Feedback-Directed Polyhedral Optimization at Run Time

Andreas Simbürger  Armin Größlinger  Christian Lengauer

University of Passau  Chair of Programming

29. Juni 2011
Agenda

1. The Polyhedral Model
   - Introduction
   - Existing Problems

2. Polyhedral Optimization at Run Time
   - Run Time Optimization
   - The Feedback Loop

3. Conclusions
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)**

```plaintext
for (i=1; i≤n; i++)
  for (j=1; j≤n-i; j++)
```

- 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:**
  \[ 1 \leq i \leq n \]
  \[ 1 \leq j \leq n - i \]

- **Dependences:**
  \[(i, j) \rightarrow (i+1, j)\]
  \[(i, j) \rightarrow (i, j+1)\]

- **Iteration space:**
  \[ 1 \leq t \leq n \]
  \[ 1 \leq p \leq t \]

- **Dependences:**
  \[(t, p) \rightarrow (t+1, p)\]
  \[(t, p) \rightarrow (t+1, p+1)\]
Problem 1: Non-Linearity

- 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

- 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

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 framework from compile time to run time:

- eliminates static non-linearity.
- enables target code optimization.
Polyhedral Optimization at Run Time

Moving the polyhedral optimization framework 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:

\[
\text{Maximal amount of parallelism} \quad \downarrow \quad \text{Minimal power consumption}
\]
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.

\[
\text{for } (i=1; i \leq n; j++) \\
\phantom{.} A[n*i + n*m] \\
\text{for } (i=1; i \leq n; j++) \\
\phantom{.} A[1*i + 1*m]
\]
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.

\[
\text{for } (i=1; i \leq n; j++) \\
A[n*i + n*m]
\]

Parameter $\leftarrow$ Small value set

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

\[
\text{for } (i=1; i \leq n; j++) \\
A[n*i + n*m]
\]

\[
\text{if } (n==1) \\
\text{for } (i=1; i \leq n; j++) \\
A[i + m]
\]

\[
\text{else if } (n==2) \\
\text{for } (i=1; i \leq n; j++) \\
A[2*i + 2*m]
\]
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)

<table>
<thead>
<tr>
<th>Test</th>
<th># of SCoPs</th>
<th>% of Run Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>static</td>
<td>approx. max</td>
</tr>
<tr>
<td>403.gcc</td>
<td>562</td>
<td>2752</td>
</tr>
<tr>
<td>smg2000</td>
<td>85</td>
<td>1418</td>
</tr>
<tr>
<td>lencod</td>
<td>153</td>
<td>1405</td>
</tr>
<tr>
<td>cns-typeset</td>
<td>78</td>
<td>1330</td>
</tr>
<tr>
<td>pairlocalalign</td>
<td>153</td>
<td>850</td>
</tr>
<tr>
<td>sqlite3</td>
<td>68</td>
<td>819</td>
</tr>
<tr>
<td>clamscan</td>
<td>185</td>
<td>664</td>
</tr>
<tr>
<td>ldecod</td>
<td>77</td>
<td>636</td>
</tr>
</tbody>
</table>
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.
Approximated run-time constraint requires us to limit dimension $i$ or $j$ to values smaller than 2.
Why Feedback-Directed?

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

$1 \leq i \leq 3$
$1 \leq j \leq 3$

$1 \leq i \leq 3$
$1 \leq j \leq 3$
$i \leq 2$

$1 \leq i \leq 3$
$1 \leq j \leq 3$
$j \leq 2$
Approximated run-time constraint requires us to limit dimension $i$ or $j$ to values smaller than 2.

\[
1 \leq i \leq 3 \\
1 \leq j \leq 3 \\
i \leq 2 \\
1 \leq i \leq 3 \\
1 \leq j \leq 3 \\
i \leq 2 \\
j \leq 2
\]
Conclusions

Limitations of static polyhedral methods
- Inefficient 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?