Code Generation & Optimization for Deep-Learning Computations on GPUs via Multi-Dimensional Homomorphisms

Richard Schulze, Ari Rasch, Sergei Gorlatch

SC21
St. Louis, MO | science & beyond.

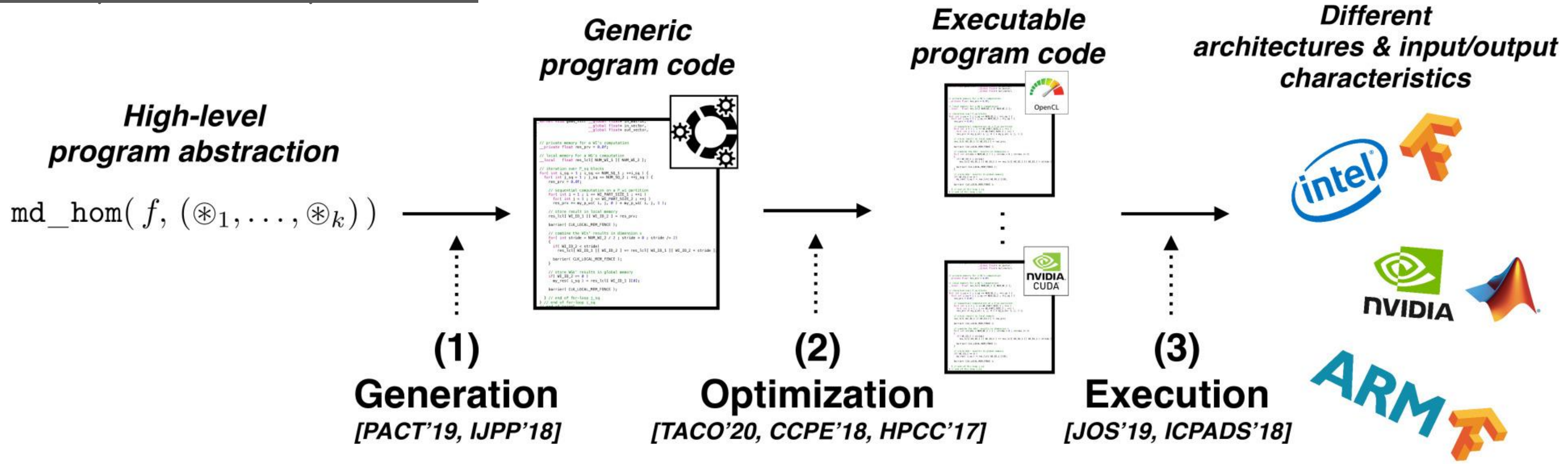
Introduction

We present our work-in-progress code generation and optimization approach for Deep Learning (DL) computations:

- based on our approach of **Multi-Dimensional Homomorphisms (MDH)** *[IJPP'18]*
- achieves **high performance** for popular DL computations by exploiting the already existing MDH GPU code generation *[PACT'19]* & optimization *[TACO'20]* & execution *[JOS'19]* approach
- **more expressive** than the state-of-the-art DL abstractions (e.g., as provided by TensorFlow): we show that MDH can express multiple DL computations as a single MDH expression, enabling optimization across computations (parallelization, tiling, etc.)

Excursion: MDH in a Nutshell

A holistic approach toward automatic code *generation* & *optimization* & *execution* for *data-parallel computations*:



- We **formally define data-parallel computations** (linear algebra routines (BLAS), convolutions, ...) as **Multi-Dimensional Homomorphisms (MDHs)**.
- We enable **conveniently** implementing MDHs by providing a **high-level DSL** for them.
- We provide a **DSL compiler** for automatically **generating executable low-level code** (CUDA, etc) -- the code is **fully automatically optimized** (auto-tuned) for the target device and data characteristics (size, layout, etc).

