TRACKING OPERATIONS THROUGH MLIR PASS PIPELINES USING SOURCE LOCATIONS: A PRACTICAL DEBUGGING WORKFLOW FOR LARGE MLIR PIPELINES

Florian Walbroel
walbroel@roofline.ai

roofline

QUANTIZED CONVOLUTIONS IN IREE CPU LOWERING

Optimizing IREE's CPU pipeline

- Quantized convolutions had poor performance
- Heuristic lowering changed the generated IR

Manual tracking across large MLIR dumps did not scale

What does this operation decompose into?

```
622   %619 = tosa.slice %618, %610, %609 : (tensor<?x160x160x3xi8>, !tosa.shape<4
623   %620 = tosa.conv2d %619, %612, %611, %616, %615 {acc_type = i32, dilation =
624   %621 = tosa.rescale %620, %607, %606, %608, %616 {input_unsigned = false, o
```

```
1846  %collapsed_453 = tensor.collapse_shape %10 [[0, 1], [2, 3, 4]] : tensor<79x
1847  %collapsed_454 = tensor.collapse_shape %__hoisted_tensor_3x3x3x32xi8 [[0, 1,
1848  %collapsed_455 = tensor.collapse_shape %8 [[0, 1], [2]] : tensor<79x79x32xi
1849  %11 = linalg.matmul ins(%collapsed_453, %collapsed_454 : tensor<6241x27xi8>,
1850  %__hoisted_tensor_32xi32 = util.global.load immutable @__hoisted_tensor_32x
1851  %expanded_456 = tensor.expand_shape %11 [[0, 1], [2]] output_shape [79, 79,
```

SOURCE LOCATIONS IN MLIR

MLIR operations can carry source locations

```
static Operation *create(Location location, OperationName name,
                         TypeRange resultTypes, ValueRange operands,
                         NamedAttrList &&attributes,
                         OpaqueProperties properties, BlockRange successors,
                         unsigned numRegions);
```

mlir/include/mlir/IR/Operation.h

- Operator API enforces preservation

In practice, locations often survive transformations well

```
#loc50 = loc("source.mlir":201:12)
...
#loc220 = loc(callsite(#loc50 at #loc1))
```

```
%240 = arith.addi %239, %c-128_i32 : i32 loc(#loc220)
%241 = arith.maxsi %240, %c-128_i32 : i32 loc(#loc220)
%242 = arith.minsi %241, %c127_i32 : i32 loc(#loc220)
%243 = arith.trunci %242 : i32 to i8 loc(#loc220)
```

mlir-opt --mlir-print-debuginfo...

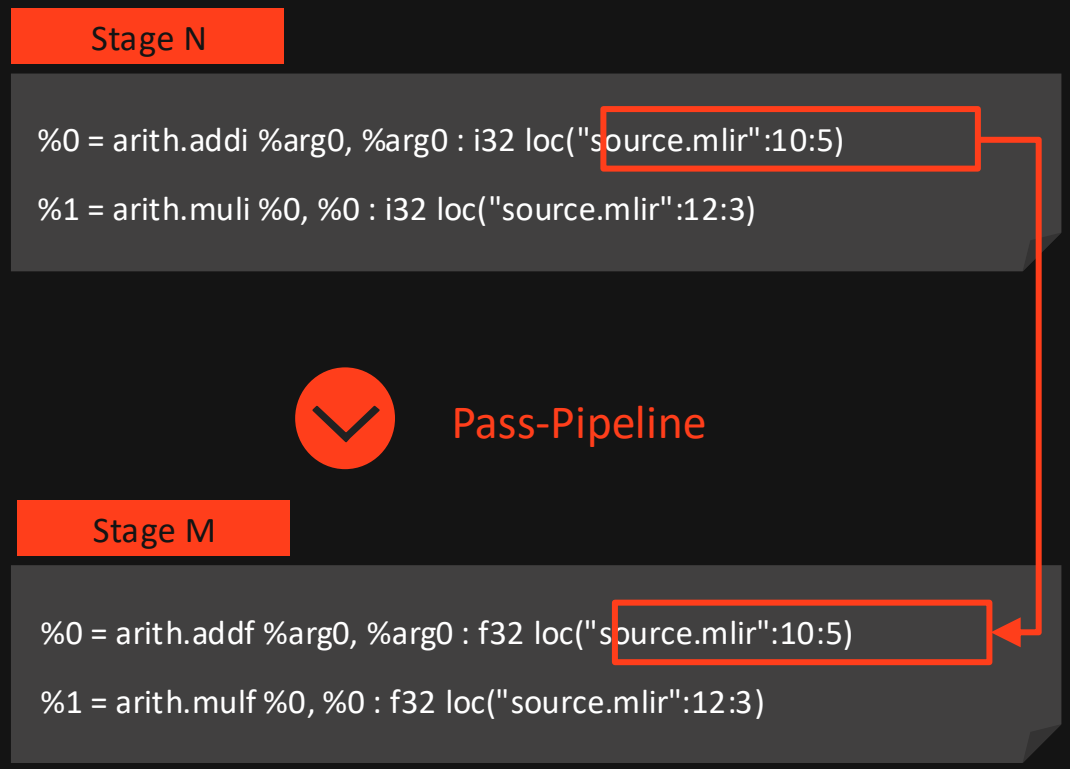
→ Raw locations exist, but they are not yet a usable tracking workflow

