Recognizing Complex Arithmetic

Nicholai Tukanov, Chengyue Wang
{ntukanov, cw4}@cmu.edu
https://github.com/nicholaiTukanov/RecognizingComplex

Description

A single complex matrix multiplication can be computed as four separate real matrix multiplications.

Equation 1:

$$C_r = (A_r * B_r - A_i * B_i + C_r)$$
$$C_i = (A_r * B_i - A_r * B_r + C_i)$$

In languages like Tensorflow, the four real matrix multiplication method is the only way to represent complex arithmetic within an ML model. This is due to the inability to support ML layers that perform complex arithmetic. By separating out all the complex arithmetic into four separate kernel calls, it adds extra function overhead and gets rid of potential data reuse (i.e., matrices needs to be reread multiple times rather than being held resident in faster levels of the memory hierarchy). More importantly, it doesn't take advantage of high performance complex matrix multiplication kernels that already exist.

Our project will focus on developing a compiler pass using LLVM MLIR and Tensorflow to replace the four matrix multiplication method with a single high performance complex kernel (see Fig. 1). This requires our pass to preserve correctness and increase the performance of the original program. This is done by using data format conversion routines along with high performance BLAS libraries for the various architectures we will test on (i.e., MKL, cuBLAS), respectively.

Milestones

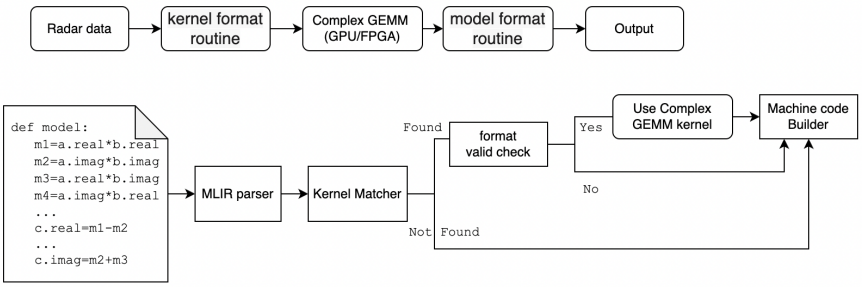

Fig. 1

April 14th

- Infrastructure for the pass
- 1 architecture
- Suite of conversion routines

April 27th

- 75%
  - Pass working for one format on one architecture (GPU)
  - Framework for being able to add more formats and architectures
- 100%

    - Pass working for multiple architectures on a heterogeneous system (CPU, GPU, FPGA)
    - Framework contains many format routines
    - Format routines can be auto-generated
- 125%
    - Convert the entire model to a format that benefits across high performance libraries calls

Plan of Attack and Schedule (week-by-week)

Week 1: Set up a simple pass using MLIR that only prints Function info, and begin to implement conversion routines

- Nicholai: write simple pass using MLIR
- Chengyue: writing test models for pass and also run through the code about the progressive Raising in MLIR
- Both: begin creating the infrastructure and conversion routines

****Week 2-3**:** Begin creating the frontend and backend infrastructure for the pass (critical path)

- Nicholai: frontend (i.e., parser for MLIR to find complex arithmetic)
- Chengyue: backend (i.e., linking libraries and conversion routines)
- Both: build system and conversion routines

Week 4: Collect results for milestone from 1 architecture

Both: implement testing system

Week 5: Expand pass infrastructure to more architectures

- Nicholai: finding more CPU libraries/creating our own complex kernel
- Chengyue: writing complex arithmetic kernels for FPGAs
- Both: generalizing infrastructure for the pass

Week 6: Collect results for poster

- Both: test the pass on various architectures and models and draw up poster

Literature search

- MLIR paper
- Progressive Raising in Multi-level IR
- Mixed data layout
- Implementing high-performance complex matrix multiplication via the 3m and 4m methods
- Implementing high-performance complex matrix multiplication via the 1m method
- BLIS: A Framework for Rapidly Instantiating BLAS Functionality
- Tensorflow
- LLVM

Resources

- Software (locally installed on test machines):

    - LLVM+MLIR
    - Tensorflow
    - Python
- Hardware (all owned in lab):
    - Intel Rocket Lake CPU
    - Nvidia Volta GPU
    - Intel Statrix FPGA

Getting Started

- Started writing Tensorflow test models for the pass
- Reading documentation on MLIR's OperatorPass and Tensorflow's interaction with MLIR
- Can the website just be a GitHub repo?