Scalable Polyhedral Compilation, Syntax vs. Semantics: 1–0 in the First Round

IMPACT — January 22th 2020

Riyadh Baghdadi, MIT
Albert Cohen, Google

Polyhedral/Affine Scheduling

(Based on the Pluto algorithm [Bondhugula et al. 2008])
Iteratively produce affine schedule functions such that:

- dependence distances are *lexicographically* **positive**
- dependence distances are **small** ⇒ **temporal locality**
- dependence distances are **zero** ⇒ **parallelism**
- dependences have **non-negative distance** along **consecutive** dimensions
  ⇒ **permutability** (which enables tiling)

permutable	permutable	
(0,**1**,0,0)	(0,**1**,-2,3)	(0,0,**-1**,42)
valid	also valid	violated

Polyhedral/Affine Scheduling

(Based on the Pluto algorithm [Bondhugula et al. 2008])
Iteratively produce affine scheduling functions of the form

$$t_{S,k} = \vec{a} \cdot \vec{i} + \vec{b} \cdot \vec{P} + d$$

Statement S, scheduling step k
a,b,d – coefficients
i – original loop iterators
P – symbolic parameters

minimize $(t_{S,k} - t_{R,k})$

for every "proximity" dependence R⟶S
while enforcing dependence constraints

Polyhedral/Affine Scheduling

(Based on the Pluto algorithm [Bondhugula et al. 2008])
Iteratively produce affine scheduling functions of the form

$$t_{S,k} = \vec{a} \cdot \vec{i} + \vec{b} \cdot \vec{P} + d$$

Statement S, scheduling step k
a,b,d – coefficients
i – original loop iterators
P – symbolic parameters

minimize $(t_{S,k} - t_{R,k}) \leq \vec{u} \cdot \vec{P} + w$

for every “proximity dependence” R⟶S
while enforcing dependence constraints

$$\text{lexmin } \vec{u}, w, \vec{a}, \vec{b}, d \quad \text{s.t.} \quad \vec{u} \succ \vec{0}$$

use the affine form of the Farkas lemma to linearize the inequality

⟶ Integer Linear Programming (ILP) problem

State of the Art Scheduling Algorithm Template

[Zinenko et al. CC 2018]

- Multiple notions of “proximity”, including temporal and spatial locality
- Integrate parallelization as “optional constraints”
- Iterate on two parameterizable ILP problems
  - carry as little *spatial proximity* relations as possible and produce coincident dimensions for parallelism
    (based on the Pluto algorithm [Bondhugula et al. 2008])
  - carry multiple *spatial proximity* relations without skewing
    (based on the Feautrier algorithm [Feautrier 1992])
  - play with weights and reorder dimensions in lexicographic minimization

Scalability — Principles

Challenges

- ILP, feasibility
- Projection, simplification
- Dimensionality of scheduling
- Random sampling
- Precise proximity modeling
- Precise profitability modeling

Solutions

- LP, incomplete heuristics
- Sub-polyhedral abstractions (TVPI)
- Structure and cluster statements
- Pairwise and hierarchical scheduling
- Empirical search heuristics
- Restrictions (permutations, bound coeffs)

Sub-polyhedra [Upadrasta et al. POPL 2013]
Pluto+ and LP relaxation [Acharya et al. PPoPP 2015, TOPLAS 2016, PLDI 2015]

More references in the paper

Scalability — Exposing and Exploiting Structure

isl Schedules Trees [Verdoolaege et al. IMPACT 2014] [Grosser et al. TOPLAS 2015]

Domain $\left[\begin{array}{ll} \{\mathsf{S}(i,j) & \mid 0 \le i < N \wedge 0 \le j < K\} \\ \{\mathsf{T}(i,j,k) & \mid 0 \le i < N \\ & \wedge 0 \le j < K \wedge 0 \le k < M\} \end{array}\right.$

Sequence
  Filter $\{\mathsf{S}(i,j)\}$
    Band $\{\mathsf{S}(i,j) \to (i,j)\}$
  Filter $\{\mathsf{T}(i,j,k)\}$
    Band $\{\mathsf{T}(i,j,k) \to (i,j,k)\}$

