A Hybrid Framework to Accelerate Adaptive Compilation

Gabriel K. Bertazi  Anderson Faustino da Silva  Edson Borin

April 19, 2015
Compiler optimizations are not always beneficial

- A compiler optimization can degrade performance for a specific program.
Compiler optimizations are not always beneficial

- A compiler optimization can degrade performance for a specific program.
- Hard to predict the interaction among several compiler optimizations.
A compiler optimization can degrade performance for a **specific program**.

Hard to predict the interaction among several compiler optimizations.

Find and disable compiler optimizations that are harmful for a **specific program**.
Compiler optimizations are not always beneficial

- A compiler optimization can degrade performance for a specific program.
- Hard to predict the interaction among several compiler optimizations.
- Find and disable compiler optimizations that are harmful for a specific program.
- **Adaptive Compilation.**
Classical Adaptive Compilation Algorithm

1. Search optimization set
2. Compile
3. Execute
4. Is search over?
   - no: go back to search optimization set
   - yes: Pick the best set
Classical Adaptive Compilation Algorithm

1. Search optimization set
2. Compile
3. Execute
4. Is search over?
   - no
   - yes
5. Pick the best set
Virtual Execution

- ACME: Virtual Execution Framework.
Virtual Execution

- ACME: Virtual Execution Framework.
- Estimates performance of compiled code via static analysis.
Virtual Execution

- ACME: Virtual Execution Framework.
- Estimates performance of compiled code via static analysis.
- Uses basic block frequency to predict the execution time.
Virtual Execution

- **ACME**: Virtual Execution Framework.
- Estimates performance of compiled code via static analysis.
- Uses basic block frequency to predict the execution time.
- Greatly reduces adaptive compilation time.
Virtual Execution

- ACME: Virtual Execution Framework.
- Estimates performance of compiled code via static analysis.
- Uses basic block frequency to predict the execution time.
- Greatly reduces adaptive compilation time.
- Selects the optimization set with the smallest number of instructions executed.
Virtual Execution Mechanism

1. Search optimization set
2. Compile
3. Virtual Execution Analysis
   - Is search over?
     - no
     - yes: Pick the best set
Virtual Execution Mechanism

- Search optimization set
- Compile
- Virtual Execution Analysis
- Is search over?
  - no
  - yes
    - Pick the best set

- How accurate is the Virtual Execution method?
Virtual Execution Mechanism

- How accurate is the Virtual Execution method?
- How close can we get to the best configuration found on the classical approach?

1. Search optimization set
2. Compile
3. Virtual Execution Analysis
4. Is search over?
   - no
   - yes
5. Pick the best set
Experimental Setup

- Correlate: Number of executed instructions $\Rightarrow$ Execution time.
- Adaptive compilation of SPEC CPU2006.
- Optimization search: Adapted Case-Based Reasoning algorithm.
- LLVM 3.4.
- Core i7-2260 with 4GB RAM.
Estimating Execution Time – 400.Perlbench

![Graph showing execution time and instructions executed over optimizations sets.](image-url)
Estimating Execution Time – 456.hmmr

![Graph showing execution time and instructions executed over optimization sets.](image-url)
The Hybrid Algorithm

VIRTUAL EXECUTION

Search optimization set

Compile

Virtual Execution
Analysis

Is search over?

yes

no

Select k best sets

REAL EXECUTION

Choose one among the k sets

Execute

Analyze performance

Is search over?

yes

no

Pick the best set
Speedups

- Never improve baseline
- No difference between baseline and optimization sets
- K=1
- K=5
- K >= 11

Graph showing speedups for various benchmarks with different K values.
Some correlation on the number of instructions executed and the execution time.
Some correlation on the number of instructions executed and the execution time.

The virtual execution method provides us with good speedup for some benchmarks but fails to obtain the optimal solution for others.
Some correlation on the number of instructions executed and the execution time.

The virtual execution method provides us with good speedup for some benchmarks but fails to obtain the optimal solution for others.

The Hybrid model may provide us with better configurations for these benchmarks.
Future Work

- Find new parameters or variables to improve the prediction of execution time.
- Build a framework to provide hybrid adaptive compilation on the LLVM compiler.
Thanks!

Gabriel Krisman Bertazi - gabriel.bertazi@students.ic.unicamp.br
Anderson Faustino da Silva - anderson@dim.uem.br
Edson Borin - edson@ic.unicamp.br