



HAL
open science

Meta-programming for Cross-Domain Tensor Optimizations

Adilla Susungi, Norman A. Rink, Jeronimo Castrillon, Claude Tadonki

► **To cite this version:**

Adilla Susungi, Norman A. Rink, Jeronimo Castrillon, Claude Tadonki. Meta-programming for Cross-Domain Tensor Optimizations. conference on Systems, Programming, Languages and Applications: Software for Humanity (SPLASH), Nov 2018, Boston, United States. hal-01939535

HAL Id: hal-01939535

<https://minesparis-psl.hal.science/hal-01939535>

Submitted on 29 Nov 2018

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.

Meta-programming for Cross-Domain Tensor Optimizations

Adilla Susungi, Norman A. Rink, Albert Cohen, Jerónimo Castrillón, Claude Tadonki

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

TeML Grammar

```
(program) ::= (stmt) (program)
           | ε
(stmt) ::= (id) = (expression)
         | (id) = @(id): (expression)
         | codegen ((ids))
         | init (...)
(expression) ::= (Texpression)
              | (Lexpression)
(Texpression) ::= tensor ((ints))
              | eq ((id), (iters)? → (iters))
              | vop ((id), (id), [(iters)?, (iters)?])
              | op ((id), (id), [(iters)?, (iters)?] → (iters))
(Lexpression) ::= build ((id))
              | stripmine ((id), (int), (int))
              | interchange ((id), (int), (int))
              | ...
(iters) ::= [(ids)]
(ids) ::= (id) (, (id))*
(ints) ::= (int) (, (int))*
```

Code Sample

```
# -- Begin program specification
w = tensor(double, [13])
u = tensor(double, [13, 13, 13])
L = tensor(double, [13, 13])
M_ = outerproduct([w, w, w])
Lh = div(L, w, [[i1, i2], [i2]] → [i1,
→ i2])
M = entrywise_mul(M_, u)
r1 = contract(Lh, M, [[2, 1]])
r2 = contract(Lh, M, [[2, 2]])
r3 = contract(Lh, M, [[2, 3]])
# -- End program specification
# Code generation with loop fusions only
codegen([11, 12, 13, 14, 15, 16])
# Code generation with fusion, parallelism
and vectorization
19 = parallelize(l1, 1, None)
110 = parallelize(12, 1, None)
111 = parallelize(13, 1, None)
112 = parallelize(18, 1, None)
113 = vectorize(19, 3)
114 = vectorize(110, 2)
115 = vectorize(111, 3)
codegen([113, 114, 115, 112])
```

Design and semantics of a tensor optimization meta-language

- Functional transformation meta-language with high-level abstractions for tensor computations to enable the composition of different type of transformations (e.g. loop, algebraic or layout transformations).
- Formal specification of tensor operations and loop transformations using denotational semantics.
- Examples of use: empirical tuning engines, meta-programming optimizations by experts.

Semantics Foundations

Domains and state

$$\mathbf{T} = \{ \langle (op, S, I), ts \rangle \mid (ts = []) \vee (ts = [t_1, \dots, t_k] \wedge t_i \in \mathbf{T}) \}$$

$$\mathbf{L} = \{ \langle id, [x_1, \dots, x_k] \rangle \mid x_i \in \mathbf{L} \cup \mathbf{T} \}$$

$$\sigma : identifier \rightarrow (\mathbf{T} + \mathbf{L})$$

Valuation functions

$$\mathcal{P}_{prog}[S p] = \mathcal{P}_{prog}[p] \circ \mathcal{P}_{stmt}[S]$$

$$\mathcal{E}_t[\text{tensor}(S)] = \lambda\sigma. \langle (\square, S, \epsilon), [] \rangle$$

$$\mathcal{E}_t[\text{eq}(t, I_0 \rightarrow I_1)] =$$

$$\lambda\sigma. \text{let } \langle (op, S, I'), ys \rangle = \sigma(t)$$

$$y = \langle (op, S, I''), ys \rangle$$

$$x = \langle (\square, S', I_1), [] \rangle$$

$$\text{in } \langle (\bullet, \bullet, \bullet), [x, y] \rangle,$$

$$\text{where } \begin{cases} I' \neq \epsilon \wedge I'' = I', & \text{if } I_0 = \epsilon \\ I' = \epsilon \wedge I'' = I_0, & \text{if } I_0 \neq \epsilon \end{cases}$$