(a) canonical sgemm

Domain $\left[\begin{array}{ll} \{\mathsf{S}(i,j) & \mid 0 \le i < N \wedge 0 \le j < K\} \\ \{\mathsf{T}(i,j,k) & \mid 0 \le i < N \wedge 0 \le j < K \wedge 0 \le k < M\} \end{array}\right.$

Context $\{N = M = 16 \wedge K > 1000\}$

Band $\left[\begin{array}{ll} \{\mathsf{S}(i,j) & \to (i,j)\} \\ \{\mathsf{T}(i,j,k) & \to (i,j)\} \end{array}\right.$

Sequence
  Filter $\{\mathsf{S}(i,j)\}$
  Filter $\{\mathsf{T}(i,j,k)\}$
    Band $\{\mathsf{T}(i,j,k) \to (k)\}$

(b) fused

Domain $\left[\begin{array}{ll} \{\mathsf{S}(i,j) & \mid 0 \le i < N \wedge 0 \le j < K\} \\ \{\mathsf{T}(i,j,k) & \mid 0 \le i < N \wedge 0 \le j < K \wedge 0 \le k < M\} \end{array}\right.$

Context $\{N = M = K = 512 \wedge 0 \le b_x, b_y < 32 \wedge 0 \le t_x, t_y < 16\}$

Filter $\left[\begin{array}{ll} \{\mathsf{S}(i,j) & \mid i - 32b_x - 31 \le 32 \times 16\lfloor i/32/16 \rfloor \le i - 32b_x \wedge \\ & \quad j - 32b_y - 31 \le 32 \times 16\lfloor j/32/16 \rfloor \le j - 32b_y\} \\ \{\mathsf{T}(i,j,k) & \mid i - 32b_x - 31 \le 32 \times 16\lfloor i/32/16 \rfloor \le i - 32b_x \wedge \\ & \quad j - 32b_y - 31 \le 32 \times 16\lfloor j/32/16 \rfloor \le j - 32b_y\} \end{array}\right.$

Band $\left[\begin{array}{ll} \{\mathsf{S}(i,j) & \to (32\lfloor i/32 \rfloor, 32\lfloor j/32 \rfloor)\} \\ \{\mathsf{T}(i,j,k) & \to (32\lfloor i/32 \rfloor, 32\lfloor j/32 \rfloor)\} \end{array}\right.$

Sequence
  Filter $\{\mathsf{S}(i,j)\}$
  Filter $\{\mathsf{S}(i,j) \mid (t_x - i) = 0 \bmod 16 \wedge (t_y - j) = 0 \bmod 16\}$
    Band $\{\mathsf{S}(i,j) \to (i \bmod 32, j \bmod 32)\}$
  Filter $\{\mathsf{T}(i,j,k)\}$
    Band $\{\mathsf{T}(i,j,k) \to (32\lfloor k/32 \rfloor)\}$
      Band $\{\mathsf{T}(i,j,k) \to (k \bmod 32)\}$
        Filter $\{\mathsf{T}(i,j,k) \mid (t_x - i) = 0 \bmod 16 \wedge (t_y - j) = 0 \bmod 16\}$
          Band $\{\mathsf{T}(i,j,k) \to (i \bmod 32, j \bmod 32)\}$

(e) fused, tiled, sunk and mapped

Domain $\left[\begin{array}{ll} \{\mathsf{S}(i,j) & \mid 0 \le i < N \wedge 0 \le j < K\} \\ \{\mathsf{T}(i,j,k) & \mid 0 \le i < N \\ & \wedge 0 \le j < K \wedge 0 \le k < M\} \end{array}\right.$

Band $\left[\begin{array}{ll} \{\mathsf{S}(i,j) & \to (32\lfloor i/32 \rfloor, 32\lfloor j/32 \rfloor)\} \\ \{\mathsf{T}(i,j,k) & \to (32\lfloor i/32 \rfloor, 32\lfloor j/32 \rfloor)\} \end{array}\right.$

