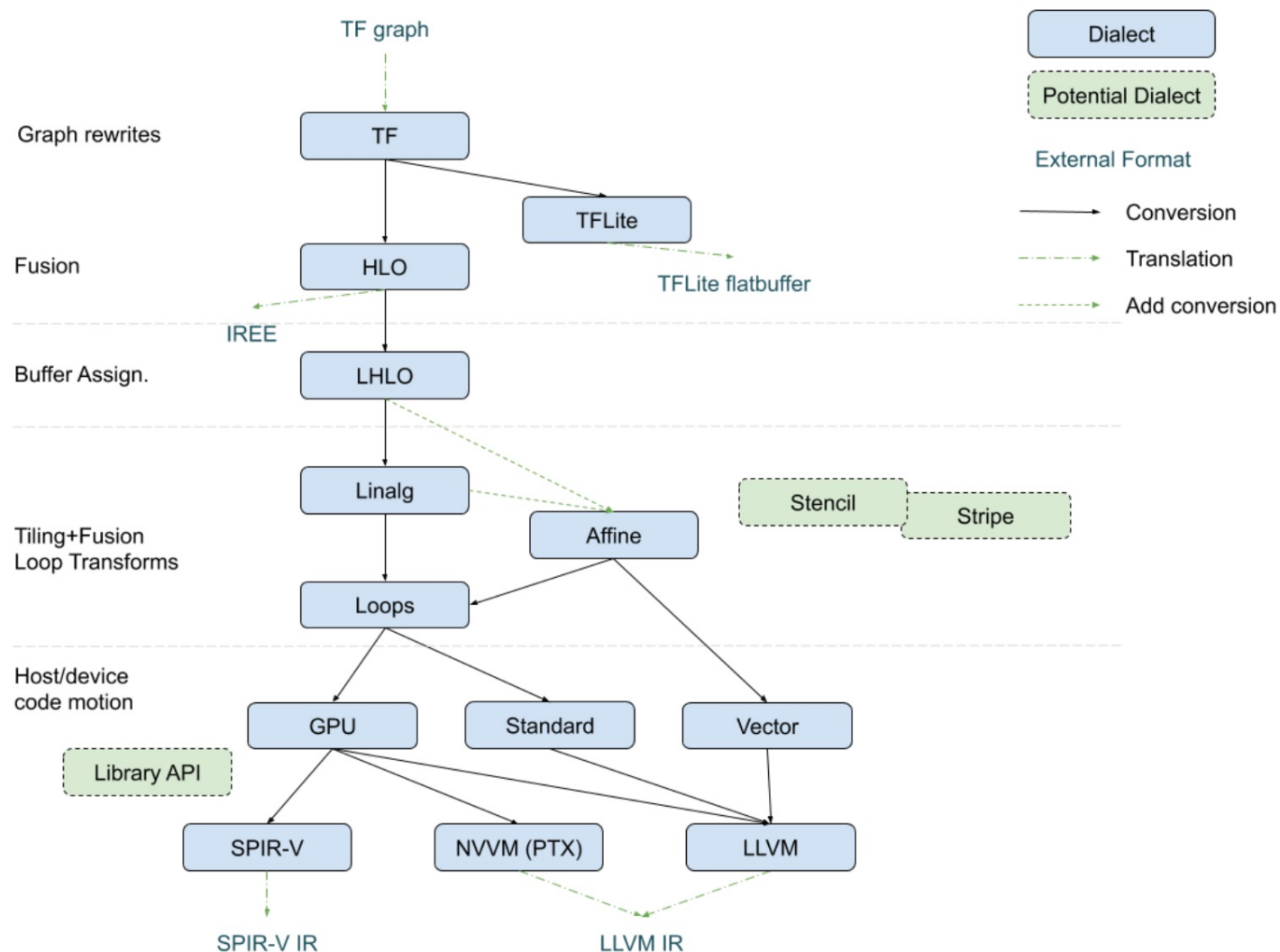


使用/扩展 MLIR Python Bindings 的项目：IREE，MHLO，Torch-MLIR，CIRCT，etc.

