Predicated Worst-Case Execution-Time (WCET) Analysis

Amine Marref, Guillem Bernat

{marref, bernat}@cs.york.ac.uk

University of York
Roadmap

- Background
- Motivation
- Predicated WCET Analysis
- Results
- Conclusions
Background

- Generalities
  - Schedulability analysis needs WCET
  - Also optimization
  - *WCET of a task is the maximum execution time that a task can ever exhibit*
  - Goals: safety + tightness

- Types of analysis
  - Static analysis (SA)
    - flow analysis, hardware modeling, calculation
  - Dynamic analysis (end-to-end)
    - Random, GAs, best-effort, engineering wisdom
  - Measurement-based (MB)
    - flow analysis, measurements, calculation
Background: IPET

- General procedure
  - Partition into segments
  - Find execution times of segments
  - Calculate: path-based, tree-based, IPET (Implicit Path-Enumeration Technique)

Calculation Methods

- Path-Based
- IPET
- Tree-Based

- ILP
- Non-Linear
- CLP
- Others?
Background: ILP Issue

- IPET widely used
  - Powerful constraint modeling
  - Efficient ILP solvers
  - \[ f = x_1 \times c_1 + x_2 \times c_2 + \ldots + x_n \times c_n \ (n \text{ segments}) + \text{a set of constraints} \]

- Issue with complex hardware
  - Variable execution times
  - Constant execution times: pessimism

- IPET based on ILP
  - Augment model with hardware effects
  - Augment objective function with gains/penalties
  - Becomes messy for more than 1 hardware speed-up feature
Motivation: Example 1

Blocks
\[ x_A, x_B, x_C, x_D \]
\[ c_A, c_B, c_C, c_D \]
\[ c_D \in \{ c_{D/B}, c_{D/C} \} \]
\[ c_{D/B} = \hat{c}_D - g_1 \]
\[ c_{D/C} = \hat{c}_D - g_2 \]

Gains
\[ x'_1 = x_{BD} \]
\[ x'_2 = x_{CD} \]
\[ c'_1 = g_1 \]
\[ c'_2 = g_2 \]

\[ f = \max(\sum_{i=A}^{D} x_i \times c_i + \sum_{j=1}^{2} x'_j \times c'_j) \]
Motivation: Example 2

Blocks
\[ x_A, x_B, x_C, x_D, x_E, x_F, x_G \]
\[ c_A, c_B, c_C, c_D, c_E, c_F, c_G \]
\[ c_E \in \{c_{E/B}, c_{E/C}\} \]
\[ c_{E/B} = \hat{c} - g_1 \]
\[ c_{E/C} = \hat{c} - g_2 \]

Gains
\[ x'_1 = x_{BDE} = ? \]
\[ x'_2 = x_{CDE} = ? \]
\[ c'_1 = g_1 \]
\[ c'_2 = g_2 \]

\[ f = \max (\sum_{i=A}^{G} x_i \times c_i + \sum_{j=1}^{2} x'_j \times c'_j) \]
Motivation: Summary

- The problem of modeling the variability in execution times using ILP reduces to the problem of mapping the $x'$ variables to some $x$ variables in the model.
  - The mapping is straightforward in Example 1: The effect of B on D occurs whenever B executes.
  - The mapping in Example 2 is not obvious.
  - Ermedahl suggested bounding the effect from top and bottom.
    - Tedious if affected block far from affecting block.
    - Because ILP is not path-sensitive, negative effects can be included in the final solution without the block sequences causing them.
    - This causes pessimism.

- Need to include some path-sensitivity.
  - A particular execution time of some basic block only occurs given some block has executed before.
  - e.g. $x_B > 0 \Rightarrow c_D = 10$ (Example 1)
A Solution Using ILP

- ILP supports conjunction and negation only
- Disjunction is supported through model duplication
- We can implement path-sensitivity through mutual exclusive constraints
  - Implications become disjunctions
  - \((x_B > 0 \Rightarrow c_D = 10) \iff (x_B \leq 0 \lor c_D = 10)\)
  - Solve all instances of the disjunctive ILP
  - a model with \(n\) disjunctions solved in at least \(2^n\) runs

- exponential behaviour
CLP, PWA

- Use Constraint-Logic Programming
  - Conditional execution times expressed through implication

- This yields Predicated WCET Analysis
  - Performing WCET analysis by considering all different execution times of a program segment and expressing them as the outcomes of executing some other segments in the past

- Derive constraints
  - Find segments that affect execution time of current segment
  - Link these effects to execution times

- Solve model using CLP
## Results: Tightness

<table>
<thead>
<tr>
<th>program</th>
<th>blocks</th>
<th>implications</th>
<th>wcet</th>
<th>gain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>HMU</td>
<td>PWA</td>
</tr>
<tr>
<td>select</td>
<td>40</td>
<td>27</td>
<td>558627</td>
<td>432803</td>
</tr>
<tr>
<td>cover</td>
<td>599</td>
<td>2593</td>
<td>44801</td>
<td>38081</td>
</tr>
<tr>
<td>fdct</td>
<td>12</td>
<td>6</td>
<td>77759</td>
<td>66975</td>
</tr>
<tr>
<td>fir</td>
<td>17</td>
<td>4</td>
<td>87822</td>
<td>81742</td>
</tr>
<tr>
<td>lms</td>
<td>134</td>
<td>86</td>
<td>747776</td>
<td>724752</td>
</tr>
<tr>
<td>cnt</td>
<td>36</td>
<td>2</td>
<td>94672</td>
<td>92912</td>
</tr>
<tr>
<td>bsort</td>
<td>20</td>
<td>4</td>
<td>58179</td>
<td>57539</td>
</tr>
<tr>
<td>ns</td>
<td>22</td>
<td>5</td>
<td>892708</td>
<td>888148</td>
</tr>
</tbody>
</table>
Results: Solution Time - Uninformed Constraint Search

![Bar chart showing solution times for different program numbers. The chart has a vertical axis labeled 'solution time (seconds)' ranging from 0 to 50, and a horizontal axis labeled 'program number.' The chart includes data points for program number 1, 2, 3, 4, 5, 6, 7, and 8. Program number 2 has the longest solution time, indicating a significant difference in performance compared to the other programs.]
Results: Solution Time & Scalability
Summary/Conclusions

- Presented predicated WCET Analysis
- Logic programming can be used to model execution dependencies
- Hardware analysis integration rendered possible
- Enforces path-sensitivity in execution times
- ILP not powerful enough to handle execution time variations
- If model has a manageable number of disjunctions, use ILP, otherwise CLP
- Also use CLP to handle unusual flow facts e.g. A xor B or not C
Current Work

- Deriving constraints from traces
- Performing WCET coverage
- Implementing search procedures to solve constraints more efficiently
- Investigating the scalability of the approach