md_poly: A Performance-Portable Polyhedral Compiler Based on Multi-Dimensional Homomorphisms

a.rasch@wwu.de

Ari Rasch and Sergei Gorlatch

WWU MÜNSTER

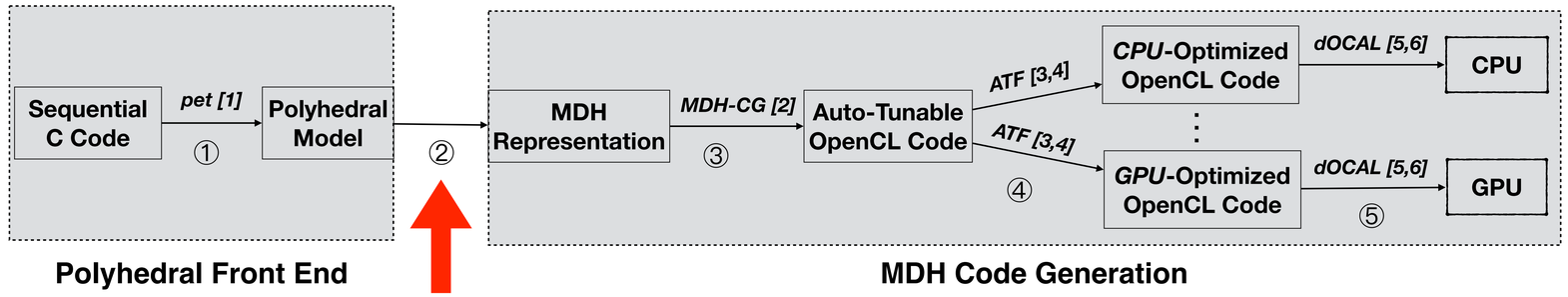
Observation

- ***Polyhedral approaches [1]*** provide high *productivity* → automatically parallelize sequential program code

- ***The MDH approach [2]*** achieves often higher *performance* than polyhedral compilers; its generated code is *portable* over different architectures (e.g., GPU and CPU).

We aim to combine both advantages!

Approach



1. Transforming sequential C program to polyhedral model.
2. **Transforming polyhedral model to MDH representation.**
3. Generating auto-tunable OpenCL code from MDH representation.
4. Auto-tuning OpenCL code for particular device and problem size.
5. Executing auto-tuned OpenCL code.

[1] Verdoolaege, Grosser, "Polyhedral Extraction Tool.", IMPACT'12
[2] Rasch, Schulze, Gorlatch, "Generating Portable High-Performance Code via Multi-Dimensional Homomorphisms.", PACT'19
[3] Rasch, Haidl, Gorlatch, "ATF: A Generic Auto-Tuning Framework.", HPCC'17
[4] Rasch, Gorlatch, "ATF: A Generic, Directive-Based Auto-Tuning Framework.", CCPE'19
[5] Rasch, Wrodarczyk, Schulze, Gorlatch, "OCAL: An Abstraction for Host-Code Programming with OpenCL and CUDA.", *ICPADS'18*
[6] Rasch, Bigge, Wrodarczyk, Schulze, Gorlatch. "dOCAL: High-Level Distributed Programming with OpenCL and CUDA.", JOS'19

Transformation: PM → MDH

Polyhedral Model (PM) is "structured" representation of the sequential code

isl [7]

```
for( int i = 0; i < M ; ++i )
  for( int j = 0; i < N ; ++j )
    for( int k = 0; i < K ; ++k )
      C[i][j] += A[i][k] * B[k][j];
```

Matrix Multiplication in C

```
md_hom( f, (++,++,?) ) o view( A,B )( i,j,k )( A[i,k], B[k,j] )
```

means: *Unknown Combine Operator (UCO)*
→ NO parallelization, BUT tiling, caching, ...

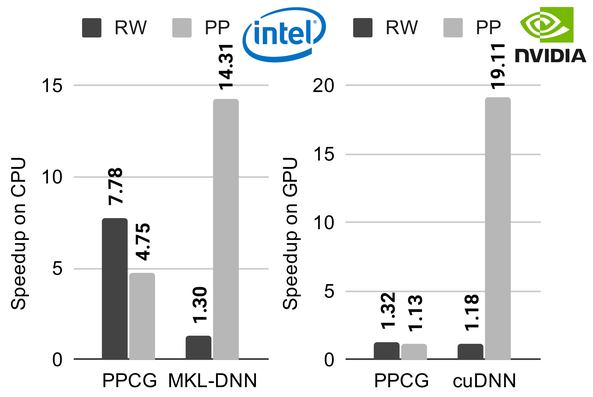
```
T f(T A_i_k, T B_k_j, T C_i_j)
{
  C_i_j += A_i_k * B_k_j;

  return C_i_j;
}
```

- Variables with `read` or `read-write` access are set as arguments of `f`.
- Variables with `write` access are declared and zero initialized in `f`.
- Variables with `write` or `read-write` access are returned by `f`.

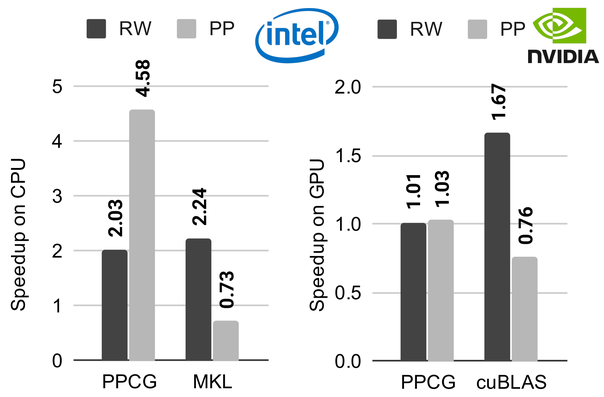
[7] Verdoolaege, "isl: An Integer Set Library for the Polyhedral Model", *ICMS'10*

Experimental Results

Hardware
- CPU: Intel Xeon E5
- GPU: NVIDIA V100



Gaussian Convolution



Matrix Multiplication

- **md_poly vs. PPCG:**
  - Competitive performance on GPU: `1.01x – 1.32x`
  - Better performance on CPU: `2.03x – 7.78x`
- **md_poly vs. Intel MKL/MKL-DNN & NVIDIA cuBLAS/cuDNN:**
  - Competitive and sometimes better performance: `0.73x – 2.24x`

Matrix Multiplication
- RW: M,N,K = 10,500,64
- PP: M,N,K = 1024

Gaussian Convolution
- RW: 1×512×7×7×512
- PP: 1x1x4096x4096x1