A Performance Characterization of a Hardware Mechanism for Dynamic Optimization

Brian Fahs  Satarupa Bose  Matthew Crum  Brian Slechta  Francesco Spadini  Tony Tung  Sanjay J. Patel  Steven S. Lumetta

Advanced Computing Systems Group
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

http://www.crhc.uiuc.edu/ACS
Overview

• Basic Components of the rePLay Framework

• Performance Analysis

• Current Work

• Conclusion
The rePlay Framework
The Basics of RePLay
Frames, Assertions and Recovery
• **Purpose**
  - Combine basic blocks to form a frame

• **Requirements**
  - long frames
  - high completion/low assertion rate
  - high I-stream coverage

• **Components**
  - frame construction buffer
  - branch bias table

• **Statistics**
  - average frame length: 100 instructions
  - very low assertion rate (<3%)
  - covers 70% of the dynamic instruction stream
  - results reported in MICRO 2000
The RePLay Optimization Engine

- Programmable, with local memory
- Hardware blocks to assist with dead code, etc…
- Possible implementation: optimizer running as a thread on an MT processor
Example of Optimization
[from gcc]

```
beq r1, A
bne r1, B

ldq r1,24(r1)

lda r29, 0(r1)
sll r29,48,r16
srl r16,48,r16
cmpeq r16,29,r16
beq r16, C

lda r16,0(r29)
sll r0,48,r0
srl r0,48,r0
cmpeq r0,28,r0
beq r0, D

ldq r1,24(r1)
asst_i r1, NE

lda r1, A
bne r1, B

beq r1, A
bne r1, B
```

```
beq r0, D
beq r16, E
```

```
beq r16, C
```
Dynamic Optimization

• Classic optimizations
  • dead code removal
  • constant propagation
  • reassociation
  • common sub-expression elimination
  • subroutine inlining
  • miscellaneous

• RePlay optimization
  • fetch scheduling
Fetch Scheduling of Branches

[T. Tung, MS Thesis, UIUC 2001]
Performance Analysis
Experimental Framework

- **IC**: 64K Icache

- **TC**: 16K Icache w/ 48K trace cache

- **RP**: 16K Icache w/ 48K frame cache w/ out optimizations

- **RPO**: 16K Icache w/ 48K frame cache w/ optimizations

8-wide fetch/ execute/ retire
Performance on Statically Optimized Binaries

![Bar chart showing performance metrics for various binaries. The chart includes categories such as FrameCycles, NormalCycles, WrongPathCycles, AssertCycles, StalledCycles, and FetchMissCycles. Specific percentages are highlighted for each category across different binaries, indicating performance variations.]
Performance on Unoptimized Binaries

Millions of Cycles

FrameCycles  NormalCycles  WrongPathCycles  AssertCycles  StalledCycles  FetchMissCycles

bzp  cra  eon  gap  gcc  gzp  mcf  par  per  twf  vor  vpr

11.4%  11.1%  19.8%  26.9%  7.9%  10.2%  15.8%  11.2%  15.9%  11.0%  13.7%  15.9%
Effect of Individual Optimizations

The graph shows the relative performance of different optimization techniques for various benchmarks. The x-axis represents different benchmarks (bzp, cra, gap, gcc, gzp, par, per, twf, vor, vpr), and the y-axis represents the relative performance. The graph includes bars for different optimization techniques:

- dcr
- cp
- ra
- inlining
- misc
- cse
- fs

The graph highlights the impact of individual optimizations on performance across different benchmarks.
Effects of Optimization Latency
Effects of Frame Length

![Graph showing the effects of frame length on relative performance. The y-axis represents relative performance, ranging from 0% to 120%. The x-axis represents percent of maximum frame length, ranging from 0% to 100%. The graph includes various symbols and colors for different benchmarks, such as bzp, cra, eon, gap, gcc, gzp, mcf, par, per, twf, vor, vpr, and gap-RP. The performance increases as the frame length increases.]
Current Work
Generalized Assertion Model

- A
- B
- C
- D
- E

simple opts
control flow assertions

A'

speculative opts
general assertions
memory assertions

A''
Conclusion
Important Points

• Effective dynamic optimization
  • on large regions, spanning multiple basic blocks
  • low overhead recovery

• With simple optimizations, we are able to attain a reduction in execution cycles
  • 13% for the optimized SPEC2000 integer benchmarks
  • 21% for the unoptimized benchmarks
Advanced Computing Systems

University of Illinois at Urbana-Champaign

http://www.crhc.uiuc.edu/ACS