Learning Compilers for Configurable Processors

Hugh Leather
University of Edinburgh
Overview

- Members of the Milepost group
- Introduction to machine learning in compilers
- Learning compiler flags for an embedded processor
- Learning compilers over multiple architectures
- List of open source software in the project
- Pros/cons of open source software
Milepost Members

Funded by:
Information Society Technologies (IST) of the European Union under 6th FWP
Machine Learning in Compilers

- Writing compiler heuristics is hard
  - Need deep knowledge
  - Change elsewhere needs change everywhere
  - Architectures/libraries/OS change
- Humans can't keep up
  - Heuristics remain unoptimised
- Can machine learning help?
We start by choosing some information that might be useful to the heuristic (features):

- number of instructions
- mean dependency depth
- branch count
- loop nest level
- trip count
Collect lot of example benchmarks and compute their features
Execute the programs with different compilation strategies and determine best for each
Machine Learning in Compilers

- Pass these to a machine learning tool
- It learns a model

Diagram:
- Programs
- Features
- Best Heuristic Values
- Supervised Machine Learner
- Model
A model is really just a way of fitting a curve to data
A model is really just a way of fitting a curve to data
Machine Learning in Compilers

- Gives heuristic for unseen points
Machine Learning in Compilers

- Insert model into compiler – heuristic replaced
- Automatic
  - No human expert required
  - Can be redone after changes
An Experiment

- Learn optimisation flags for MiBench with GCC
- Target the ARC 725 customisable processor
- Trained on
  - 500 random flag settings
  - 5 executions each
  - (2 months runtime on 2 machines)
An Experiment
Filling the Database

Benchmarks → Milepost GCC

Iterative compilation → Arc board

Execution time, Code size → Database at Inria

Program features and compiler flags
An Experiment Building the Model

Database at Inria

- Execution time
- Compiler flags
- Program features

Model Learning

Matlab code

Milepost GCC Model

Features

Predicted compiler flags
An Experiment Using Improved Compiler
### An Experiment

#### Results

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>automotive_bitcount</td>
<td>1.38</td>
</tr>
<tr>
<td>automotive_qsort1</td>
<td>1.00</td>
</tr>
<tr>
<td>automotive_susan_c</td>
<td>1.10</td>
</tr>
<tr>
<td>automotive_susan_e</td>
<td>1.00</td>
</tr>
<tr>
<td>automotive_susan_s</td>
<td>0.95</td>
</tr>
<tr>
<td>consumer_jpeg_c</td>
<td>1.14</td>
</tr>
<tr>
<td>consumer_jpeg_d</td>
<td>1.00</td>
</tr>
<tr>
<td>network_dijkstra</td>
<td>0.88</td>
</tr>
<tr>
<td>network_patricia</td>
<td>1.00</td>
</tr>
<tr>
<td>office_stringsearch1</td>
<td>1.38</td>
</tr>
<tr>
<td>security_blowfish_d</td>
<td>1.16</td>
</tr>
<tr>
<td>security_blowfish_e</td>
<td>1.16</td>
</tr>
<tr>
<td>security_rijndael_d</td>
<td>1.01</td>
</tr>
<tr>
<td>security_rijndael_e</td>
<td>1.03</td>
</tr>
<tr>
<td>security_sha</td>
<td>1.12</td>
</tr>
<tr>
<td>telecom_adpcm_c</td>
<td>1.25</td>
</tr>
<tr>
<td>telecom_CRC32</td>
<td>1.25</td>
</tr>
<tr>
<td>telecom_gsm</td>
<td>1.00</td>
</tr>
</tbody>
</table>

**Average:** 1.10
Co-Design Exploration

- Designing an embedded system for some application
- Create many architectures in a 'family'
- Choose the most efficient
Co-Design Exploration
Co-Design Exploration

Tuned Compiler

Features -> Architecture

Performance

- Green
- Red
- Brown
- Green
Co-Design Exploration

Tuned Compiler

Features

Architecture

Performance

✓

✗
Co-Design Exploration

- Add some architectural features
  - Cache sizes
  - Number of registers
  - Number & type of FUs
- Incorporate into machine learning for heuristics
- Automatically learned a tuned compiler for a new architecture!
Co-Design Exploration

Tuned Compiler Across Architectures

Features → Architecture → Performance

- Features
- Architecture
- Performance

- Green tick
- Green cross
Open Source Work

- Must have extensible compiler
  - Alter heuristics
  - Gather features
  - Push button compiler tuning
  - Work on configurable processors
Open Source Work

- MilePost GCC
  - Extensibility through shared libraries
  - Extensions to control heuristics
  - Extensions to compute features
  - Provides an 'Iterative Compilation Interface'

(milepost branch in GCC SVN)
Open Source Work

- Continuous Collective Compilation Framework (http://sourceforge.net/projects/ccccpf)
  - Driver framework – automatically tunes compiler
  - Runs iterative compilations
    - Benchmarks and settings included
  - Learns heuristics
    - Heuristics predefined
  - Uses MilePost GCC
Open Source Work

Training

Program_1

...

Program_N

Predicting

New program

MILEPOST GCC
(with ICI and ML routines)

IC Plugins

- Recording pass sequences
- Extracting static program features

MILEPOST GCC

- Extracting static program features
- Selecting "good" passes

CCC
Continuous Collective Compilation Framework

Drivers for iterative compilation and model training

Global Optimization Database

ML driver to predict "good" passes
(will be implemented as an IC Plugin later)
Open Source Work

- **ARC tool chain** ([http://www.arc.com/software/development/gnutools.html](http://www.arc.com/software/development/gnutools.html))
  - Provides back-end for ARC processors
  - GCC, GDB, BinUtils, uClibc
  - MilePost GCC can be layered on top
OSS Challenges

- Political
  - Extensibility in GCC seen as a threat
  - Major changes need substantial backing

- Legal
  - GPL3 makes lawyers quiver

- How to make money from OSS

- Moving target
  - Just when your code works someone changes the compiler
OSS Benefits

- Huge user base
  - Peer provided support better than most companies'
  - Much wider feedback / testing
  - Allows greater visibility
- Companies can make use of others' work
  - No NDAs needed once OSS approval in house
- Customers expect GCC availability
  - Not so good for compiler vendors
Conclusion

- A system for automatically tuning compilers
- Improves over human created heuristics
- May learn across architectures
- Heavy use of open source software
- Huge and exciting new field opening up
Questions?