$$\mathcal{E}_t[\text{build}(t)] = \lambda\sigma. \text{let } r = \text{"number of iterators in } \sigma(t)\text{"}$$

$$i_k = (0, ub_k, 1) \text{ for } k = 1, \dots, r$$

$$\text{in } \langle i_1, \dots, \langle i_r, [\sigma(t)] \rangle \dots \rangle,$$

$$\text{where } \sigma(t) = \langle (\bullet, \bullet, \bullet), [x, y] \rangle$$

$$\mathcal{E}_t[\text{stripmine}(l, r, v)] =$$

$$\lambda\sigma. \text{let } \langle i_1, \dots, \langle i_r, xs \rangle \dots \rangle = \sigma(l)$$

$$(b, e, 1) = i_r$$

$$i'_r = (0, (e - b)/v - 1, 1)$$

$$i'_{r+1} = (b + v \cdot i'_r, b + v \cdot i'_r + (v - 1), 1)$$

$$\text{in } \langle i_1, \dots, \langle i'_r, [\langle i'_{r+1}, xs \rangle] \rangle \dots \rangle$$

$$\mathcal{E}_t[\text{interchange}(l, r_1, r_2)] =$$

$$\lambda\sigma. \text{let } \langle i_1, \dots, \langle i_{r_1}, \dots, \langle i_{r_2}, xs \rangle \dots \rangle \dots \rangle = \sigma(l)$$

$$\text{in } \langle i_1, \dots, \langle i_{r_2}, \dots, \langle i_{r_1}, xs \rangle \dots \rangle \dots \rangle$$

Tensor Expressions

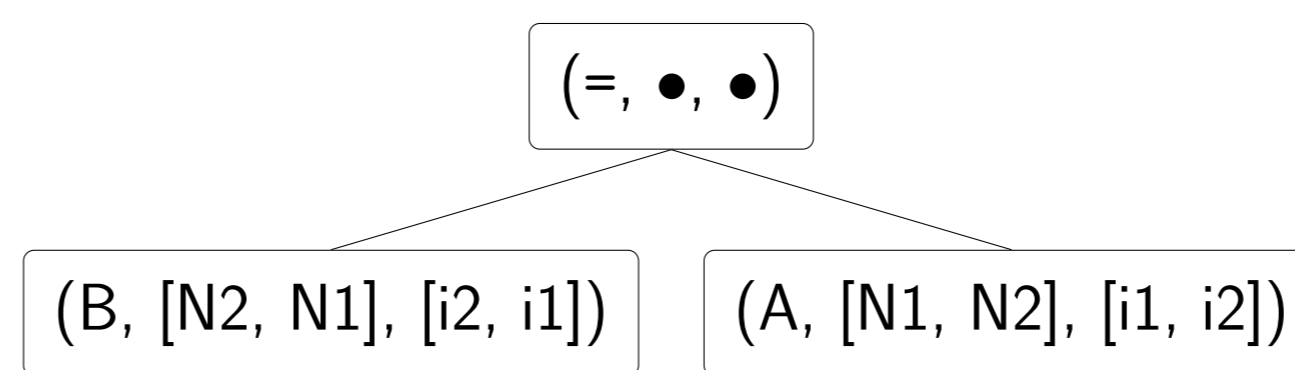
Matrix transposition

$$A = \text{tensor}([N1, N2])$$

$$B = \text{eq}(A, [i1, i2] \rightarrow [i2, i1])$$

$$\mathcal{E}_t[\text{build}(B)]\sigma_2 = \langle i1, [\langle i2, [\sigma_2(B)] \rangle] \rangle :$$

```
for (int i1 = 0; i1 <= (N1-1); i1++)
  for (int i2 = 0; i2 <= (N2-1); i2++)
    B[i2][i1] = A[i1][i2];
```



