Feedback-Directed Polyhedral Optimization at Run Time

Andreas Simbürger Armin Größlinger Christian Lengauer

University of Passau
Chair of Programming

29. Juni 2011

Agenda

The Polyhedral Model

- Abstract from the input language to expose parallelism.
- Generic framework for all loop transformations.
- Linear programming to find the optimal transformation polyhedron in terms of a given objective function.
  - Maximal amount of parallelism.
  - Minimal amount of processors.
  - Minimal power consumption.

But

The polyhedral model is limited to (piecewise) affine-linear expressions for the iteration space, the memory accesses and the dependencies.

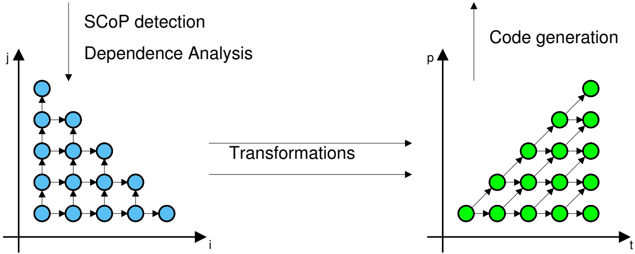
Modelling

Static Control Program (SCoP)

```
for (i=1; i≤n; i++)
 for (j=1; j≤n-i; j++)
  A[i][j] = A[i-1][j] + A[i][j-1];
```

- Structured control flow
  - Regular loop nests
  - Conditions
- Loop bounds and memory accesses restricted to (piecewise) affine-linear expressions.
- Side-effect free function calls.

Transformation



- Iteration space:

  1≤i≤n

  1≤j≤n-i

- Dependences:

  (i,j)→(i+1, j)

  (i,j)→(i, j+1)

- Iteration space:

  1≤t≤n

  1≤p≤t

- Dependences:

  (t,p)→(t+1, p)

  (t,p)→(t+1, p+1)

Problem 1: Non-Linearity

Problem

- Non-linear parameters: `A[n*i + m*j + n*m]`
- Non-linear variables: `A[i*i]`

- Incomplete knowledge about parameters.
- Static solutions yield non-efficient target code.

Problem 1: Non-Linearity

Problem

- Non-linear parameters: `A[n*i + m*j + n*m]`
- Non-linear variables: `A[i*i]`

- Incomplete knowledge about parameters.
- Static solutions yield non-efficient target code.
- Non-linear variables cannot be handled at all.

Problem 2: Generation of Efficient Code

Problem

The optimal static solution is not always the optimal solution for a given target architecture.

- Cache size/-hierarchy unknown.
- Unknown co-processors.
- Synchronization costs unknown.
- Static solution is not adaptable to dynamic run-time constraints.

Solution

Polyhedral Optimization at Run Time

Moving the polyhedral optimization framewok from compile time to run time:

- eliminates static non-linearity.
- enables target code optimization.

Solution

Polyhedral Optimization at Run Time

Moving the polyhedral optimization framewok from compile time to run time:

- eliminates static non-linearity.
- enables target code optimization.

Adaption

Changing run time requirements may result in a change of the objective function:

Maximal amount of parallelism

$\downarrow$

Minimal power consumption

Eliminating Static Non-Linearity

Knowledge about parameters increases the coverage of the model.

Eliminating Static Non-Linearity

Knowledge about parameters increases the coverage of the model.

Parameter $\leftarrow$ Constant

Parameters are a constant at run time $\rightarrow$ non-linearity disappears.

```
for (i=1; i≤n; j++)
 A[n*i + n*m]
```

```
for (i=1; i≤n; j++)
 A[1*i + 1*m]
```

Eliminating Static Non-Linearity

Knowledge about parameters increases the coverage of the model.

Parameter ← Constant

Parameters are a constant at run time → non-linearity disappears.

```
for (i=1; i≤n; j++)
 A[n*i + n*m]
```

```
for (i=1; i≤n; j++)
 A[1*i + 1*m]
```

Parameter ← Small value set

Parameters vary from a small set of values (multi-versioning).

```
for (i=1; i≤n; j++)
 A[n*i + n*m]
```

```
if (n==1)
 for (i=1; i≤n; j++)
  A[i + m]
else if (n==2)
 for (i=1; i≤n; j++)
  A[2*i + 2*m]
```

PolyJIT - Coverage

Coverage

How much execution time is spent inside a SCoP?

- Coverage testing on top of the LLVM Polly project.
- LLVM TestSuite, SPEC2006, PolyBench.
- Evaluate impact of the (static) polyhedral model.
- Extend coverage analysis to run time.
- Run time tests in terms of one single test input.

