Tensorization with MLIR

Liangfu Chen, SDE@AWS
Compilation Flow

- TVM/Relay
- Schedule
- Translate
- CodeGen

Easier to control workflow
Faster execution
Why tensorize in MLIR?

Idea: Use the right tool to do the right job.

Graph level optimizations
- AlterOpLayout
- BackwardFoldScaleAxis
- Conv2dToSparse
- DenseToSparse
- FoldConstant
- PartitionGraph
- SimplifyInference
- ...

TIR transformations
- BF16Legalize
- BackwardFoldScaleAxis
- LoopPartition
- LowerIntrin
- Simplify
- UnrollLoop
- VectorizeLoop
- ...

Affine loop transformations
- affine-loop-fusion
- affine-loop-invariant-code-motion
- affine-loop-tile
- affine-loop-unroll
- affine-super-vectorize
- cse: Eliminate common sub-expressions
- normalize-memrefs: Normalize memrefs
- ...

TVM/Relay ➔ TE ➔ TIR ➔ LLVM/MLIR ➔ Machine Code
How to tensorize in MLIR?

Idea: Loop tiling and pattern matching.

• Loop tiling (`-affine-loop-tile="tile-size=32"`)
How to tensorize in MLIR?

Idea: Loop tiling and pattern matching.

• Pattern matching

```cpp
class MyPattern : public RewritePattern {
public:
  /// This overload constructs a pattern that only matches operations with the
  /// root name of 'MyOp'.
  MyPattern(PatternBenefit benefit, MLIRContext *context)
    : RewritePattern(MyOp::getOperationName(), benefit, context) {}

  /// In this section, the 'match' and 'rewrite' implementation is specified
  /// using the separate hooks.
  LogicalResult match(Operation *op) const override {
    /// The 'match' method returns 'success()' if the pattern is a match, failure
    /// otherwise.
    /// ...
    return success();
  }

  void rewrite(Operation *op, PatternRewriter &rewriter) {
    /// The 'rewrite' method performs mutations on the IR rooted at 'op' using
    /// the provided rewriter. All mutations must go through the provided
    /// rewriter.
  }
};
```
Conclusion

• Transform Relay to MLIR for tensorization
  • Transform Relay -> TIR -> MLIR to get tensor expressions

• Leverage MLIR to perform tensorize operations
  • Perform loop tiling and pattern matching
  • Polyhedral analysis and optimizations

• Build whole graph into a single function
  • Scanning whole graph helps management of scratchpad in compiler

• Perform data rate matching between loop nests
  • Improves temporal locality