MLIR 编译框架的使用与探索

编译原理课程大作业

1 背景简介

1.1 MLIR 简介

MLIR 的全称为 Multi-Level Intermediate Representation，即“多层中间表示”。它本质上是一种全新的编译基础设施，提供了一种构建可重用和可扩展编译器基础结构的新方法。MLIR 的目标是解决软件碎片化的问题，降低异构计算的编译难度，降低领域专用编译器和加速器的构建成本，并协助将现有优秀的编译器相关工作连接在一起。近年来，越来越多基于 MLIR 的工作出现在计算机体系结构领域的顶会上。更多关于 MLIR 的具体信息可参考 MLIR 官方文档。

1.2 Tiny 语言简介

Tiny 语言是本次大作业的使用语言。它是一种自定义的，基于张量（tensor）的语言。Tiny 语言支持少量功能，包括函数定义，基本数学运算以及打印功能。以下是一个具体实例：在该实例中，

```
def main() {
  var a = [[1, 2, 3], [4, 5, 6]];
  var b<2, 3> = [1, 2, 3, 4, 5, 6];
  print(transpose(a) * transpose(b));
}
```

图 1. Tiny 语言实例

定义了 var a 和 var b 两个 tensor 变量。a 的 shape 隐式定义为 <2,3>，可以由后面的赋值推断。b 的 shape 显示定义为 <2,3>。另外，transpose() 函数执行矩阵转置运算，print() 函数可以输出两个矩阵相乘的结果。两者都为 Tiny 的内置函数，可以直接调用。目前 Tiny 语言只支持**一维和二维**张量以及 64 位浮点数 (C 语言中的 double) 数据类型。在实验过程中，同学们不必详细了解 Tiny 语言的内置函数功能和具体实现，我们的重点在于从编译的角度去解析这样一种语言。

2 实验步骤

2.1 环境搭建

虚拟机软件（推荐）： Vmware
操作系统（推荐）： Windows 系统：Ubuntu 20.04 （下载地址：Ubuntu20.04 镜像文件）
具体步骤： 见 tiny_project 中的 README.pdf 文件。

2.2 主要内容

在本次大作业中，我们将基于 MLIR 编译框架进行一系列代码编译与优化探索。同学们可自行组队，2 人一组，完成大作业。大作业的截止时间为 2022 年 6 月 20 日（周一）24 时。为不影响同学们期末复习，请大家提前规划完成作业。

作业主要内容分为基础部分和进阶部分，合计 20 分。在基础部分，我们将考察词法与语法分析知识，要求补全 Tiny 语言的词法与语法分析代码，并对 “var” 的表示引入新的扩展。在进阶部分，我们将带领同学们对代码优化问题进行初步探索，要求对 Tiny 语言的 transpose() 函数进行冗余代码消除。2.3 与 2.4 将给出问题的简要描述，具体实验步骤可参考 README 文档。

2.3 基础部分：词法分析与语法分析（14 分）

词法分析：对关键字和变量名进行分析。要求:

1. 能够识别 “return”、“def” 和 “var” 三个关键字
2. 按照如下要求识别变量名：
   - 变量名以字母开头
   - 变量名由字母、数字和下划线组成
   - 变量名中有数字时，数字应该位于变量名末尾
     例如：有效的变量名可以是 a123, b_4, placeholder 等。

语法分析：对 Tiny 语言中一维或二维 Tensor 的声明及定义进行语法分析。语法必须符合以下要求：

1. 必须以 “var” 开头，后面为变量名及变量类型，最后为变量的初始化
2. 语法分析器已支持以下两种声明及初始化方式，以一个二维 Tensor 为例：
   - var a = [[1, 2, 3], [4, 5, 6]];
   - var a<2,3> = [1, 2, 3, 4, 5, 6];

   需要同学们拓展 parser 功能支持第三种形式：
   - var a[2][3] = [1, 2, 3, 4, 5, 6];

结果测试：在基础部分中，我们要对词法分析器与语法分析器的功能进行验证。Tiny_project 提供了多种测试用例。其中，test_1 至 test_4 为词法分析器的测试用例，我们要求词法分析器能够对不同变量名进行正确解析，对于违法的变量名，词法分析器应该给出具体错误信息。test_5 和 test_6 为语法分析器的测试用例，要求语法分析器能成功解析测试用例，并输出。其中输出有两种形式：1. 合法的 AST；2. 在终端打印出 Tensor，即测试用例运行成功。详细步骤见 README。
注：var a [2][3] = ... 为我们新增的声明及定义方式，在本次作业中对于此种形式，我们只要求语法分析器能识别这种表示，生成对应的合法 AST 即可。如果想要在终端打印出 Tensor 信息，则需拓展 Tiny dialect 以支持此新增表示，感兴趣的同学可以自己了解尝试。

2.4 进阶部分：代码优化（6 分）

代码优化：进行冗余代码的消除。Tiny 语言内置的 transpose 函数会对矩阵进行转置操作。然而，对同一个矩阵进行两次转置运算会得到原本的矩阵，相当于没有转置。矩阵的转置运算是通过嵌套 for 循环实现的，而嵌套循环是影响程序运行速度的重要因素。因此，侦测到这种冗余代码并进行消除

是十分必要的。这里同学们需要在对应文件中根据提示补充优化 pass 的关键代码，并测试是否真正实现了冗余代码消除。

结果测试：在进阶部分中，我们要对冗余代码的优化效果进行验证。test_7 提供了一个冗余转置操作的实例，我们要求编译器能够通过代码优化去掉冗余的转置操作，输出优化后的结果，同学们可以对比下优化前后的结果，从而更直观地理解冗余消除的效果。详细步骤见 README。

3 提交内容

请在截止日期前在 Canvas 平台指定位置提交大作业的压缩文件，应包含一份实验报告和代码。

1）实验报告：

实验报告应指出实验环境（操作系统和工具版本等）、实验过程、实验结果，并能完整、准确地说出自己的设计思路，描述清楚各个函数的作用、具体实现方法等。总结部分可以分享自己在实验过程中遇到的问题和解决方法，对 MLIR 框架的认识。也可以对大作业提出自己的意见，我们将在后面的学期不断进行完善。**实验报告中应写明组内两位同学的姓名、学号、联系方式及各自贡献**。

2）代码：

请将完整的源代码、实验结果的截图（按照对应测试用例 test_n 命名）一起打包提交。如果你在 TODO 以外的地方修改代码，请在你修改/添加代码的位置加上注释（基础部分：// basic part；进阶部分：// advanced part），方便我们后续审核。