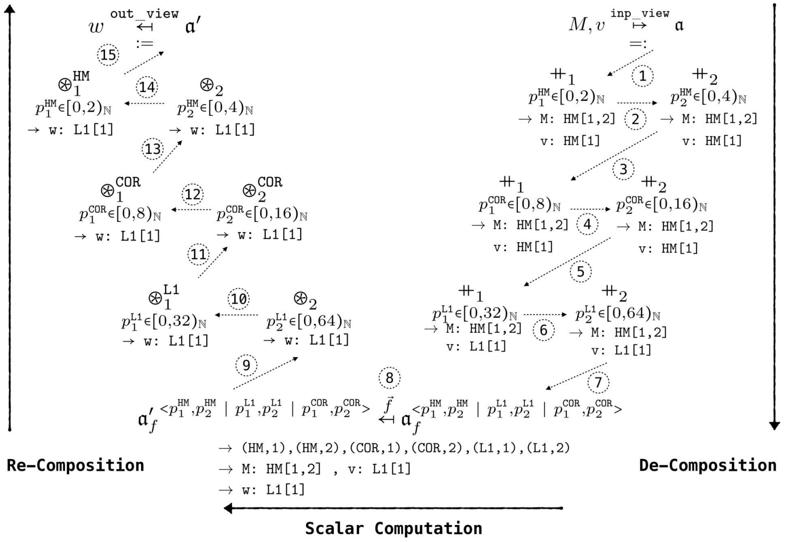
Excursion: MDH in a Nutshell

Behind the scenes:

```
MatVec<T∈TYPE|I,K∈ℕ> :=

out_view<T>( w:(i,k)↦(i) ) ∘

md_hom<I,K>( *, (++,+) ) ∘

inp_view<T,T>( M:(i,k)↦(i,k) , v:(i,k)↦(k) )
```

→

formally sound, auto-tunable lowering process



High-Level MDH Representation

- Expresses *what* to compute, via algebraic higher-order functions
- Agnostic from hardware and optimization details

Low-Level MDH Representation

- Expresses *how* to compute, by explicitly expressing (de-)composition of computations
- straightforwardly transformable to executable program code

Excursion: MDH in a Nutshell

The MDH high-level representation at example *Matrix Multiplication (MatMul)*:

MDH needs exactly three higher-order functions (patterns) to express data-parallel computations:

$$\texttt{MatMul}^{<T\in\texttt{Type}\,|\,M,N,K\in\mathbb{N}>} :=$$

$$\texttt{out\_view}^{<\dots>}(\dots)\ \circ \qquad \textbf{(c)}$$

$$\texttt{md\_hom}^{<\dots>}(\dots)\ \circ \qquad \textbf{(b)}$$

$$\texttt{inp\_view}^{<\dots>}(\dots) \qquad \textbf{(a)}$$

MDH pattern instances for MatMul:

(a) $$\texttt{inp\_view}^{<\texttt{T,T}>}(\ \texttt{A}:(\texttt{i,j,k})\mapsto(\texttt{i,k})\ ,\ \texttt{B}:(\texttt{i,j,k})\mapsto(\texttt{k,j})\ )$$

scalar types of input matrices
input matrices
indices for input matrices
data accesses

(b) $$\texttt{md\_hom}^{<\texttt{M,N,K}>}(\ *,\ (\mathbin{+\!\!+}_1,\ \mathbin{+\!\!+}_2,\ +)\ )$$

dimension sizes
multiply elements in A and B
concatenate in dimensions i & j
sum in dimension k

(c) $$\texttt{out\_view}^{<\texttt{T}>}(\ \texttt{C}:(\texttt{i,j,k})\mapsto(\texttt{i,j})\ )$$

scalar types of output matrix
output matrix
indices for output matrix
data accesses

Excursion: MDH in a Nutshell

Important functions can naturally be expressed as MDHs:

Linear Algebra

```
MatMul<...> = out_view<...>( ... ) o md_hom<...>( *, (++, ++, +) ) o inp_view<...>( ... )
MatVec<...> = out_view<...>( ... ) o md_hom<...>( *, (++, +)     ) o inp_view<...>( ... )
DOT<...>    = out_view<...>( ... ) o md_hom<...>( *, (+)         ) o inp_view<...>( ... )
```