Compositions

Contraction

$$\mathcal{P}_{stmt}[t' = \text{contract}(t_0, t_1, [r_0, r_1])] = \mathcal{P}_{prog} \left[\begin{array}{l} t_2 = \text{vmul}(t_0, t_1, [I, J]) \\ t' = \text{add}(t', t_2, [I', \epsilon] \rightarrow I') \end{array} \right]$$

where

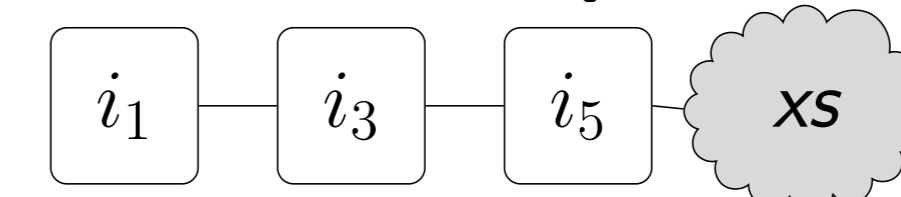
$$I = [i_0, \dots, i_{(r_0-1)}, k, i_{(r_0+1)}, \dots, i_{s_0}]$$

$$J = [j_0, \dots, j_{(r_1-1)}, k, j_{(r_1+1)}, \dots, j_{s_1}]$$

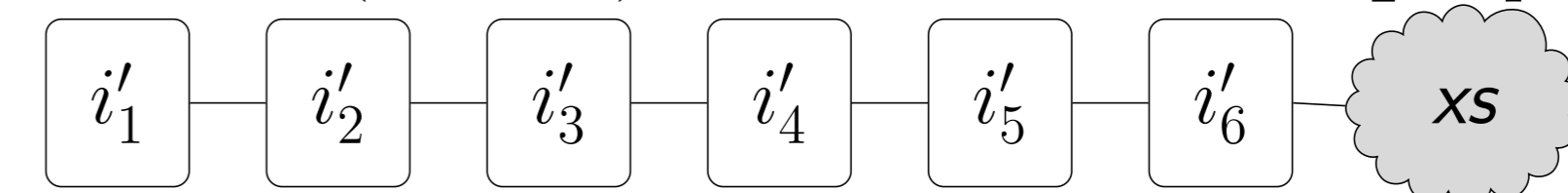
$$I' = (I \setminus \{k\}) \parallel (J \setminus \{k\})$$