MLIR 实践 – MLIR RISC-V Vector Dialect

Main Transformations

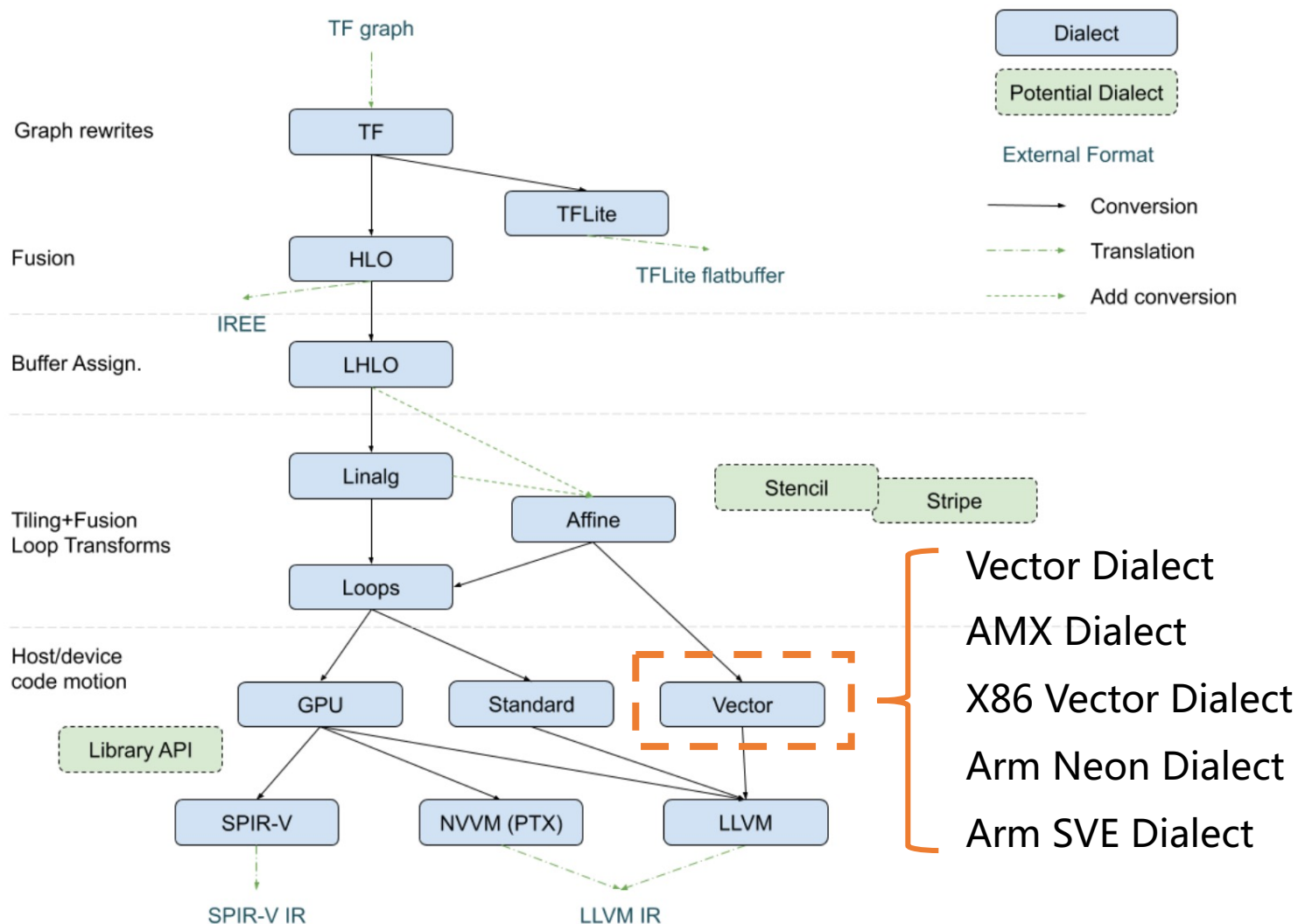
Abstractions / Dialects



MLIR 实践 – MLIR RISC-V Vector Dialect

Main Transformations

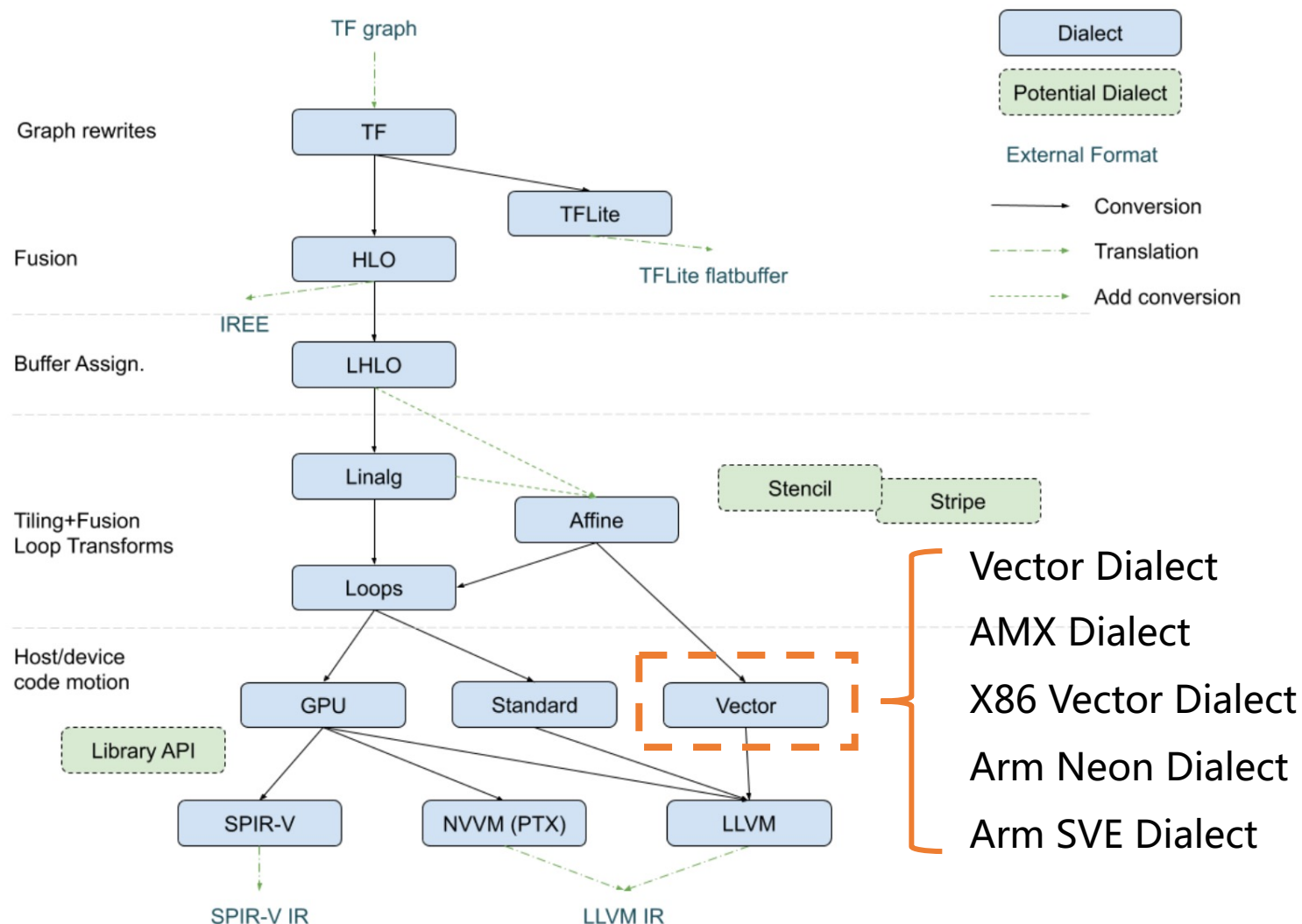
Abstractions / Dialects



MLIR 实践 – MLIR RISC-V Vector Dialect

Main Transformations

Abstractions / Dialects



MLIR RISC-V Vector Dialect

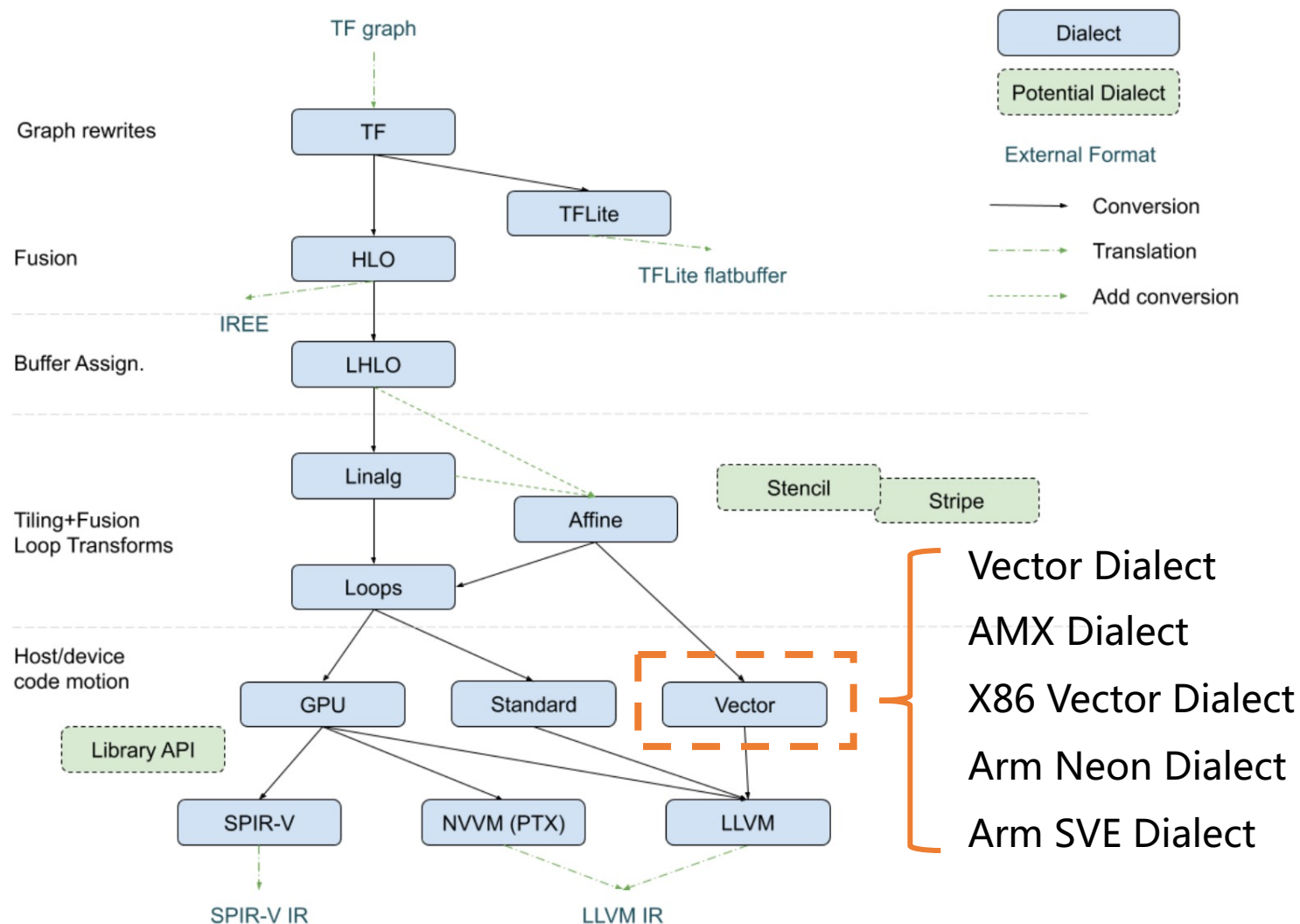
- Operation
 - RVV Operation
