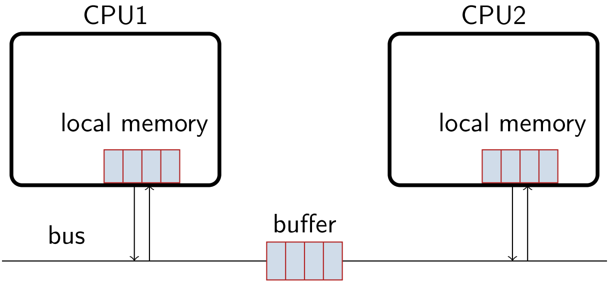
Rephrasing Polyhedral Optimizations with Trace Analysis

Hugo Thievenaz,
Christophe Alias and Keiji Kimura

12th International Workshop on Polyhedral Compilation Techniques
IMPACT'22

Plan

Context: Compile-time Storage Optimization



Challenges:

- Allocate local memory buffers: $local(i) \mapsto local_{opt}(\sigma_{local}(i))$
- Packing/unpacking of communicated data: $buffer(i) \mapsto buffer_{opt}(\sigma_{buffer}(i))$

Approach: Array Contraction

General approach: Contract temporary arrays (one-by-one) under the given scheduling constraints

- Focus: modular linear mappings $\sigma : \vec{i} \mapsto \vec{i} \bmod b(\vec{N})$
- Correctness: Conflicting cells are mapped to different locations:

$$\forall \vec{i} \bowtie \vec{j}, \overrightarrow{i} \neq \overrightarrow{j} \Rightarrow \sigma(\overrightarrow{i}) \neq \sigma(\overrightarrow{j})$$

- Efficiency: Minimize σ range

Approach: Array Contraction

General approach: Contract temporary arrays (one-by-one) under the given scheduling constraints

- Focus: modular linear mappings $\sigma : \vec{i} \mapsto \vec{i} \bmod b(\vec{N})$
- Correctness: Conflicting cells are mapped to different locations:

$$\forall \vec{i} \bowtie \vec{j}, \overrightarrow{i} \neq \overrightarrow{j} \Rightarrow \sigma(\overrightarrow{i}) \neq \sigma(\overrightarrow{j})$$

- Efficiency: Minimize σ range

State-of-the-art:

- Successive minima, *Lefebvre et al.*, 1997, $\vec{i} \mapsto \vec{i} \bmod b(N)$
- Admissible lattices, *Alias et al.*, 2007, $\vec{i} \mapsto A\vec{i} \bmod b(N)$
- Global array contraction, *Bhaskaracharya et al.*, 2016

Approach: Array Contraction

General approach: Contract temporary arrays (one-by-one) under the given scheduling constraints

- Focus: modular linear mappings $\sigma : \vec{i} \mapsto \vec{i} \bmod b(\vec{N})$
- Correctness: Conflicting cells are mapped to different locations:

$$\forall \vec{i} \bowtie \vec{j}, \overrightarrow{i} \neq \overrightarrow{j} \Rightarrow \sigma(\overrightarrow{i}) \neq \sigma(\overrightarrow{j})$$

- Efficiency: Minimize σ range

State-of-the-art:

- Successive minima, *Lefebvre et al.*, 1997, $\vec{i} \mapsto \vec{i} \bmod b(N)$
- Admissible lattices, *Alias et al.*, 2007, $\vec{i} \mapsto A\vec{i} \bmod b(N)$
- Global array contraction, *Bhaskaracharya et al.*, 2016

Goal: Reduce computational cost by using offline execution traces

Contributions

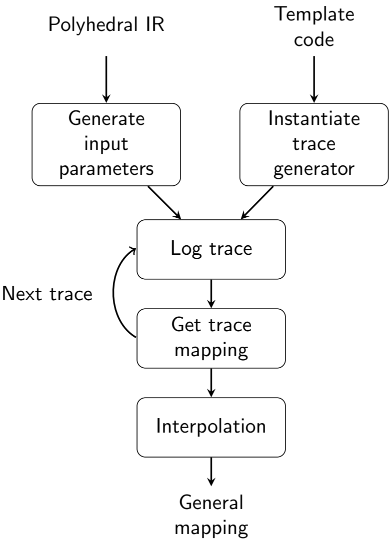
Lightweight array contraction using offline execution traces and interpolation