Tiling

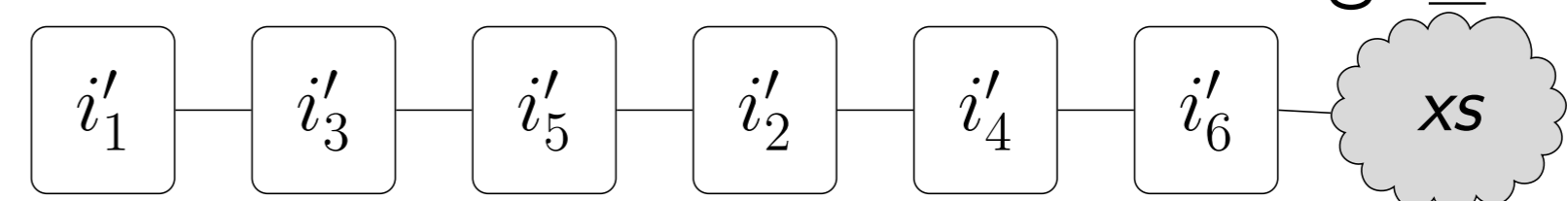
Initial loop nest



stripmine_n(, 3, v) has introduced $i'_2, i'_4,$ and i'_6

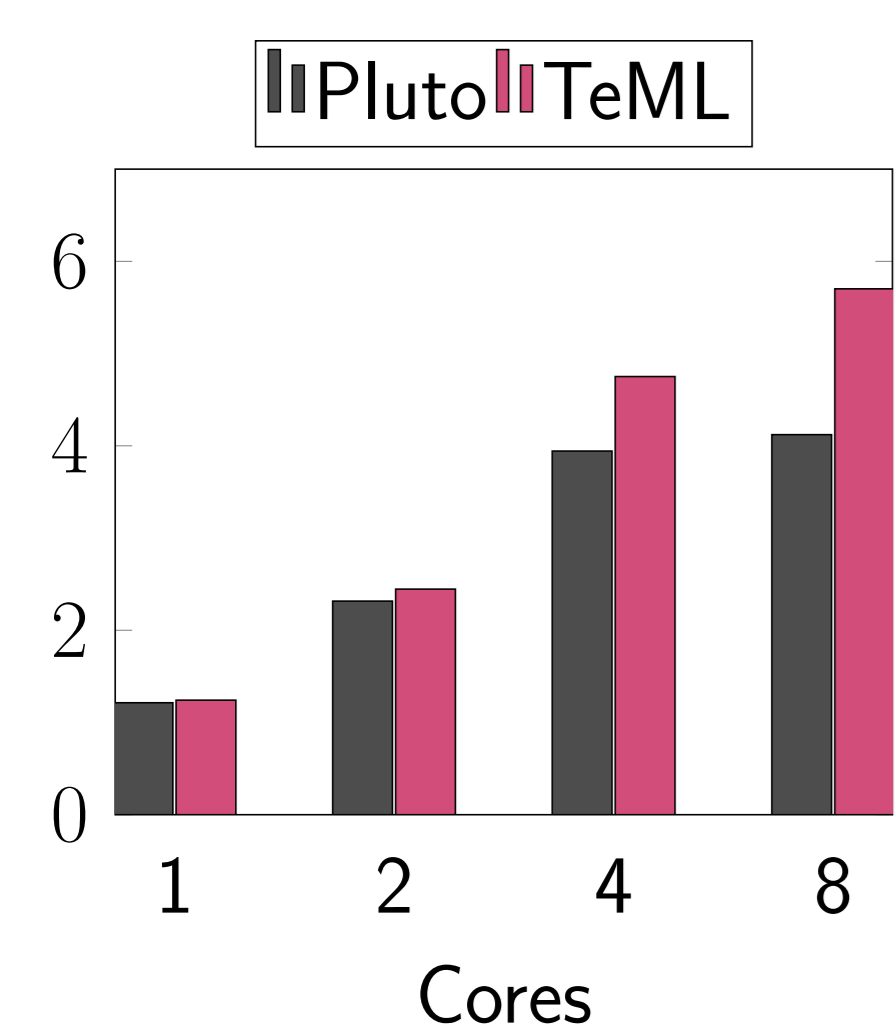


* After three times interchange_n.