Access neighboring elements within their input buffer

Stencil Computations

```
Gaussian_2D<...> = out_view<...>( ... ) o md_hom( f_G, (++, ++)     ) o inp_view<...>( ... )
Jacobi_3D<...>   = out_view<...>( ... ) o md_hom( f_J, (++, ++, ++) ) o inp_view<...>( ... )
```

Data Mining

```
PRL<...> = out_view<...>( ... ) o md_hom( weight, (++, Ⓧmax) ) o inp_view<...>( ... )
```

Access user-defined combine operator that operates on user-defined data type

Often very high dimensional (e.g., 7 dims)

Tensor Contractions

```
TC<...> = out_view<...>( ... ) o md_hom( *, (++,…,++ , +,…,+) ) o inp_view<...>( ... )
```

Further examples: MLP, SVM, ECC, …, Mandelbrot, Parallel Reduction, ...

Excursion: MDH in a Nutshell

Stencils

CPU	Gaussian (2D)		Jacobi (3D)	
	RW	PC	RW	PC
Lift [2]	4.90	5.96	1.94	2.49
MKL-DNN	6.99	14.31	N/A	N/A

GPU	Gaussian (2D)		Jacobi (3D)	
	RW	PC	RW	PC
Lift [2]	2.33	1.09	1.14	1.02
cuDNN	3.78	19.11	N/A	N/A

[2] Hagedorn et. al, "High Performance Stencil Code Generation with LIFT.", **CGO'18 (Best Paper Award)**.

MDH proved in previous work to achieve high performance on CPUs & GPUs [1]

[1] Rasch, Schulze, Gorlatch. "Generating Portable High-Performance Code via Multi-Dimensional Homomorphisms.", PACT'19

Linear Algebra

CPU	GEMM		GEMV	
	RW	PC	RW	PC
Lift [6]	fails	3.04	1.51	1.99
MKL	4.22	0.74	1.05	0.87

GPU	GEMM		GEMV	
	RW	PC	RW	PC
Lift [6]	4.33	1.17	3.52	2.98
cuBLAS	2.91	0.83	1.03	1.00

[6] Steuwer et. al, "Lift: A Functional Data-Parallel IR for High-Performance GPU Code Generation", **CGO'17**.

Tensor Contractions

GPU	Tensor Contractions								
	RW 1	RW 2	RW 3	RW 4	RW 5	RW 6	RW 7	RW 8	RW 9
COGENT [3]	1.26	1.16	2.12	1.24	1.18	1.36	1.48	1.44	1.85
F-TC [4]	1.19	2.00	1.43	2.89	1.35	1.54	1.25	2.02	1.49

[3] Kim et. al. "A Code Generator for High-Performance Tensor Contractions on GPUs.", **CGO'19**.

[4] Vasilache et al. "The Next 700 Accelerated Layers: From Mathematical Expressions of Network Computation Graphs to Accelerated GPU Kernels, Automatically.", ***TACO'19***.

Data Mining

CPU	Probabilistic Record Linkage					
	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}
EKR [5]	1.87	2.06	4.98	13.86	28.34	39.36

[5] Forchhammer et al. "Duplicate Detection on GPUs.", **HFSL'13**.

Goal of this Poster

Can MDH also
express DL computations and achieve
good performance results for them?