PolyJIT - Coverage (cont'd)

Test	# of SCoPs		% of Run Time	
	static	approx. max	static	approx. max
403.gcc	562	2752	7.97916	22.72334
smg2000	85	1418	0.51119	75.44391
lencod	153	1405	6.17812	70.42025
cns-typeset	78	1330	0.02395	17.08722
pairlocalalign	153	850	0.01527	18.44517
sqlite3	68	819	2.23905	16.20938
clamscan	185	664	0.10215	1.68986
ldecod	77	636	6.17812	56.60804

Enable Target Code Optimization

- Run time knowledge introduces new constraints.
  - Cache sizes.
  - Communication costs.
  - Actual system load.
  - Power constraints.
- Constraints introduced at run time have no impact on program semantics.

Problem

- No knowledge about the impact of given run time constraints.
- No knowledge about interactions between run time constraints.
- Non-linear constraints.

Feedback-Directed Optimization

Problem

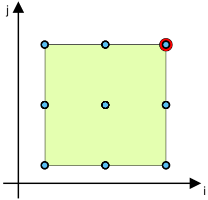
- Unknown interactions between run time constraints.
- Linear approximation for these constraints.
- Approximation yields inefficient target code.

Solution

- Apply run time constraints iteratively.
- Evaluate every single solution with profiling information.
- Accept transformation if benefit passes a given threshold.

Why Feedback-Directed?

Approximated run-time constraint requires us to limit dimension `i` or `j` to values smaller than `2`.

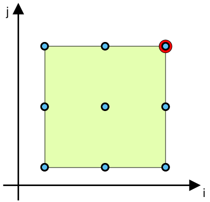


$1 \leq i \leq 3$

$1 \leq j \leq 3$

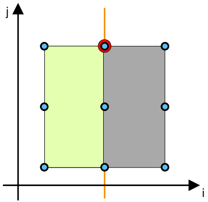
Why Feedback-Directed?

Approximated run-time constraint requires us to limit dimension `i` or `j` to values smaller than `2`.



$1 \leq i \leq 3$

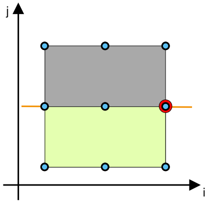
$1 \leq j \leq 3$



$1 \leq i \leq 3$

$1 \leq j \leq 3$

$i \leq 2$



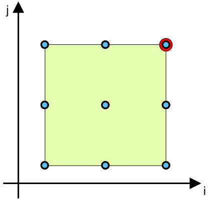
$1 \leq i \leq 3$

$1 \leq j \leq 3$

$j \leq 2$

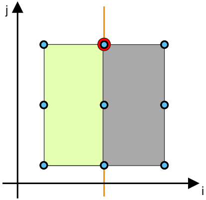
Why Feedback-Directed?

Approximated run-time constraint requires us to limit dimension `i` or `j` to values smaller than `2`.



$1 \leq i \leq 3$

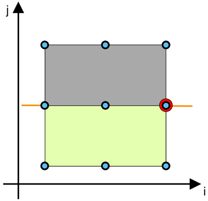
$1 \leq j \leq 3$



$1 \leq i \leq 3$

$1 \leq j \leq 3$

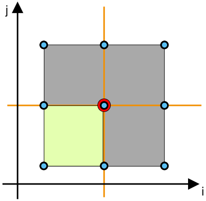
$i \leq 2$



$1 \leq i \leq 3$

$1 \leq j \leq 3$

$j \leq 2$



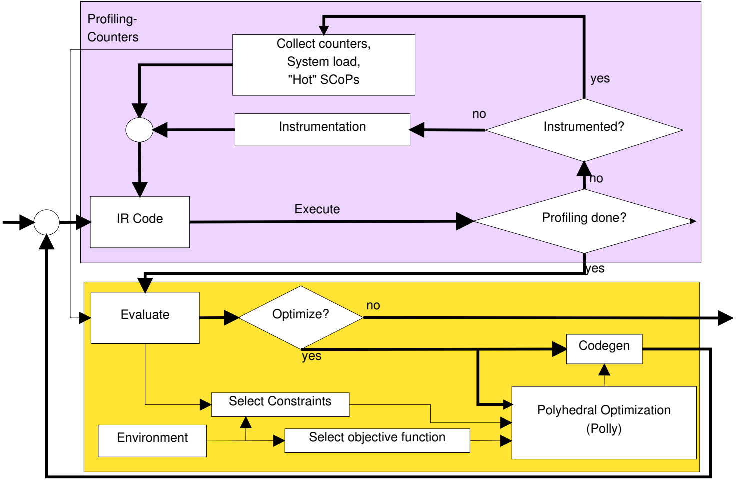
$1 \leq i \leq 3$

$1 \leq j \leq 3$

$i \leq 2$

$j \leq 2$

Feedback Loop



Conclusions

Limitations of static polyhedral methods

- Ineffecient generation of target code.
- Non-linearity restricts the applicability of the model.
- Non-adaptability to changing run time constraints.

Added benefits of dynamic polyhedral methods

- Non-linearity vanishes in many cases.
- Adaptability to changing run time constraints.
- Feedback directed adaptive optimization to evaluate transformations.

Thank You!

Thanks for your attention! Questions?