APPROACH

TURNING SOURCE LOCATIONS INTO A TRACKING MECHANISM

Idea

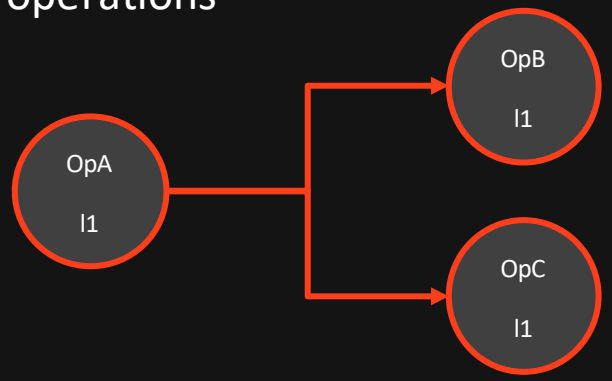
- Dump MLIR at different stages of the pipeline
- Extract each operation’s source location / range
- Associate operations across stages by overlapping source ranges
- Query forward or backward to reconstruct lineage



HOW LOCATIONS BEHAVE UNDER TRANSFORMATIONS

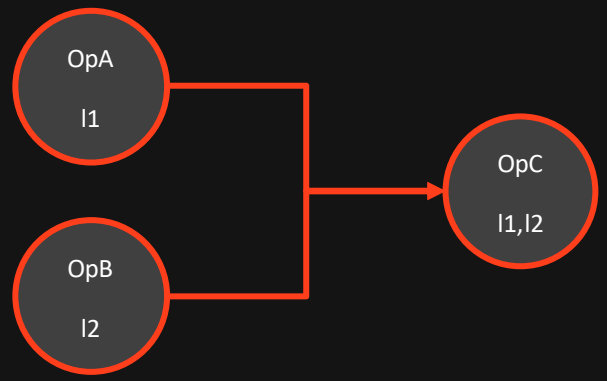
Copied

- Operation copied / replaced
- Expansion into multiple operations



Combined

- Fusion
- Nesting operations

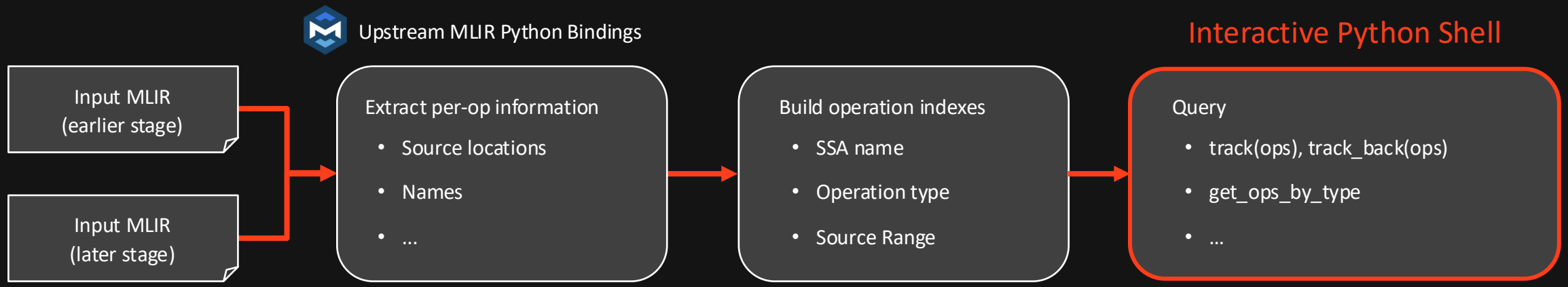


Lost

- Eliminated operations
- Some passes preserve locations poorly

- Tracking works because locations often survive or propagate
- Meaningful source ranges require post-processing