→ **Our WIP results look encouraging**

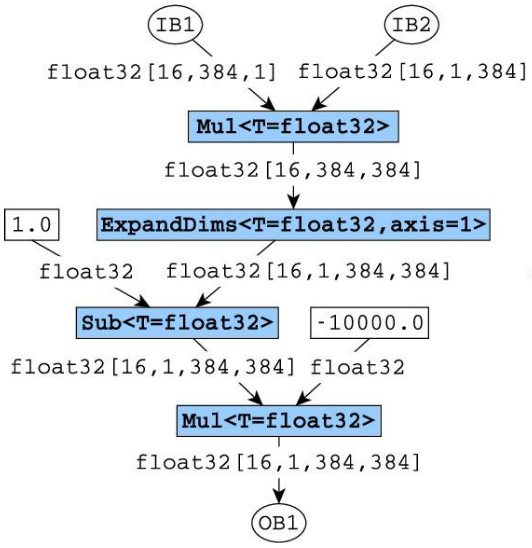
DL Computations Expressed in the MDH Formalism

Operator	out_view$^{<\ldots>}$	md_hom$^{<\ldots>}$		inp_view$^{<\ldots>}$
Mul$^{<\ldots>}$	$\mathtt{OB1}:(i,j) \mapsto (i,j)$	$*$	$+\!\!+,+\!\!+$	$\mathtt{IB1}:(i,j) \mapsto (i,j)$, $\mathtt{IB2}:(i,j) \mapsto (i,j)$
Sub$^{<\ldots>}$	$\mathtt{OB1}:(i,j) \mapsto (i,j)$	$-$	$+\!\!+,+\!\!+$	$\mathtt{IB1}:(i,j) \mapsto (i,j)$, $\mathtt{IB2}:(i,j) \mapsto (i,j)$
ExpandDims$^{<\mathrm{axis},D\in\mathbb{N} \mid \ldots>}$	$\mathtt{OB1}:(i_1,\ldots,i_D) \mapsto (\ldots,i_{\mathrm{axis}-1},0,i_{\mathrm{axis}},\ldots)$	id	$+\!\!+,\ldots,+\!\!+$	$\mathtt{IB1}:(i_1,\ldots,i_D) \mapsto (i_1,\ldots,i_D)$
BiasAddGrad$^{<\mathrm{NHWC} \mid \ldots>}$	$\mathtt{OB1}:(i,j) \mapsto (j)$	id	$+\ ,\ +\!\!+$	$\mathtt{IB1}:(i,j) \mapsto (i,j)$
BatchMatMul$^{<N,N \mid \ldots>}$	$\mathtt{OB1}:(b1,b2,i,j,k) \mapsto (b1,b2,i,j)$	$*$	$+\!\!+,\ldots,+\!\!+\ ,\ +$	$\mathtt{IB1}:(b1,b2,i,j,k) \mapsto (b1,b2,i,k)$, $\mathtt{IB2}:(b1,b2,i,j,k) \mapsto (b1,b2,k,j)$

Popular DL computations[1] are conveniently expressed in the MDH formalism.

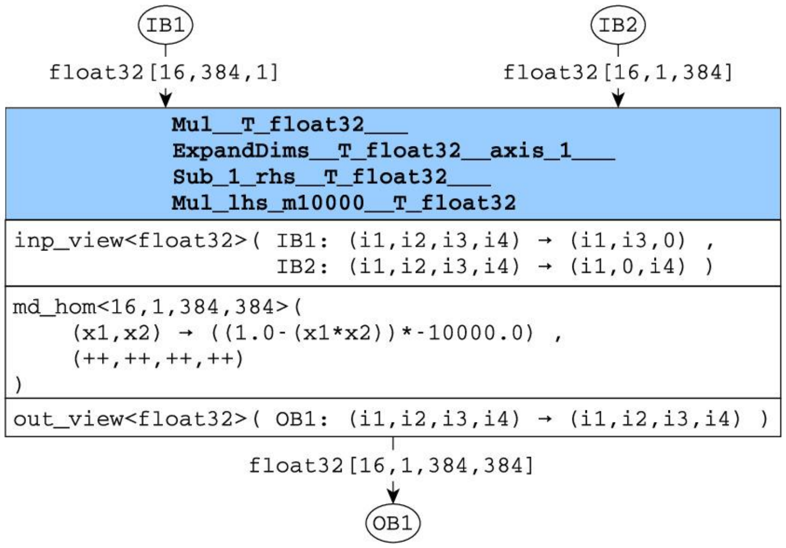
[1] Taken from the TensorFlow implementation of the real-world BERT neural network.