 - RVV Intrinsic Operation
- Type
 - Scalable Vector Type
- Conversion/Translation
 - RVV Dialect
 - LLVM Dialect
 - LLVM IR



MLIR 实践 – MLIR RISC-V Vector Dialect

Main Transformations

Abstractions / Dialects



MLIR RISC-V Vector Dialect

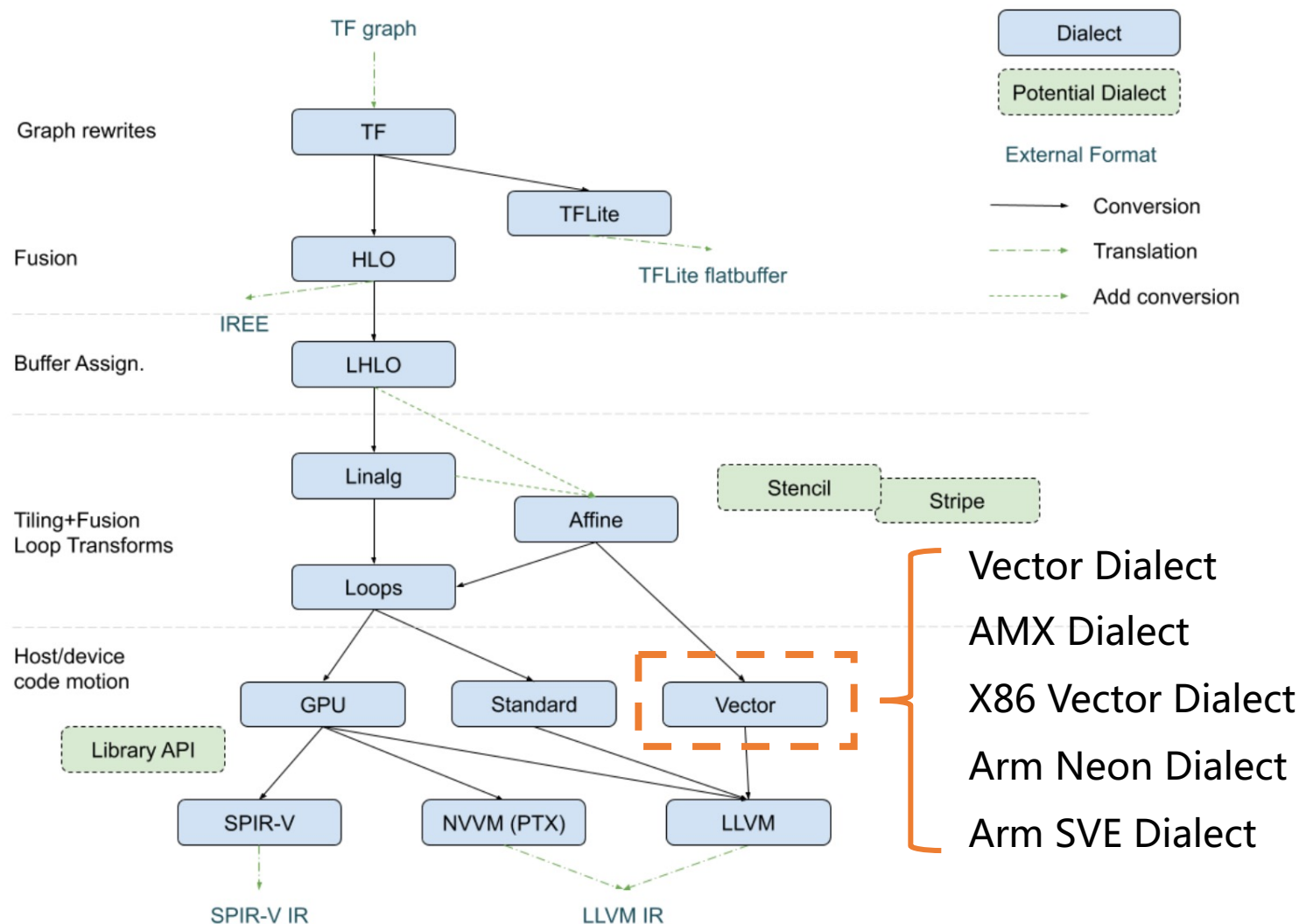
- Operation
 - RVV Operation
 - RVV Intrinsic Operation
- Type
 - Scalable Vector Type
- Conversion/Translation
 - RVV Dialect
 - LLVM Dialect
 - LLVM IR



