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

Code Artifact and Video Narration.

Richard Schulze, Ari Rasch, Sergei Gorlatch

WWU MÜNSTER

Introduction

We present our work-in-progress **code generation and optimization approach** for **DL computations**:

- based on the formalism of **Multi-Dimensional Homomorphisms (MDH)** [1]
- achieves **high-performance** for popular DL computations by exploiting the already existing MDH GPU code generation and optimization approach
- **more expressive** than the state-of-the-art DL abstractions (e.g., as provided by TensorFlow): we are capable of expressing multiple DL computations as a single MDH expression

[1] Rasch, Gorlatch, "Multi-Dimensional Homomorphisms and Their Implementation in OpenCL", IJPP'18

The MDH Formalism

$$\texttt{MatMul}^{<\texttt{T}\in\texttt{Type}\,|\,\texttt{M},\texttt{N},\texttt{K}\in\mathbb{N}>} := \texttt{out\_view}^{<\dots>}(\dots)\circ \quad \textbf{(c)}$$
$$\texttt{md\_hom}^{<\dots>}(\dots)\circ \quad \textbf{(b)}$$
$$\texttt{inp\_view}^{<\dots>}(\dots) \quad \textbf{(a)}$$

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

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

MDH allows us conveniently expressing data-parallel computations and automatically generate CUDA code for them [2, 3].

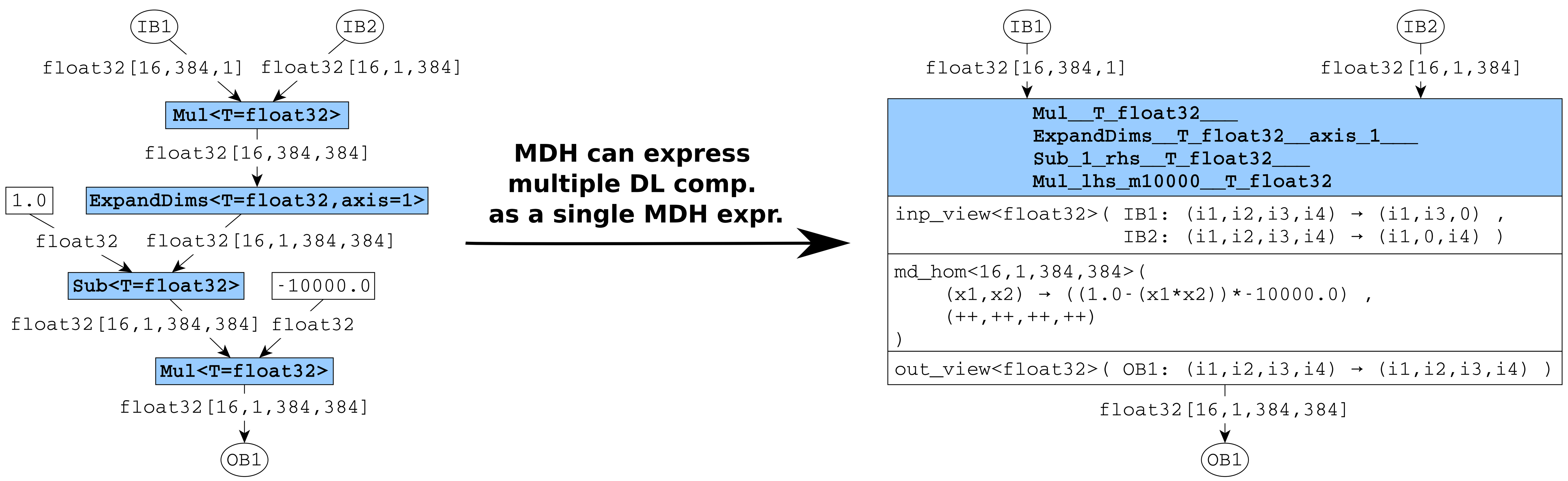
[2] Rasch, Schulze, Gorlatch, "Generating Portable High-Performance Code via Multi-Dimensional Homomorphisms", PACT'19

[3] Rasch, Schulze, Steuwer, Gorlatch, "Efficient Auto-Tuning of Parallel Programs with Interdependent Tuning Parameters via Auto-Tuning Framework (ATF)", TACO'21

DL Computations Expressed in the MDH Formalism

Operator	out_view$^{<\dots>}$	md_hom$^{<\dots>}$	inp_view$^{<\dots>}$
Mul$^{<\dots>}$	OB1:(i,j) $\mapsto$ (i,j)	$*$, $(+\!\!+_1, +\!\!+_2)$	IB1:(i,j) $\mapsto$ (i,j) , IB2:(i,j) $\mapsto$ (i,j)
Sub$^{<\dots>}$	OB1:(i,j) $\mapsto$ (i,j)	$-$, $(+\!\!+_1, +\!\!+_2)$	IB1:(i,j) $\mapsto$ (i,j) , IB2:(i,j) $\mapsto$ (i,j)
ExpandDims$^{<\text{axis},D\in\mathbb{N}\,|\,\dots>}$	OB1:$(i_1,\dots,i_D) \mapsto (\dots,i_{\text{axis}-1},0,i_{\text{axis}},\dots)$	id , $(+\!\!+_1,\dots+\!\!+_D)$	IB1:$(i_1,\dots,i_D) \mapsto (i_1,\dots,i_D)$
BiasAddGrad$^{<\text{NHWC}\,|\,\dots>}$	OB1:(i,j) $\mapsto$ (j)	id , $(+, +\!\!+_2)$	IB1:(i,j) $\mapsto$ (i,j)
BatchMatMul$^{<\text{N,N}\,|\,\dots>}$	OB1:(b1,b2,i,j,k) $\mapsto$ (b1,b2,i,j)	$*$, $(+\!\!+_1,\dots,+\!\!+_4, +)$	IB1:(b1,b2,i,j,k) $\mapsto$ (b1,b2,i,k) , IB2:(b1,b2,i,j,k) $\mapsto$ (b1,b2,k,j)

Popular DL computations¹ are conveniently expressed in the MDH formalism.



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

Experimental Results

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.

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

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

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

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