Band $\left[\begin{array}{ll} \{\mathsf{S}(i,j) & \to (i \bmod 32, j \bmod 32)\} \\ \{\mathsf{T}(i,j,k) & \to (i \bmod 32, j \bmod 32)\} \end{array}\right.$

Sequence
  Filter $\{\mathsf{S}(i,j)\}$
  Filter $\{\mathsf{T}(i,j,k)\}$
    Band $\{\mathsf{T}(i,j,k) \to (k)\}$

(c) fused and tiled

Domain $\left[\begin{array}{ll} \{\mathsf{S}(i,j) & \mid 0 \le i < N \wedge 0 \le j < K\} \\ \{\mathsf{T}(i,j,k) & \mid 0 \le i < N \wedge 0 \le j < K \wedge 0 \le k < M\} \end{array}\right.$

Band $\left[\begin{array}{ll} \{\mathsf{S}(i,j) & \to (32\lfloor i/32 \rfloor, 32\lfloor j/32 \rfloor)\} \\ \{\mathsf{T}(i,j,k) & \to (32\lfloor i/32 \rfloor, 32\lfloor j/32 \rfloor)\} \end{array}\right.$

Sequence
  Filter $\{\mathsf{S}(i,j)\}$
    Band $\{\mathsf{S}(i,j) \to (i \bmod 32, j \bmod 32)\}$
  Filter $\{\mathsf{T}(i,j,k)\}$
    Band $\{\mathsf{T}(i,j,k) \to (32\lfloor k/32 \rfloor)\}$
      Band $\{\mathsf{T}(i,j,k) \to (k \bmod 32)\}$
        Band $\{\mathsf{T}(i,j,k0 \to (i \bmod 32, j \bmod 32)\}$

(d) fused, tiled and sunk

Optimization steps for sgemm

Scalability — Mixing Oil and Water

isl Schedules Trees [Verdoolaege et al. IMPACT 2014] [Grosser et al. TOPLAS 2015]

Also:

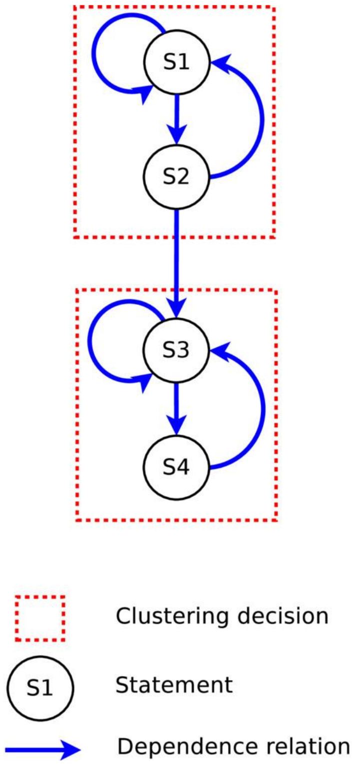
Structured/modular scheduling [Feautrier IJPP 2006]

PolyAST [Shirako et al. SC 2014]
PolyMage [Mullapudi et al ASPLOS 2015]
Tensor Comprehensions [Vasilache et al. TACO 2019]
MLIR/affine https://mlir.llvm.org

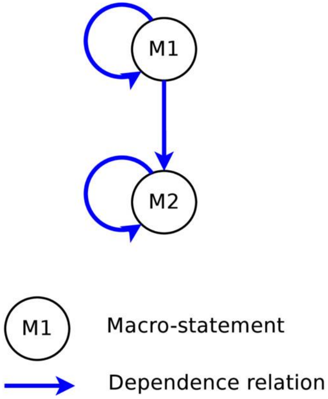
This work: exploit structure by focusing on statement clustering

Clustering SCCs — “Semantics”

Original dependence graph



Clustered dependence graph



Clustering Strongly Connected Components (SCCs) of the reduced dependence graph

Clustering SCCs — “Semantics”

```
for (i = 0; i < N; i++)
  for (j = 0; j < N; j++) {
    temp1 = A[i][j] * B[i][j];
    C[i][j] = temp1;

    temp2 = A[i][j] * C[i][j];
    D[i][j] = temp2;
  }
```