MLIR 实践 – MLIR RISC-V Vector Dialect

Main Transformations

Abstractions / Dialects



MLIR RISC-V Vector Dialect

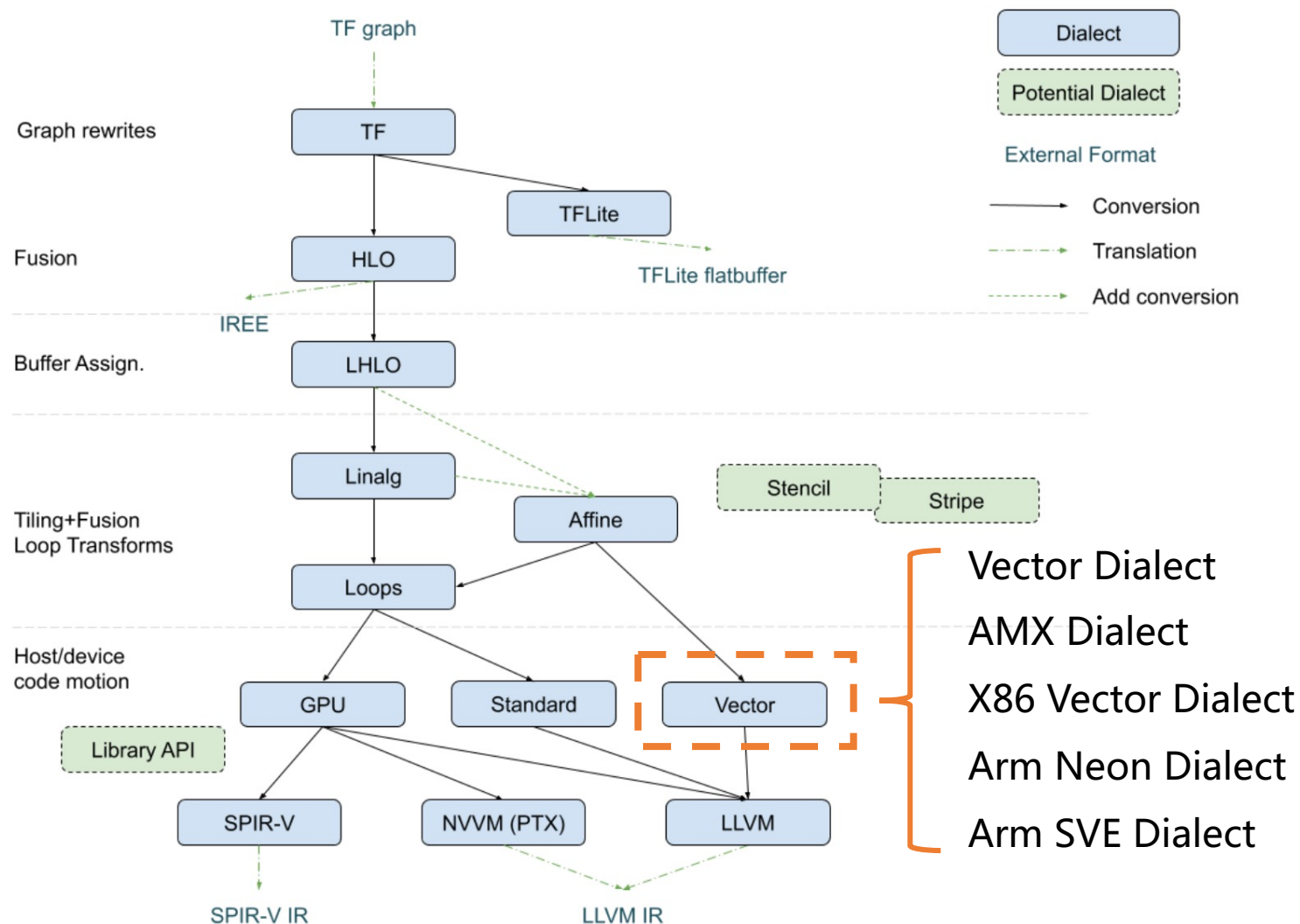
- Operation
 - RVV Operation
 - RVV Intrinsic Operation
- Type
 - Scalable Vector Type
- Conversion/Translation
 - RVV Dialect
 - LLVM Dialect
 - LLVM IR



MLIR 实践 – MLIR RISC-V Vector Dialect

Main Transformations

Abstractions / Dialects

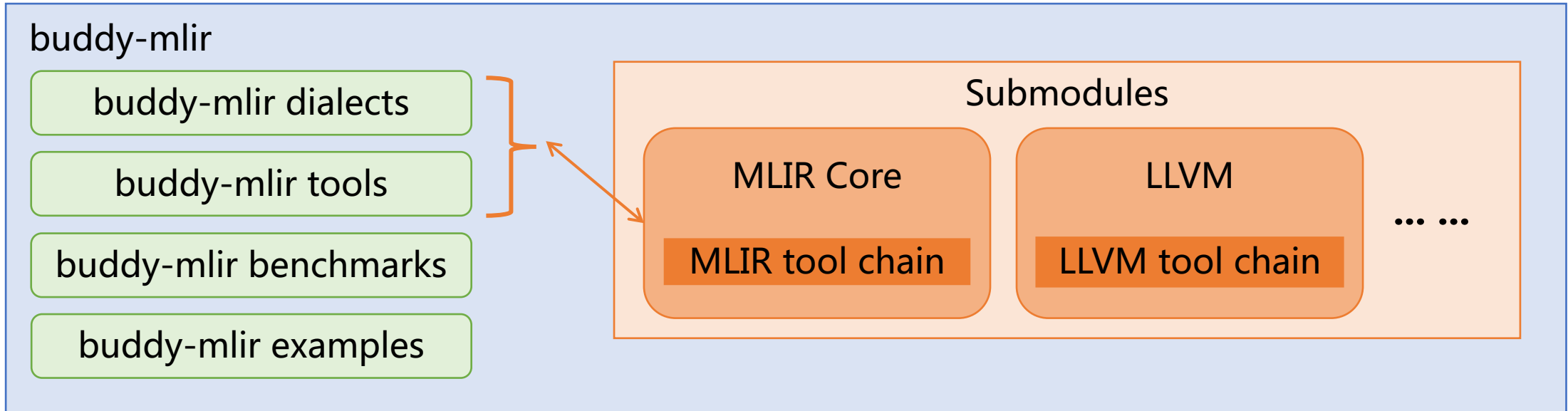


MLIR RISC-V Vector Dialect

- Operation
 - RVV Operation
 - RVV Intrinsic Operation
- Type
 - Scalable Vector Type
- Conversion/Translation
 - RVV Dialect
 - LLVM Dialect
 - LLVM IR



