Linalg vs. MDH — A Comparison of Two MLIR Dialects

Jens Hunloh, Lars Hunloh, Richard Schulze, Sergei Gorlatch, Ari Rasch, Tobias Grosser

https://mdh-lang.org

University of Cambridge

Universität Münster

MLIR is a compiler framework that offers a solid, uniform infrastructure for compiler developers to conveniently design and implement *Domain-Specific Languages (DSLs)* (a.k.a. *dialect* in MLIR terminology)

Linalg and MDH are two MLIR dialects for expressing data-parallel computations

This poster aims to identify differences between Linalg and MDH

```
#map1 = affine_map<(d0, d1) -> (d0, d1)>
#map2 = affine_map<(d0, d1) -> (d1)    >
#map3 = affine_map<(d0, d1) -> (d0)    >
module {
  func.func @main() {
    %M = memref.alloc() : memref<128x64xf32>
    %v = memref.alloc() : memref<64xf32>
    %w = memref.alloc() : memref<128xf32>
    linalg.generic
    {
      indexing_maps = [#map1, #map2, #map3],
      iterator_types = ["parallel", "reduction"]
    } ins(%M,%v:memref<128x64xf32>,memref<64xf32>)
      outs(%w:memref<128xf32>) {
    ^bb0(%in_1: f32, %in_2: f32, %out: f32):
      %0 = arith.mulf %in_1, %in_2 : f32
      %1 = arith.addf %out, %0 : f32
      linalg.yield %1 : f32
    }
    return
  }
}
```

MatVec in Linalg

MDH separates the scalar operation of a computation from the operations for combining intermediate results in a particular dimension of the iteration space

Separation allows MDH:

- parallelizing and optimizing also reduction-like parts within the computation
- avoiding unnecessary memory access (e.g., Linalg requires 0-initialized output vector)
- expressing also more advanced computations whose reduction dimensions rely on different kinds of operators

MDH

```
func.func @main()
{
  %M = memref.alloc() : memref<128x64xf32>
  %v = memref.alloc() : memref<64xf32>

  %w = mdh.compute "mdh_matvec"
  {
    inp_view =
    [
      [ affine_map<( i,k ) -> ( i,k )> ],
      [ affine_map<( i,k ) -> ( k )  > ]
    ],

    md_hom =
    {
      scalar_func = @mul,
      combine_ops = [ "cc", ["pw",@add] ]
    },

    out_view =
    [
      [ affine_map<( i,k ) -> ( i )> ]
    ]
  }
  {
    inp_types = [ f32, f32 ],
    mda_size  = [ 128,64 ],
    out_types = [ f32 ]
  }( %A,%B ) : ( memref<128x64xf32> , memref<64xf32> ) ->
                                                memref<128xf32>

  return
}
```

MatVec in MDH

This poster aims to stimulate discussions between the developers and users of Linalg and MDH