SCC Clustering →

```
for (i = 0; i < N; i++)
  for (j = 0; j < N; j++) {
    M0; // Macro-statement

    M1; // Macro-statement
  }
```

Clustering Strongly Connected Components (SCCs) of the reduced dependence graph

(SCCs considering the innermost dimension only)

Clustering Basic Blocks — “Syntax”

```
for (i = 0; i < N; i++)
  for (j = 0; j < N; j++) {
    temp1 = A[i][j] * B[i][j];
    C[i][j] = temp1;

    temp2 = A[i][j] * C[i][j];
    D[i][j] = temp2;
  }
```

Basic Block Clustering →

```
for (i = 0; i < N; i++)
  for (j = 0; j < N; j++) {
    M0; // Macro-statement

    M1; // Macro-statement
  }
```

Clustering basic blocks irrespectively of dependences, proximity, parallelism

Clustering — Questions

Soundness

- No cycles in the reduced dependence graph of macro statements
- Convexity of the macro statements

Completeness

- Do not miss (interesting) affine schedules
- Interaction with scheduling heuristics

Effectiveness

- Effective scalability benefits
- Effective performance results

Clustering — Questions

Soundness

- No cycles in the reduced dependence graph of macro statements
- Convexity of the macro statements

Completeness

- Do not miss (interesting) affine schedules
- Interaction with scheduling heuristics

Effectiveness

- Effective scalability benefits
- Effective performance results

More detail in the paper

Clustering — A Missing Experiment

Few experiment to evaluate the practical impact of clustering on scheduling effectiveness, separately from scalability

No experiment to compare different forms of clustering

- Offline, syntax: *blocks and nesting structure in the source program,* gcc/Graphite, llvm/Polly, [Mehta et a. PLDI 2015]
- Offline, semantics: *dependence SCCs,* [Meister et al. HPCS 2019]
- Online, incremental, SCCs and proximity: isl, [Zinenko et al. CC 2018]
- Online, with backtracking when clustering hurts feasibility: ?

Clustering — A Missing Experiment

Few experiment to evaluate the practical impact of clustering on scheduling effectiveness, separately from scalability

No experiment to compare different forms of clustering

- Offline, syntax: *blocks and nesting structure in the source program,* gcc/Graphite, llvm/Polly, [Mehta et a. PLDI 2015]
- Offline, semantics: *dependence SCCs,* [Meister et al. HPCS 2019]
- Online, incremental, SCCs and proximity: isl, [Zinenko et al. CC 2018]
- Online, with backtracking when clustering hurts feasibility: ?

Surprise: Negative Result! Offline, syntactic does well

caveat of the study: early experiment, considering only the Pluto optimization space, objectives and heuristics, and limited to Polybench, image processing benchmarks

Clustering — A Missing Experiment

Disclaimer… this is only a preliminary experiment…

Benchmarks

- **27** Polybench 3.2 **converted to three address code** (Polybench-3AC)
- **7** image processing benchmarks from the PENCIL suite
- Allen and Kennedy distribution/vectorization benchmark: “**dist**”
- Unconclusive experiments with SPEC and NAS from Mehta’s benchmarks

Evaluation

- PPCG 0.02 plus clustering and tweaking heuristics externally (Python)
- Dual-core x86