MLIR 实践 – Buddy MLIR



RISC-V

Vector Programming

Instruction Set Architecture

RISC-V Mentorship: MLIR Convolution Vectorization

Mentors



Mentees



边缘检测 (Sobel Kernel)



原始图片

1024 x 1024

执行 1000 次

3x3 Kernel

OpenCV: 3.31541 s

conv-opt: 1.12211 s

5x5 Kernel

OpenCV: 9.55226 s

conv-opt: 2.48518 s

7x7 Kernel

OpenCV: 21.8928 s

conv-opt: 4.47707 s

9x9 Kernel

OpenCV: 32.3142 s

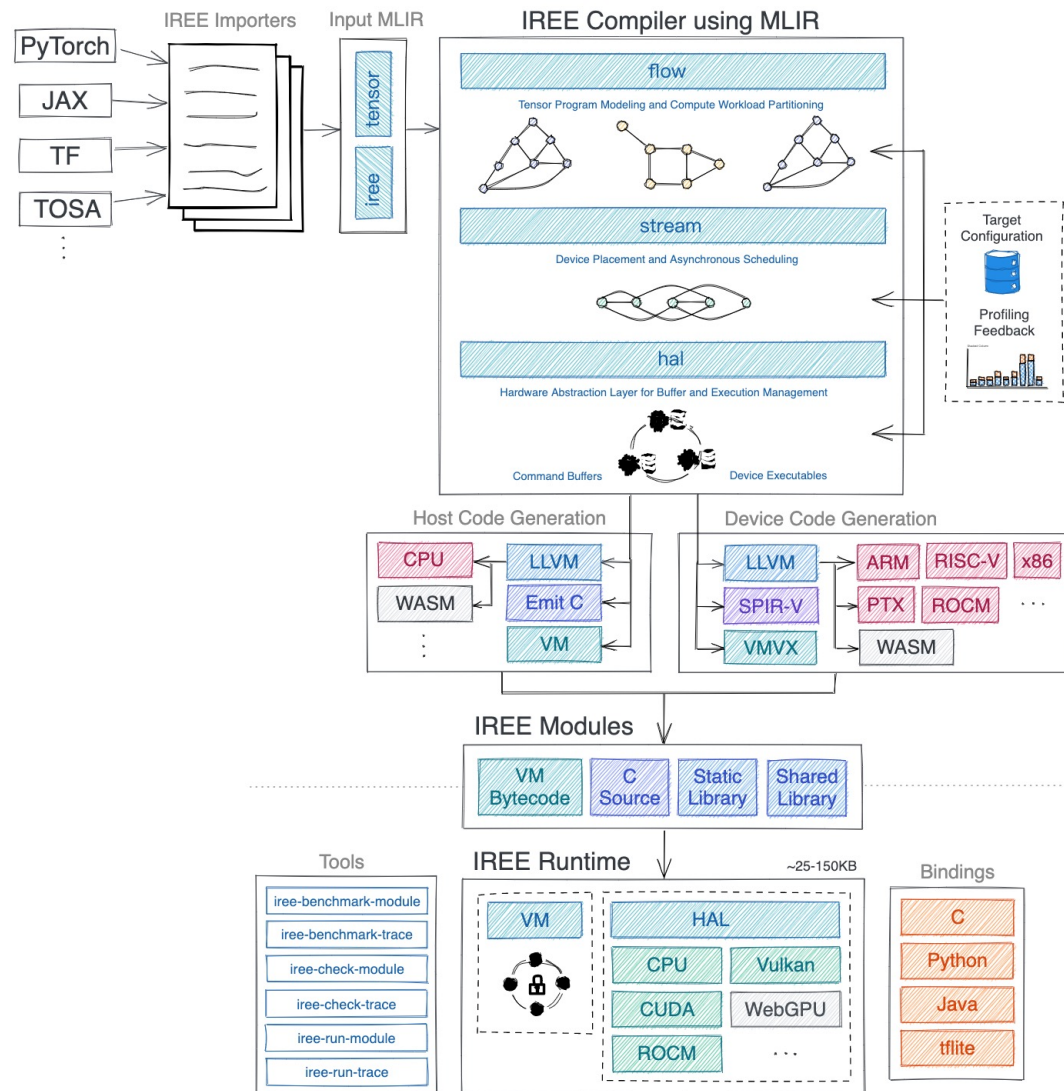