DL Computations Expressed in the MDH Formalism



BERT Subgraph
in **TensorFlow**

MDH can express multiple DL comp. as a single MDH expr.



BERT Subgraph
in **MDH**

Experimental Results

NVIDIA TensorFlow tvm

2.9x faster than **TVM** for **BiasAddGrad**

1.5x faster than **TensorFlow** for **BiasAddGrad**

1.1x faster than **TVM** for **BatchMatMul**

3.8x faster than **TVM** for a **subgraph of BERT**

Our preliminary experimental results on NVIDIA V100 GPU show that we can achieve **better performance** than well-performing **machine-** and **hand-optimized** approaches on real-world data sizes taken from the BERT neural network.

1.9x faster than **TC** for **BatchMatMul**

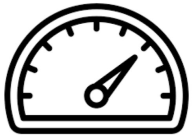
1.7x faster than **TC** for a **subgraph of BERT**

1.7x faster than **TC** for **BiasAddGrad**

4.9x faster than **TensorFlow** for a **subgraph of BERT**

Conclusion

MDH for DL— advantages we see:



Performance

encouraging WIP results

Portability

MDH targets also CPUs, etc.

Productivity

encouraging WIP results

Future Work:

- Automatizing **“DL-subgraph-to-MDH-node” process**, by exploiting MDHs’ formal properties;
- Targeting **sparse computations**;
- Analyzing MDH for DL on **further architectures** (CPU, etc);
- …

Thank you for listening!

Richard Schulze
r.schulze@uni-muenster.de

Ari Rasch
a.rasch@uni-muenster.de

Code Artifact available: https://gitlab.com/mdh-project/sc21_poster

Excursion: MDH in a Nutshell

Important functions can naturally be expressed as MDHs:

Linear Algebra

```
MatMul<T|M,N,K> = out_view<T>( C: (i,j,k) -> (i,j) ) o md_hom<M,N,K>( *, (++, ++, +) ) o inp_view<T,T>( A: (i,j,k) -> (i,k) ,
                                                                                                         B: (i,j,k) -> (k,j) )
MatVec<...> = out_view<...>( ... ) o md_hom<...>( *, (++, +)     ) o inp_view<...>( ... )
DOT<...>    = out_view<...>( ... ) o md_hom<...>( *, (+)         ) o inp_view<...>( ... )
```

Stencil Computations

```
Gaussian_2D<T|I,J> = out_view<T>( OUT: (i,j) -> (i,j) ) o md_hom( f_G, (++, ++) ) o inp_view<T>( IMG: (i,j) -> (i+0,j+0),
                                                                                                     (i,j) -> (i+1,j+0),
                                                                                                     ...,
                                                                                                     (i,j) -> (i+2,j+2) )
Jacobi_3D<...>   = out_view<...>( ... ) o md_hom( f_J, (++, ++, ++) ) o inp_view<...>( ... )
```

Data Mining

```
PRL<...> = out_view<...>( ... ) o md_hom( weight, (++, ⊗max) ) o inp_view<...>( ... )
```

Tensor Contractions

```
TC<...> = out_view<...>( ... ) o md_hom( *, (++,…,++ , +,…,+) ) o inp_view<...>( ... )
```

Further examples: MLP, SVM, ECC, …, Mandelbrot, Parallel Reduction, ...

5:45 pm CST: Poster Presentation
5:55 pm CST: Q&A

Please also join us tomorrow at 10:30 am CST in Room 229, session “Best Research Poster Presentations”

Deep-Learning Computations on GPUs via Multi-Dimensional Homomorphisms

Richard Schulze, Ari Rasch, Sergei Gorlatch

Thank you for listening!

Richard Schulze
r.schulze@uni-muenster.de

Ari Rasch
a.rasch@uni-muenster.de

Please also join us tomorrow at 10:30 am CST in Room 229, session "Best Research Poster Presentations"