Correct-by-construction interpolation, no need for an oracle

Preliminary experimental validation on polyhedral benchmarks

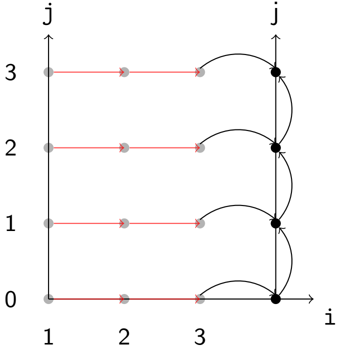
Plan

Overview


Polyhedral IR
Template code
Generate input parameters
Instantiate trace generator
Log trace
Next trace
Get trace mapping
Interpolation
General mapping

Input Parameters

```
for(i=1; i<N; i++)
  for(j=0; j<N; j++)
    a[i,j] =
        f(a[i-1,j]);
for(j=0; j<N; j++)
  result += a[N-1,j];
```



- Smallest N such that there exists one of each dependency ($N = 3$)
- Generation of next parameters, affinely independents ($N = 3, 4$)

Trace Analysis Algorithm

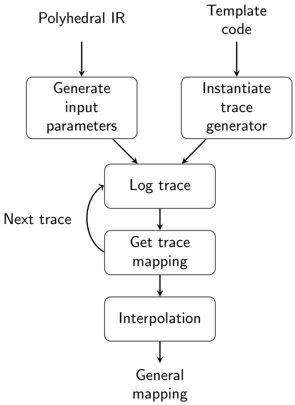
- Lefebvre-Feautrier instance on a trace $\rightarrow$ affine mapping instance
- Run on several instances to extrapolate an affine mapping

Trace Analysis Algorithm

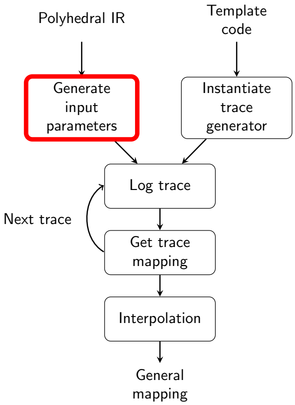
- Lefebvre-Feautrier instance on a trace → affine mapping instance
- Run on several instances to extrapolate an affine mapping

function GETMAPPING(T,a)
  Input: T : trace, a : array name
  Output: σ : scalar mapping
  $(In, Out) \leftarrow$ LIVENESS(T)
  $CS \leftarrow \bigcup_p \{(a[\vec{i}], a[\vec{j}]) \mid a[\vec{i}], a[\vec{j}] \in In(p)\}$
  $\Delta_a \leftarrow \{\vec{i} - \vec{j} \mid (a[\vec{i}], a[\vec{j}]) \in CS\}$
  for each array dimension i, starting from 0, in increasing order **do**
    $m_i \leftarrow 1 + max\{\delta_i \mid (0, ..., 0, \delta_i, ...) \in \Delta_a\}$
  end for
  return $\sigma : i \bmod \vec{m}$
end function

Running Example



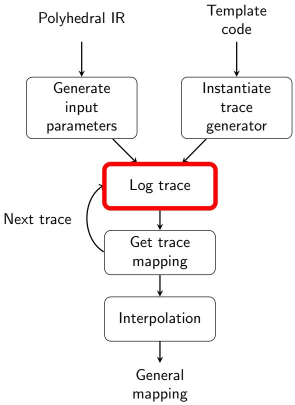
Running Example



$N = 3$

$N = 4$

Running Example



$N = 3$ $N = 4$

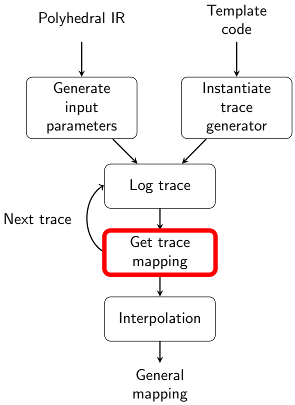
i = 1

W(1,**0**); W(**1**,1); W(1,**2**)

i = 2

R(1,0); W(**2**,0); . . .