conv-opt: 7.58269 s

卷积向量化工具 conv-opt

buddy-mlir : <https://github.com/buddy-compiler/buddy-mlir>

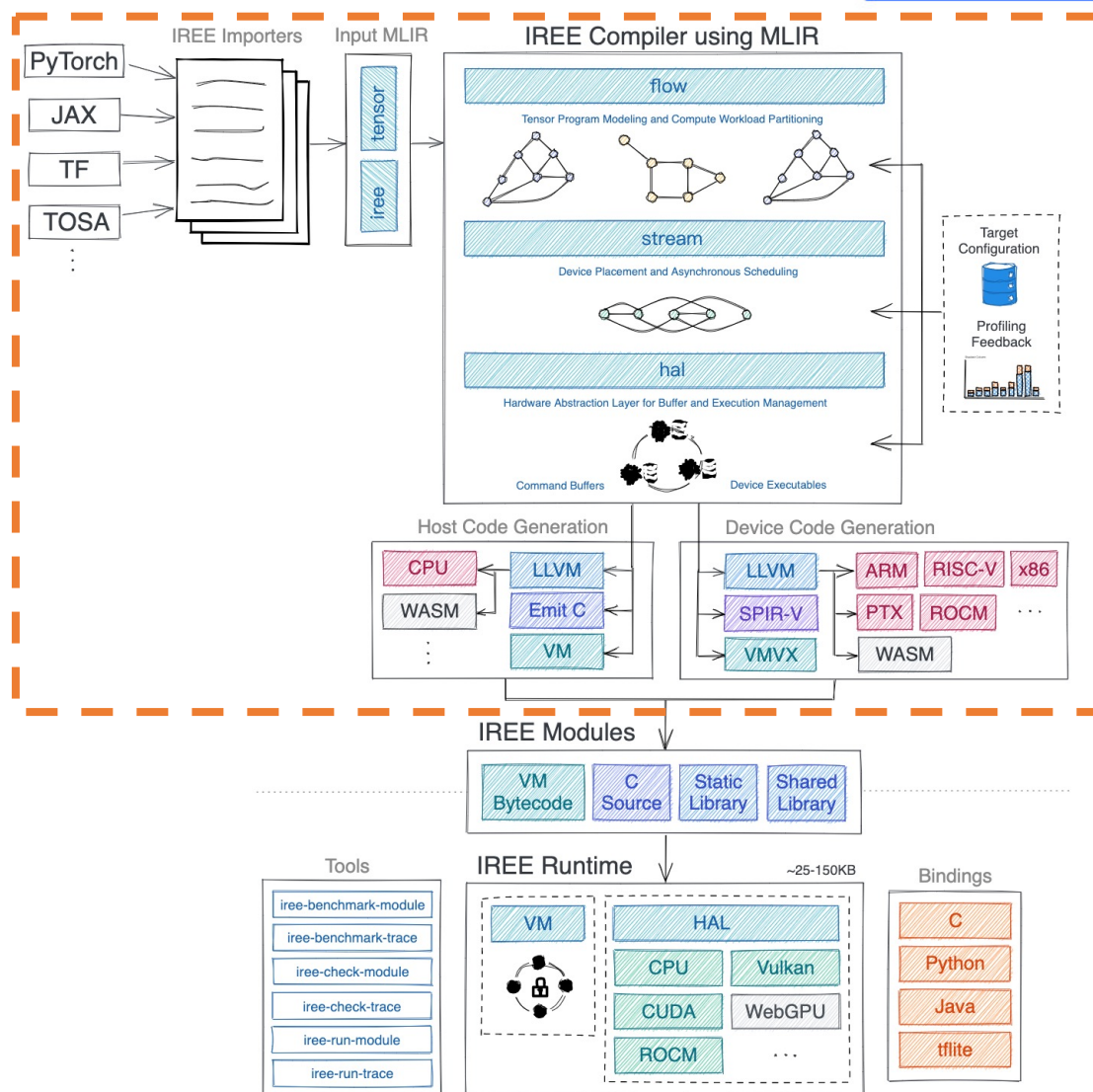
3. MLIR 思考 – 基于 MLIR 的深度学习框架

MLIR 思考 – 基于 MLIR 的深度学习框架



IREE (IR Execution Environment) 机器学习程序部署框架

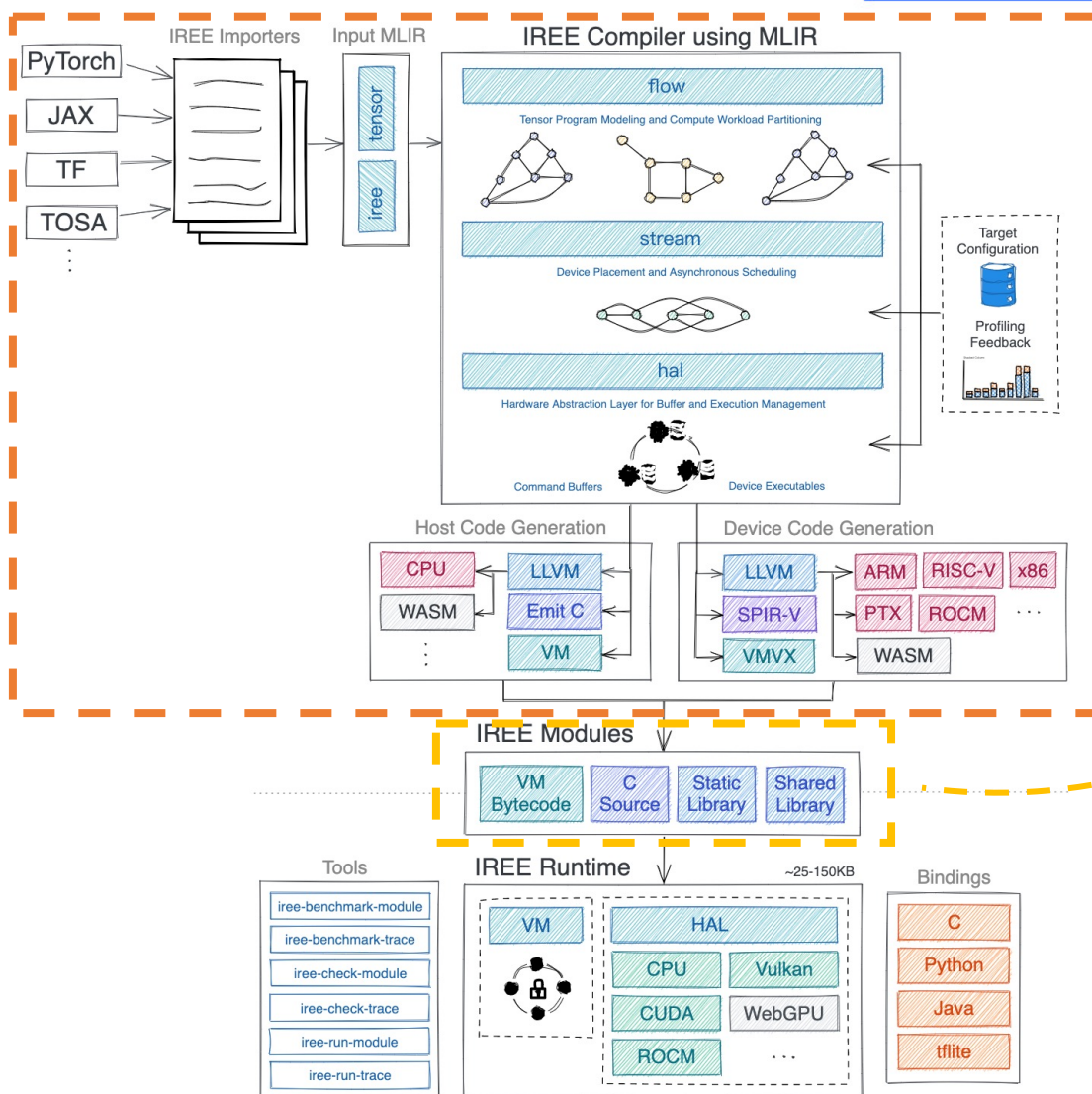
MLIR 思考 – 基于 MLIR 的深度学习框架



IREE (IR Execution Environment) 机器学习程序部署框架

从模型翻译到 MLIR
将程序分成 host 和 device
生成中间产物 (VM FlatBuffers , C file , etc.)

