Towards Compositional and Generative Tensor Optimizations
Adilla Susungi, Norman Rink, Jerónimo Castrillón, Immo Huismann, Albert Cohen, Claude Tadonki, Jörg Stiller, Jochen Fröhlich

To cite this version:

HAL Id: hal-01666818
https://hal-mines-paristech.archives-ouvertes.fr/hal-01666818
Submitted on 18 Dec 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Towards Compositional and Generative Tensor Optimizations

Adilla Susungi, Norman A. Rink, Jerónimo Castrillón, Immo Huismann, Albert Cohen, Claude Tadonki, Jörg Stiller and Jochen Fröhlich

adilla.susungi@mines-paristech.fr — norman.rink@tu-dresden.de

Tensors in Computational Fluid Dynamics (CFD)

- Loop characteristics:
  - 3 to 4 dimensions nesting
  - Few iterations per dimension (e.g., 17 or 33 iterations)
- Type of computations:
  - Tensor contractions
  - Outer products
  - Element-wise multiplications
- Computations on each element of a structured mesh

Inverse Helmholtz

\[ t_{ijk} = \sum_{l,m,n} A_{kn}^T \cdot A_{jm}^T \cdot A_{il}^T \cdot p_{lmn} \]

\[ p_{ijk} = D_{ijk} \cdot t_{ijk} \]

\[ v_{ijk} = \sum_{l,m,n} A_{kn} \cdot A_{jm} \cdot A_{il} \cdot p_{lmn} \]

Tensor Optimization Frameworks

- Domain-specific expressivity
- Flexible/Adaptive optimization heuristics
- Hidden and/or rigid optimization heuristics

Generic expressivity

Optimizing CFD Kernels with Existing Tools

- Several limitations
- Few opportunities for adaptations

Goal

A cross-domain intermediate language for tensor optimizations

Flexible/adaptive

Hidden/rigid

Specific

Generic

Related Work

Different levels of expressiveness and control on optimizations

Flexible/adaptive

Hidden/rigid

Specific

Generic

Intermediate Language

- Modular constructs
- First-class citizens:
  - Arrays
  - Tensor operators
  - Loop iterators
  - Transformations

Envisioned Tool

Meta-programming

Iterative search

Search Space Exploration

- Evaluation order of tensor contractions
- Fusions
- Permutations
- Vectorization
- Collapsing
- Unrolling

Inverse Helmholtz by Example

# Basic array declaration
A = array(2, double, [N, N])
u = array(3, double, [N, N, N])
D = array(3, double, [N, N, N])

# Transposition
At = vtranspose(A, 1, 2)

# Tensor contractions
tmp1 = contract(At, u, [2, 1])
tmp2 = contract(At, tmp1, [2, 2])
tmp3 = contract(At, tmp2, [2, 3])

# Iterator declaration
i1 = iterator(0, N, 1)
i2 = iterator(0, N, 1)

# ... other iterator declarations

# Association of iterators to computations
build(D, [td1, td2, td3])
build(tmp1, [i1, i2, i3, i4])
# Also applies to tmp3, ... tmp6
build(v, [k1, k2, k2, k32, k42])

# Loop interchanges
interchange(i4, i3)
interchange(i4, i2)
interchange(j2, j1)
interchange(j1, j4)

# Transpositions
tmp2t = vtranspose(tmp2, 1, 2)
replace_array(j3, tmp2, tmp2t)
replace_array(k4, tmp3, tmp3t)
interchange(k3, k2)
# ... other optimizations

Example of assessment: Different heuristics of loop interchanges (+ parallelization)

Variant L1: Loop interchanges only;
Variant L2: Loop interchanges + data transpositions with copying;
Variant L3: Loop interchanges + data transpositions without copying.

Baseline: sequential execution (3.32s). Machine: 24-core Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz (Haswell)

Future Work

- Applications to other domains
- Syntax refinement
- Formal semantics

This work was partially funded by the German Research Council (DFG) through the Cluster of Excellence ‘Center for Advancing Electronics Dresden’ (cfaed) and by PSL Research University through the ACOPAL project.