Single-Path Code Generation and Input-Data Dependence Analysis

Daniel Prokesch
daniel@vmars.tuwien.ac.at

July 10\textsuperscript{th}, 2014
Project Workshop Madrid
Motivation

Uncertainty in program execution time
Motivation

Uncertainty in program execution time

- **Hardware**

  Duration of operations dependent on internal state
Motivation

Uncertainty in program execution time

**Hardware**
Duration of operations dependent on internal state

**Software**
Different inputs lead to paths of varying execution time
The Single-Path Approach

- Singleton execution path
  - Reduce complexity of path analysis
The Single-Path Approach

- Singleton execution path
  - Reduce complexity of path analysis

- Execution time stable w.r.t. varying input data
  - Known ET for a particular input
    - Reasonable predictions for any input
The Single-Path Approach

Eliminate input-data dependent control flow.
The Single-Path Approach

Eliminate input-data dependent control flow.

Predicated execution

(guard) instruction;

If-conversion

```cpp
if (cond) {
    x = a + 1;
} else {
    x = b - 2;
}
```

( cond) x = a + 1;
( !cond) x = b - 2;
Single-Path Transformation Rules

- Rules to transform from high-level language to single-path sequence of predicated statements [Puschner et al., 2012]
Single-Path Transformation Rules

- Rules to transform from high-level language to single-path sequence of predicated statements [Puschner et al., 2012]

- Example: Treatment of loops

```plaintext
while (cond)
  // max N times
  {
    stmts;
  }

finished := false;
for i = 1 .. N {
  if (!cond)
    finished := true;
  if (!finished) {
    stmts;
  }
}
```
Single-Path Transformation Rules

- Rules to transform from high-level language to single-path sequence of predicated statements [Puschner et al., 2012]
- Example: Treatment of loops

```plaintext
while (cond) // max N times
{
    stmts;
}
```

```plaintext
finished := false;
for i = 1 .. N {
    if (!cond)
        finished := true;
    if (!finished) {
        stmts;
    }
}
```

- How to implement this in a compiler??
Single-Path Code Generation

Contributions

- Formulation of the single-path transformation on the CFG level
Single-Path Code Generation

Contributions

- Formulation of the single-path transformation on the CFG level
- Prototype implementation in a code generator (LLVM backend for Patmos)
Control-flow Graph

- Nodes with at most two successors
- Branch condition

- Reducible control flow graph
The Single-Path Graph Transformation

- Admissible paths in CFG: local loop bounds
- Predicated Nodes
- Semantic actions on predicates at nodes and edges
The Single-Path Graph Transformation

- Admissible paths in CFG: local loop bounds
- Predicated Nodes
- Semantic actions on predicates at nodes and edges

Control flow graph + local loop bounds \( \rightarrow \) \( SPGT \) \( \rightarrow \) Single-path graph + predicated execution
Acyclic regions


Partitioning by Control Dependence
Acyclic regions

[Park and Schlansker, 1991] describe transformation in acyclic case

- Partitioning by Control Dependence
- Nodes with same set of CD: same predicate
Acyclic regions

[Park and Schlansker, 1991] describe transformation in acyclic case

- Partitioning by **Control Dependence**
- Nodes with same set of CD: same predicate
Acyclic regions

[Park and Schlansker, 1991] describe transformation in acyclic case

Partitioning by Control Dependence

Nodes with same set of CD: same predicate

Predicate assignments at source of CD edges

Initialization with \textit{false}

Topological sort order
Loops

- Forward control flow graph of loop nodes
- Nested inner loops compacted into single node
- Proceed as in acyclic case
Loops

- Forward control flow graph of loop nodes
- Nested inner loops compacted into single node
- Proceed as in acyclic case
- Input-data independent branch condition
- Former exit edge causes clearing of header predicate
Loops

- Forward control flow graph of loop nodes
- Nested inner loops compacted into single node
- Proceed as in acyclic case
- Input-data independent branch condition
- Former exit edge causes clearing of header predicate

Recursive composition of the SP-Graph starting from outermost (pseudo-) loop
Determine input-data dependent branch conditions

\[\text{Path from } x \rightarrow y: x \text{ may depend on } y, \text{ otherwise } x \text{ is independent from } y\]

\(^1\)work mostly by Benedikt Huber
Input-Data Dependency Analysis

- Determine input-data dependent branch conditions

- Input data dependency
  - Definition based on Thinned Gated Static Assignment (TGSA) form

\(^1\)work mostly by Benedikt Huber
Input-Data Dependency Analysis

- Determine input-data dependent branch conditions
- Input data dependency
  - Definition based on Thinned Gated Static Assignment (TGSA) form
  - $\gamma$-instructions, $\mu$-instructions, $\eta$-instructions capture various types of dependencies

---

1 work mostly by Benedikt Huber
Input-Data Dependency Analysis

- Determine input-data dependent branch conditions

- Input data dependency
  - Definition based on Thinned Gated Static Assignment (TGSA) form
  - $\gamma$-instructions, $\mu$-instructions, $\eta$-instructions capture various types of dependencies
  - From TGSA, input-data dependency graph is constructed
  - Path from $x \rightarrow y$: $x$ may depend on $y$, otherwise $x$ is independent from $y$

\(^1\) work mostly by Benedikt Huber
Prototype implementation in Patmos backend (LLVM)
Implementation

- Prototype implementation in Patmos backend (LLVM)

- Patmos features . . .
  - A fully predicated instruction set
  - 7+1 predicate registers that can be stored/restored at once
  - Explicitly managed local memories
Example: Insertion Sort

- Fetch array to SPM, sort, copy back to memory
- Exhaustive simulation
Example: Insertion Sort

- Fetch array to SPM, sort, copy back to memory
- Exhaustive simulation

Results:

| Static Analysis | 771 |
Example: Insertion Sort

- Fetch array to SPM, sort, copy back to memory
- Exhaustive simulation
- Results:

<table>
<thead>
<tr>
<th>Static Analysis</th>
<th>771</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Analysis w/ LC</td>
<td>546</td>
</tr>
</tbody>
</table>
Example: Insertion Sort

- Fetch array to SPM, sort, copy back to memory
- Exhaustive simulation
- Results:

<table>
<thead>
<tr>
<th>Static Analysis</th>
<th>771</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Analysis w/ LC</td>
<td>546</td>
</tr>
<tr>
<td>Measurements</td>
<td>[473, 501]</td>
</tr>
</tbody>
</table>
Example: Insertion Sort

- Fetch array to SPM, sort, copy back to memory
- Exhaustive simulation
- Results:

<table>
<thead>
<tr>
<th>Static Analysis</th>
<th>771</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Analysis w/ LC</td>
<td>546</td>
</tr>
<tr>
<td>Measurements</td>
<td>[473, 501]</td>
</tr>
<tr>
<td>Single-Path (single-issue)</td>
<td>580</td>
</tr>
</tbody>
</table>
Example: Insertion Sort

- Fetch array to SPM, sort, copy back to memory
- Exhaustive simulation
- Results:

<table>
<thead>
<tr>
<th>Static Analysis</th>
<th>771</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Analysis w/ LC</td>
<td>546</td>
</tr>
<tr>
<td>Measurements</td>
<td>[473, 501]</td>
</tr>
<tr>
<td>Single-Path (single-issue)</td>
<td>580</td>
</tr>
<tr>
<td>Single-Path</td>
<td>485</td>
</tr>
</tbody>
</table>
Summary

- Single-path conversion as transformation on CFGs
- Prototype implementation in Patmos backend

http://github.com/t-crest/
References I