MLIR 思考 – 基于 MLIR 的深度学习框架

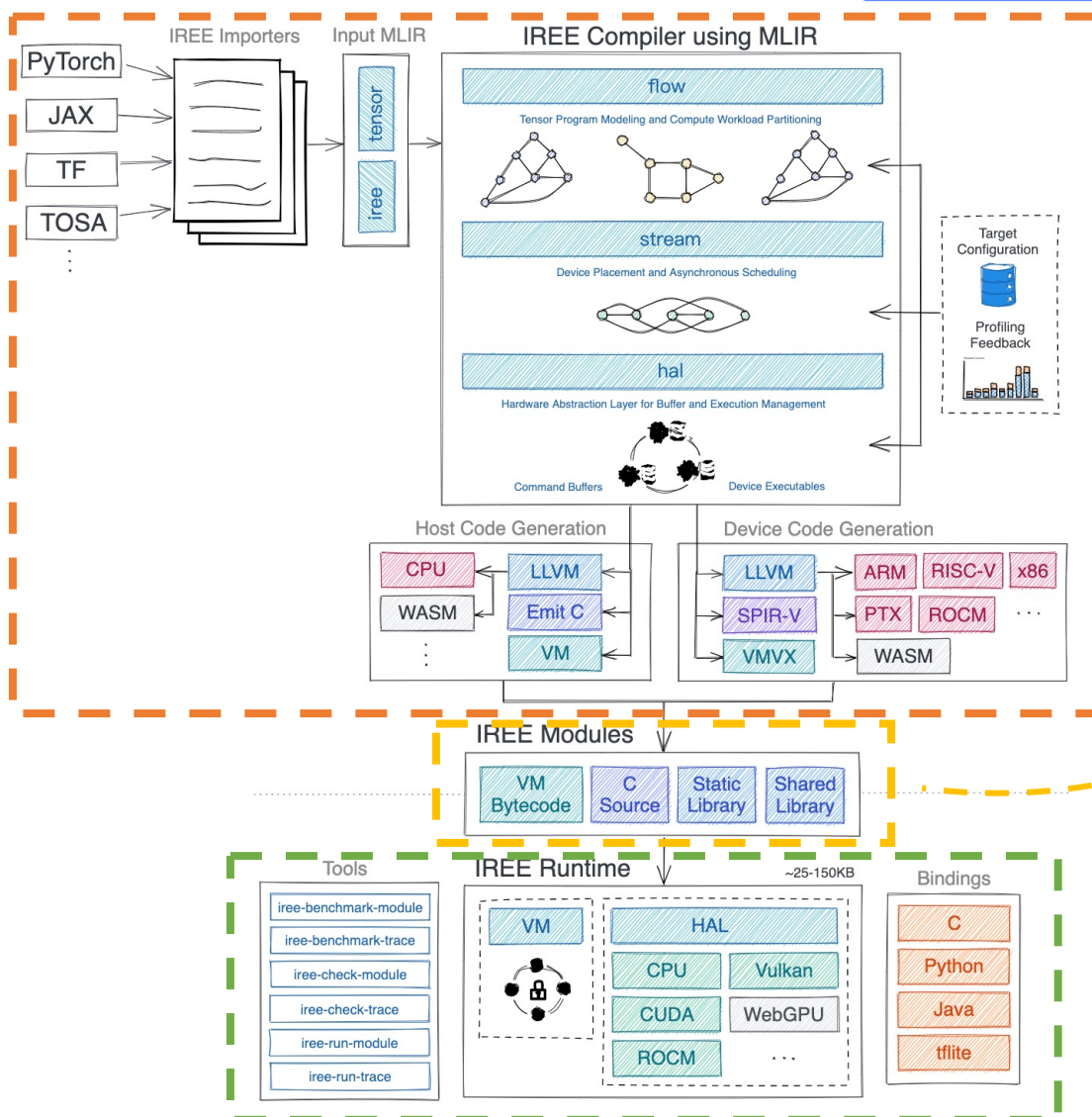


IREE (IR Execution Environment) 机器学习程序部署框架

从模型翻译到 MLIR
将程序分成 host 和 device
生成中间产物 (VM FlatBuffers , C file , etc.)

将 host 和 device 端代码生成可执行程序/库
和 IREE 硬件抽象层 (HAL) 链接

MLIR 思考 – 基于 MLIR 的深度学习框架



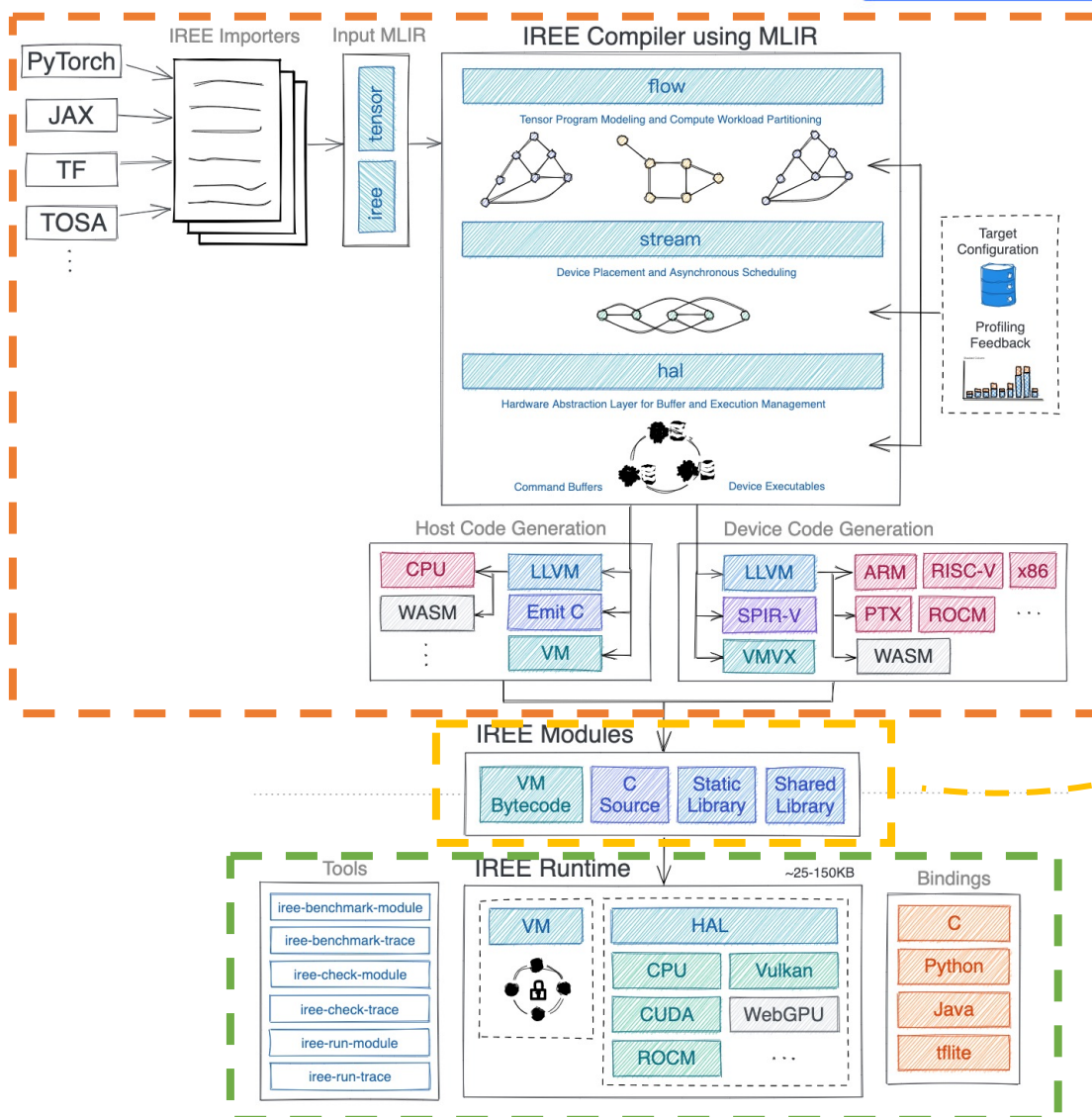
IREE (IR Execution Environment) 机器学习程序部署框架