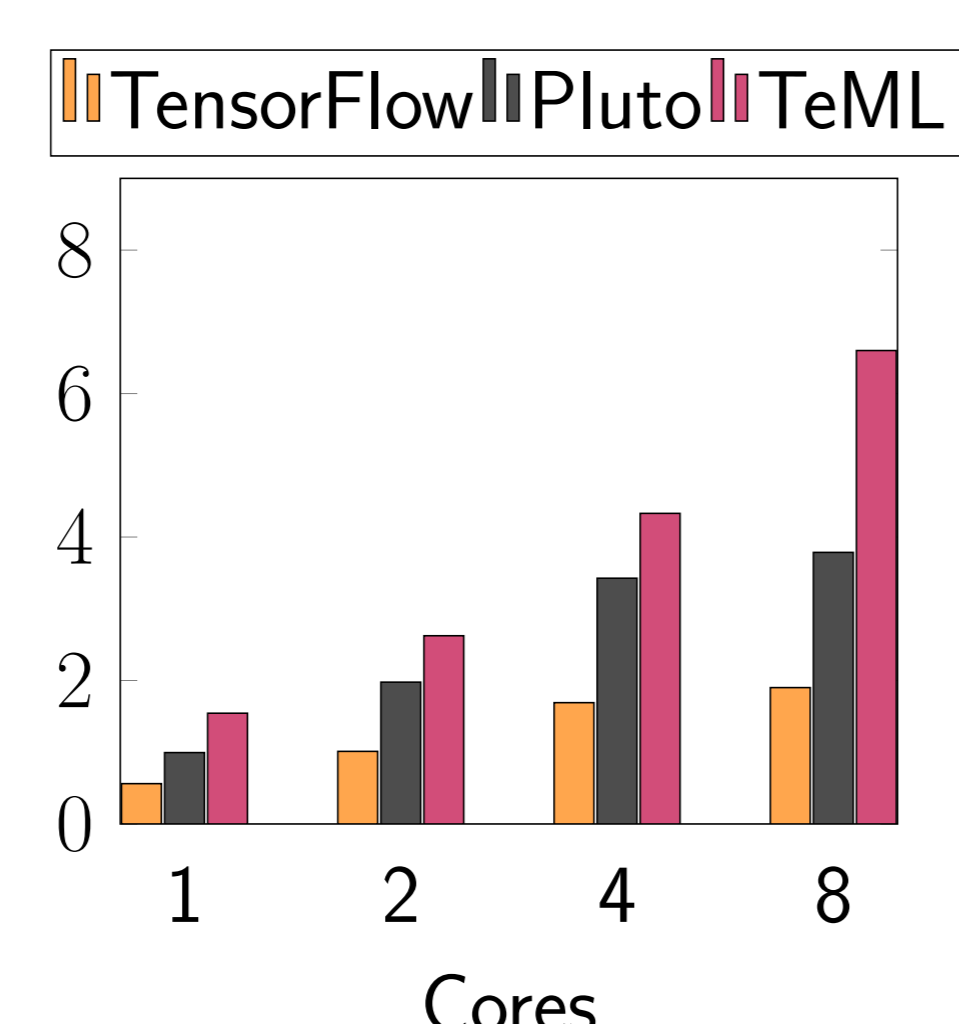


Experiments

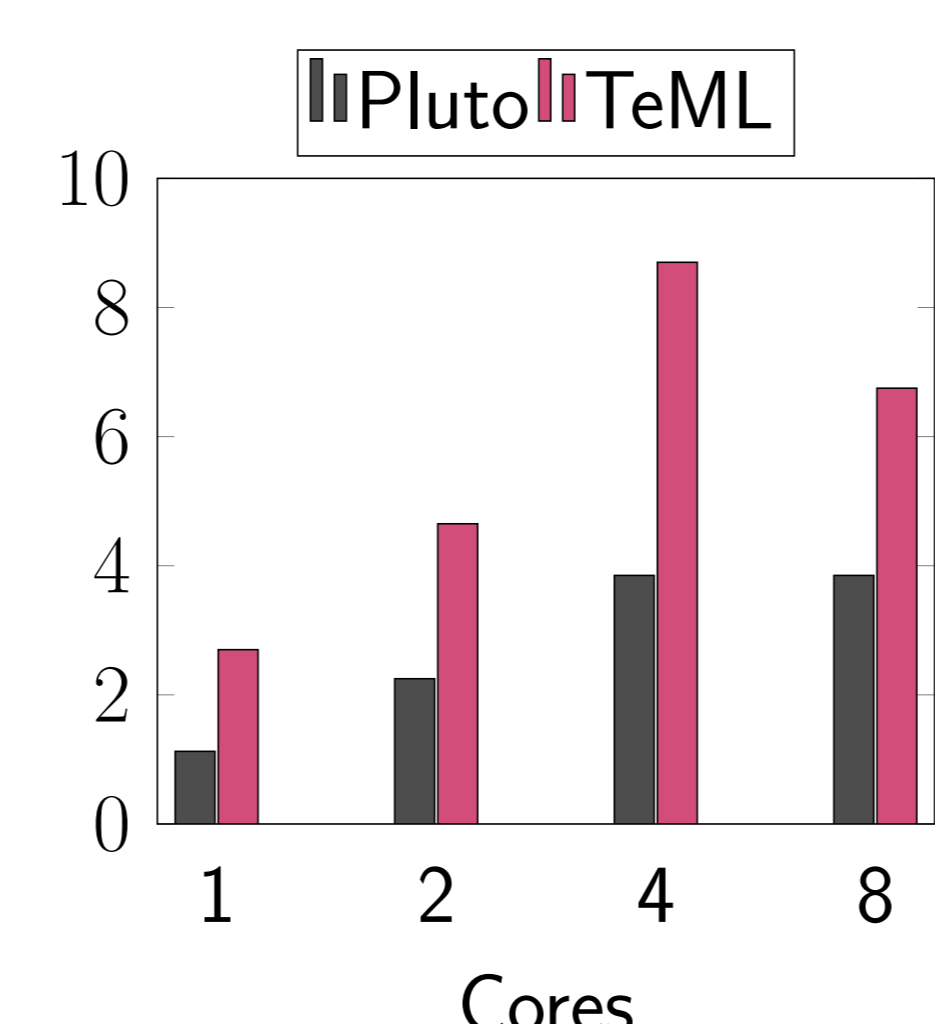
Helmholtz



Mttkrp



Grouped convolutions



- On Intel(R) Core(TM) i7-4910MQ CPU (2.90GHz, 8 hyperthreads, 8192KB of shared L3 cache), Ubuntu 16.04.
- Generated C programs compiled with the Intel C compiler ICC 18.02 (flags: -O3 -xHost -qopenmp)
- With TeML:
 - Pluto (v 0.11.4) optimizations reproducible
 - Capability to express better optimization paths

Future Work

- Abstractions for memory virtualization, stencil patterns, sparse tensors and corresponding semantics, extensions for parallelism support, type system