A LEAN PYTHON TOOL FOR OPERATION TRACKING




```
$ adts track-src --input-mlir import.mlir --track-mlir tosa.mlir --src-remap-list %i=import.mlir
Loading input MLIR module...
Loading tracking MLIR module...
Building operation index...
Built operation index with 2901 operations.
Building tracking operation index...
Built operation index with 888 operations.
Starting interactive shell...
Interactive shell. Type 'info' for available commands.
>>>
```

You can find the tool here: https://github.com/RooflineAI/mlir-track-src

THE TOOL HELPS TO UNDERSTAND WHAT OPERATIONS BECOME

MLIR

Stage N

```
622    %619 = tosa.slice %618, %610, %609 : (tensor<?x160x160x3xi8>, !to
623    %620 = tosa.conv2d %619, %612, %611, %616, %615 {acc_type = i32,
624    %621 = tosa.rescale %620, %607, %606, %608, %616 {input_unsigned
```

Stage M

```
1847   %collapsed_454 = tensor.collapse_shape %__hoisted_tensor_3x3x
1848   %collapsed_455 = tensor.collapse_shape %8 [[0, 1], [2]] : ten
1849   %11 = linalg.matmul ins(%collapsed_453, %collapsed_454 : tens
1850   %__hoisted_tensor_32xi32 = util.global.load immutable @__hois
1851   %expanded_456 = tensor.expand_shape %11 [[0, 1], [2]] output_
```

Python Shell

```
>>> show(track(op_index.get_ops_by_name("%620")))
util.global private @__hoisted_tensor_3x3x3x32xi8 {stream.affinity.default =
```

```
^bb0(%in: i8, %out: i8):
  linalg.yield %in : i8
} -> tensor<79x79x3x3x3xi8>
---------------------------------------
%collapsed_453 = tensor.collapse_shape %10 [[0, 1], [2, 3, 4]] : tensor<79x79x3x
---------------------------------------
%collapsed_454 = tensor.collapse_shape %__hoisted_tensor_3x3x3x32xi8 [[0, 1, 2],
---------------------------------------
%collapsed_455 = tensor.collapse_shape %8 [[0, 1], [2]] : tensor<79x79x32xi32> i
---------------------------------------
%11 = linalg.matmul ins(%collapsed_453, %collapsed_454 : tensor<6241x27xi8>, ten
2xi32>
---------------------------------------
%__hoisted_tensor_32xi32 = util.global.load immutable @__hoisted_tensor_32xi32 :
```

```
func track(op):
    return [t for t in track_ops
            if t.src_rng.overlaps(op.src_rng)]
```

THE TOOL CAN ALSO BE APPLIED FOR SYSTEMATIC ANALYSIS

Question

- Which tosa.conv2d operations take the matmul lowering path?

Query

- Track every convolution into a later pipeline stage
- Check whether any tracked operation is a linalg.matmul

```
>>> len(op_index.get_ops_by_type("tosa.conv2d"))
133
>>> for op in op_index.get_ops_by_type("tosa.conv2d"):
...     tracked_ops = track(op)
...     has_matmul = any(o.op_name == "linalg.matmul" for o in tracked_ops)
...     if not has_matmul:
...             non_decomposed_convs.append(op)
...
>>> len(non_decomposed_convs)
69
```

Result

- 133 convolutions in total
- 69 not decomposed into matmul
- 64 decomposed into matmul

KEY TAKEAWAYS

What works

- Source locations in MLIR are more useful than just printed debug info
- With light post-processing, they can enable cross-stage tracking
- This makes deep MLIR pipelines much easier to debug and optimize

Limits

- Tracking is approximate, not exact
- Depends on source locations being preserved, developers need to keep them in mind
- Ambiguity grows when operations share location

ABOUT US

WE BRING THE EXPERTISE TO ENABLE YOUR AI DEPLOYMENT

Founded by AI compiler and AI system engineers from RWTH Aachen University.

Built around a core team of ex-AMD engineers, researchers and strategists.

Multi-million funding secured from public institutions and private investors.

Florian Walbroel

walbroel@roofline.ai

www.roofline.ai

Trusted and supported by:

NXP arm intrinSic Co-funded by the European Union Federal Ministry for Economic Affairs and Climate Action

NVIDIA INCEPTION PROGRAM EDGEAI FOUNDATION PARTNER intel. liftoff eXIST SPRIN-D