从模型翻译到 MLIR
将程序分成 host 和 device
生成中间产物 (VM FlatBuffers , C file , etc.)

将 host 和 device 端代码生成可执行程序/库
和 IREE 硬件抽象层 (HAL) 链接

IREE 字节码虚拟机 (Bytecode VM)
硬件抽象层接口 (HAL API) 和各种硬件抽象
通用接口以及各种语言/框架的绑定

MLIR 思考 – 基于 MLIR 的深度学习框架



IREE (IR Execution Environment)

机器学习程序部署框架

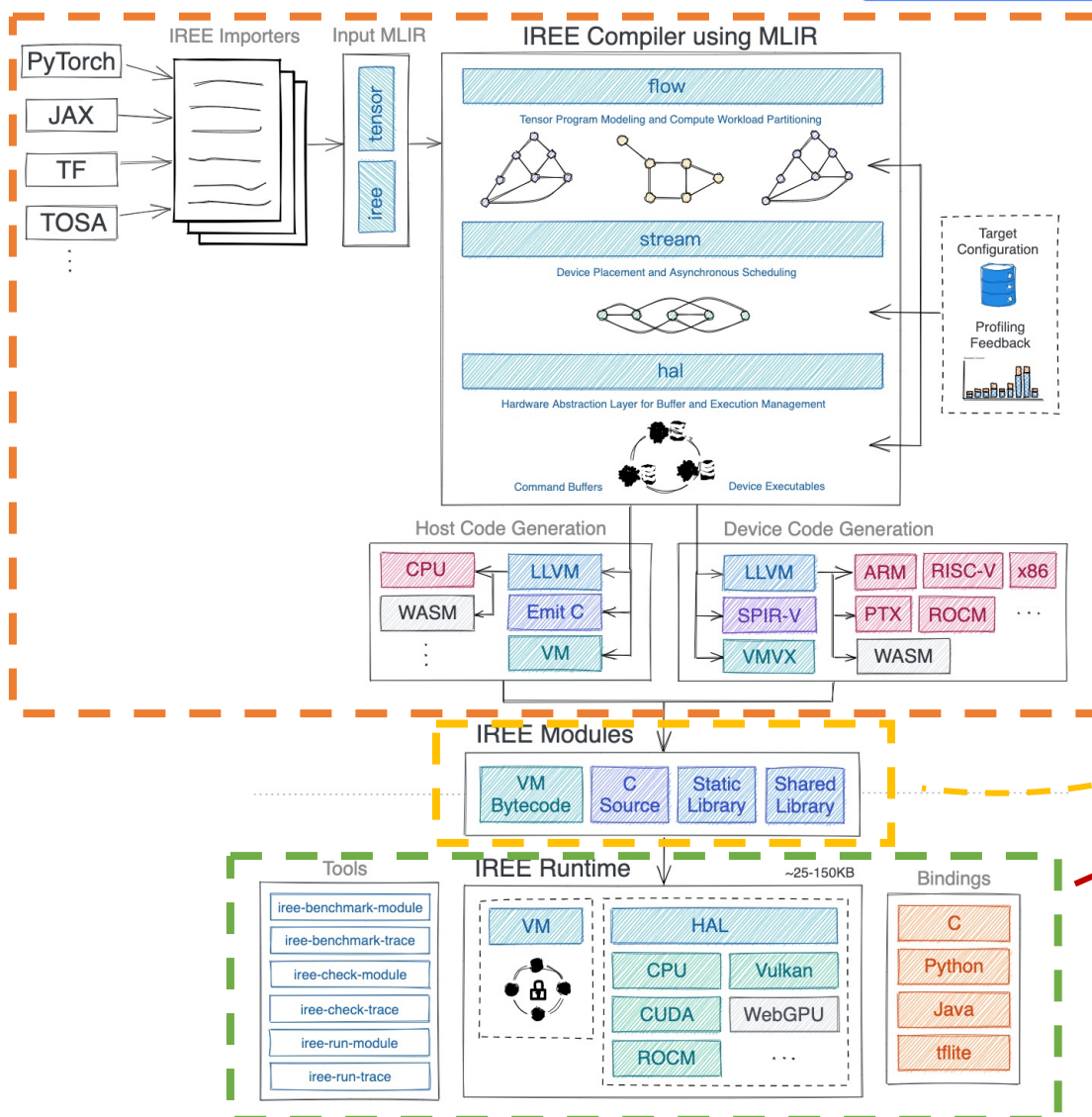
-----> 从模型翻译到 MLIR
将程序分成 host 和 device
生成中间产物 (VM FlatBuffers , C file , etc.)

-----> 如何在多层 IR 中更好的应用编译技术

将 host 和 device 端代码生成可执行程序/库
和 IREE 硬件抽象层 (HAL) 链接

-----> IREE 字节码虚拟机 (Bytecode VM)
硬件抽象层接口 (HAL API) 和各种硬件抽象
通用接口以及各种语言/框架的绑定

MLIR 思考 – 基于 MLIR 的深度学习框架



IREE (IR Execution Environment) 机器学习程序部署框架

-----> 从模型翻译到 MLIR
将程序分成 host 和 device
生成中间产物 (VM FlatBuffers , C file , etc.)

-----> 如何在多层 IR 中更好的应用编译技术

将 host 和 device 端代码生成可执行程序/库
和 IREE 硬件抽象层 (HAL) 链接

-----> 在 DSA 百花齐放的时代，如何做好统一的硬件抽象层

-----> IREE 字节码虚拟机 (Bytecode VM)
硬件抽象层接口 (HAL API) 和各种硬件抽象
通用接口以及各种语言/框架的绑定



中国科学院软件研究所
Institute of Software Chinese Academy of Sciences



智能软件研究中心
Intelligent Software Research Center

Q & A