Running Example



$N = 3$

$N = 4$

i = 1

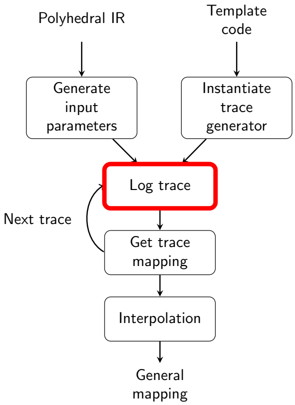
W(1,**0**); W(**1**,1); W(1,**2**)

i = 2

R(1,0); W(**2**,0); . . .

i mod 2, j mod 3

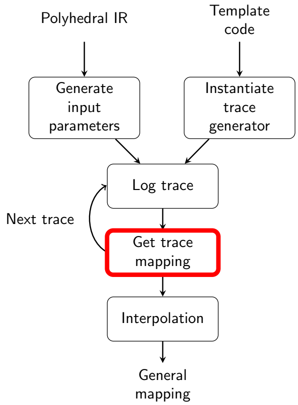
Running Example



$N = 3$	$N = 4$
i = 1	**i = 1**
W(1,**0**); W(**1**,1); W(1,**2**)	W(1,**0**); W(**1**,1); W(1,2); W(1,**3**);
i = 2	**i = 2**
R(1,0); W(**2**,0); ...	R(1,0); W(**2**,0); ...

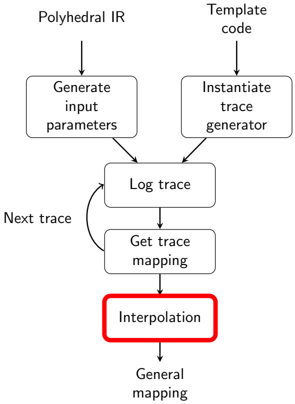
i mod 2, j mod 3

Running Example



$N = 3$	$N = 4$
i = 1	**i = 1**
W(1,**0**); W(**1**,1); W(1,**2**)	W(1,**0**); W(**1**,1); W(1,2); W(1,**3**);
i = 2	**i = 2**
R(1,0); W(**2**,0); ...	R(1,0); W(**2**,0); ...
$i \bmod 2$, $j \bmod 3$	$i \bmod 2$, $j \bmod 4$

Running Example



$N = 3$	$N = 4$
i = 1	**i = 1**
W(1,**0**); W(**1**,1); W(1,**2**)	W(1,**0**); W(**1**,1); W(1,2); W(1,**3**);
i = 2	**i = 2**
R(1,0); W(**2**,0); ...	R(1,0); W(**2**,0); ...
$i \bmod 2$, $j \bmod 3$	$i \bmod 2$, $j \bmod 4$

$$\boxed{\text{Mapping inferred: } (i, j) \mapsto (i \bmod 2, j \bmod N)}$$

Plan

Experimental Setup

Implementation: in C++ named PoLi, total of 1230 lines of code.

Benchmarks:

- **fibonacci**, example of the computation of the n-th term of the fibonacci sequence,
- **pc-2d** and **pc-2d-line**, two examples of a producer-consumer mechanic in two dimensions,
- **blur-2d**, an example of the 2D blur filter.

Baseline: Lefebvre-Feautrier algorithm (successive minima).

Setup: Intel Core i5-1135G7 CPU @ 2.40GHz, 16Gb RAM.

Experimental Results

Kernel	Mapping found	Parameters	PoLi time(ms)	LF time(ms)	**Speed-up**
fibonacci	$i \bmod 2$	$N = 2, 3$	0.00103	0.024221	**23.5**
pc-2d	$i \bmod N$ $j \bmod N$	$N = 2, 3$	0.00284	0.045513	**16.0**
pc-2d-line	$i \bmod 2$ $j \bmod N$	$N = 3, 4$	0.01022	0.064114	**6.3**
blur-2d	$y \bmod 3$ $x \bmod N$	$N = 5, 6$	0.15636	0.187037	**1.2**

Polytrace approach is lighter

- Expensive ILP $\mapsto$ max over a few points
- Tightest parameter selection is paramount

Plan

Conclusion

Contributions:

- New method for array contraction, based on a new **paradigm**
- Better performances with **small** trace **parameters**
- Scales better for examples with greater **dimensionality**

Future work:

- Contract the **global** array space all-at-once, to infer mappings of **smaller** memory footprint
- Select **smallest parameter instances**, gives smaller traces and so **faster** analysis
- Investigate new applications of this methodology to other **optimization** problems

Thank you !