Triton: An Intermediate Language and Compiler for Tiled Neural Network Computatio

JX-Ma

2024/3/30

1 摘要

本文提出了 Trion，一种围绕 tile 概念的语言和编译器，即静态形状的多维子数组。该方法围绕 1. 基于 c 语言和基于 llvm 的中间表示 (IR)，用于根据参数贴图变量的操作来表达张量程序;2. 一组新颖的贴图级优化通道，用于将这些程序编译成高效的 GPU 代码。

2 笔记

这篇文章提出了 Triton-C，一种类似 c 的语言，用于根据参数贴图变量表示张量程序。这种语言的目的是为现有的 DNN 编译器。Triton-IR 一种基于 llvm 的中间表示 (IR)，它提供了适合于 tile 级程序分析、转换和优化的环境。Triton-JIT：一个即时 (JIT) 编译器和代码生成后端，用于将 Triton-IR 程序编译成高效的 LLVM 位码。

triton-C 的目的是为现有 (和未来) 的 DNN 编译器以及熟悉低级 GPU 编程的程序员提供稳定的前端。Triton-c 的语法基于 CUDA-C 的语法，经过了修改和拓展。该语言提出了新的声明变量，Tile, 它主要强调了声明的是多维数组，例如 int Tile[16,16], 等价于 CUDA-c 中的 int tile[16][16]. 一维数组的初始化可以使用 int range[8] = 0 ... 8. 内置函数在原有 c 的基础上 (+,,&&,), 添加了各种内置函数 (dot,trans,get_global_range).n 维的 tiles 可以使用 newaxis 来沿着任意的轴广播。而后介绍了 tile 语义的好处，Triton-C 中内置的 tile 类型和操作 (即 tile 语义) 提供了两个主要好处。首先，它通过隐藏与 tile 内内存合并、缓存管理和专用硬件利用率有关的重要性能细节，简化了张量程序的结构。其次，它打开了编译器自动执行这些优化的大门. triton 编程模型每个内核都是单线程的并且与一组全局范围相关联, 这种方法可以使内核更加简单。

Triton-IR 是一种基于 llvm 的中间表示 (IR)，其目的是提供适合于 tile 级程序分析、转换和优化的环境。在最高层次上，Triton-IR 程序由一个或多个称为模块的基本编译单元组成。这些模块相互独立地编译，并最终由一个链接器聚合，其作用是解析前向声明并充分合并全局定义。Triton-IR 函数定义由返回类型、名称和可能为空的参数列表组成。如果需要，可以添加额外的可见性、对齐和链接说明符。函数属性 (如内联提示) 和参数属性 (如只读、别名提示) 也可以允许编译器后端执行更激进的优化。

triton-IR 使用静态单一赋值 (SSA) 形式，这意味着每个基本块中的每个变量必须 (1) 只被赋值一次，(2) 在使用之前定义。这样，每个基本块隐式地定义了一个数据流图 (DFG)，它的不同路径对应于程序 SSA 表示中的 use-def 链。

triton-JIT 的目标是通过一组与机器无关 (第 5.1 节) 和与机器相关 (第 5.2 节) 的传递，通过自动调优引擎 (第 5.3 节)，将 Triton-IR 程序简化并编译成高效的机器码。与机器无关的传递是指循环内部的块级内存操作可能是有问题的，因为它们可能导致严重的延迟，在缺乏足够的独立指令时无法隐藏。但是，在 Triton-IR 中可以通过检测循环并在必要的地方添加适当的预取代码来直接缓解这个问题。Triton-JIT 执行的优化包括 (1) 分层平铺，(2) 内存合并，(3) 共享内存分配和 (4) 共享内存同步。分层平铺大致的意思是把原本的二维块矩阵分层，最开始矩阵计算直接放入 CUDA 核心中进行计算，但是矩阵分层后将每一层都放入 CUDA 核心中的 SIMD 上，这种策略减少了加载一个 tile 列所需的内存事务数量。内存合并：当相邻的线程同时访问附近的内存位置时，内存访问被称为合并。由于 Triton-IR 程序是单线程和自动并行的，我们的编译器后端能够在每个微块内部对线程进行排序，以便在可能的情况下避免非合并内存访问。这种策略减少了加载一个 tile 列所需的内存事务数量。共享内存分配：共享内存分配传递的目的是确定应该在何时何地将文件存储到该空间。共享内存同步：共享内存同步通道的目标是在生成的 GPU 源代码中自动插入屏障，以保持程序的正确性。

3 总结

这篇论文主要提出了一个开源语言和编译器，Triton, 用于将平铺神经网络计算表达和编译为高效的机器码。首先提出了 Triton-C，基于 CUDA-C 的语言，它包含了很多 CUDA-C 的函数，并且在内置函数中有拓展，这些拓展有利于我们在卷积上对数据进行操作。还有 Triton-IR，一种基于 llvm 的中间表示 (IR)，和 triton-JIT，引入这个开源语言和编译的目标就是把卷积使用的张量平铺成适用于 GPU 结构的算子，充分的利用了 GPU 的结构特性，例如在共享内存上面的优化，