Scheduling Time

Median reduction in #Statements

2.5x for SCC

3x for BB

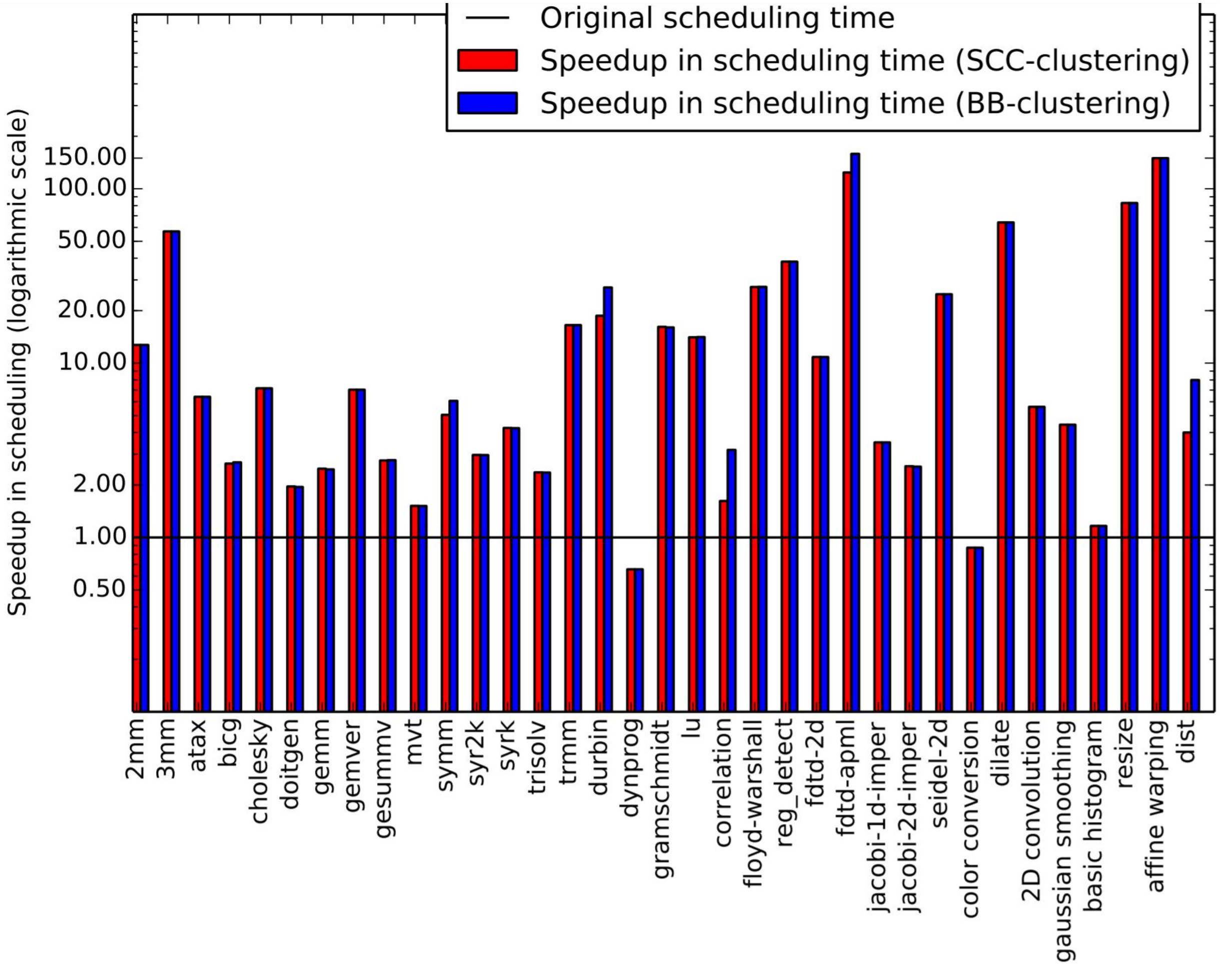
up to 25x in some cases

Median reduction in #Deps

3.67x for SCC

4x for BB

up to 72x in some cases



Execution Time of the Generated Code

4 optimization scenarios considered x 35 benchmarks

- SCC vs. BB clustering
- fusion vs. distribution heuristic

Identical performance, often identical code, in all but 9/150 cases

- BB clustering hurts “dist” benchmark with distribution heuristic
- Chaotic effects on statement ordering yield up to 25% difference

Early and Temporary Conclusion

Without additional effort on evaluating more advanced offline or online clustering heuristics, including more advanced schedulers, BB clustering happens to be just "good enough" (matching Polly folklore and experience)

Early and Temporary Conclusion

Without additional effort on evaluating more advanced offline or online clustering heuristics, including more advanced schedulers, BB clustering happens to be just “good enough” (matching Polly folklore and experience)

- IMPACT is a great venue to publish work in progress
- ... negative results
- … and even “decremental” work!