.

Modeling Superscalar Processors

via Statistical Simulation

June 27, 1998

Richard Carl

James E. Smith

Dept. of Elect. and Comp. Engr.
1415 Johnson Drive
Univ. of Wisconsin
Madison, WI 53706

jes@ece.wisc.edu
http://www.engr.wisc.edu/ece/faculty/smith_james.html

Motivation

- For evaluating computer performance simulation is method of choice
- For large systems and large problems, clock-cycle-level simulation may be too time consuming and/or difficult to model
- Analytical methods often over-simplify or are too complicated for closed-form solution
- Hybrid approach -- statistical simulation

  Simulate a simple probabilistic model using program statistics
- *Caveat:* This is work in progress

  Currently modeling/understanding superscalar processors

  Later work will study complex systems and applications

Approach

- Step 1: Simulate program(s) and collect statistics

  Program related stats:
  - instruction mixes,
  - dependence relationships

  gathered by simple functional simulation

  Implementation related stats:
  - cache miss rates,
  - branch misprediction rates

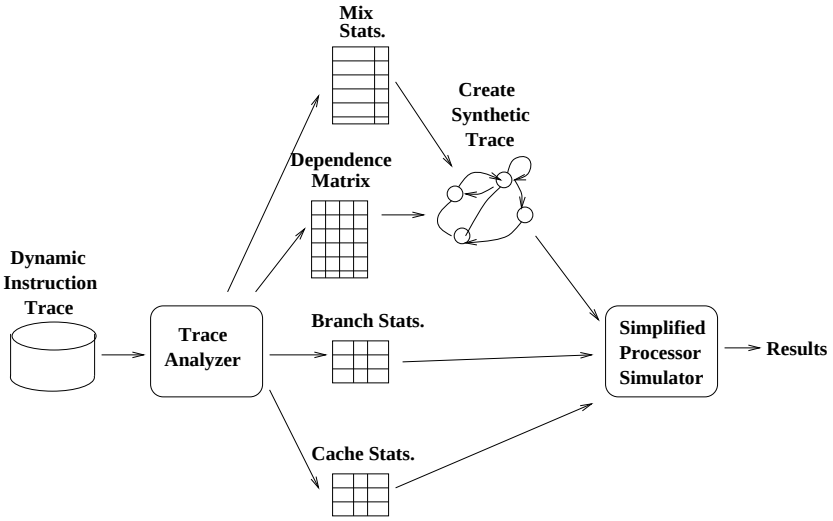
  gathered by simple trace-driven cache and branch predictor simulations

  Ideally, minimize implementation-related stats

- Step 2: Simulate implementation with synthetic traces and statistical caches/predictors

- Run until results converge

The Big Picture



Advantages

- Model construction simplified

  Caches and predictors are probabilistic

  Only major instruction types (fewer than 10) are implemented

  A complex superscalar processor only takes a few hundred lines of code

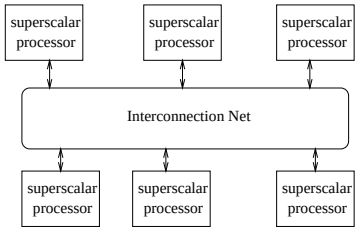
- Model "simulates" much faster

  by many orders of magnitude

  converges after a few thousand cycles

- Combining 1 & 2:

  Large system models can be constructed and simulated with reasonable time and resources

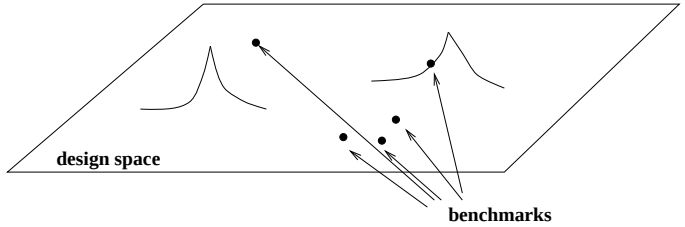


Advantages, contd.

- The design space can be more fully explored

  varying prediction/hit rates and instruction stats., not limited to few points that benchmarks allow

  designer generates tables that drive simulation



Initial Results

- Comparison with detailed simplescalar simulation

  determine errors introduced and causes

  alternatively: what are important statistics?

- First, use real traces to determine errors introduced by probabilistic caches/branches

- Then, use synthetic traces to determine additional error

Real Traces

A: perfect branch prediction, perfect caches
- error in *compress* caused by memory RAW
- goes away as imperfections slow performance

B: real branch prediction, perfect caches
- no modeling of wrong speculations
- no modeling of read addresses (forwarding)

C: perfect branch prediction, real caches
- error due to no contention in memory system,

D: prob. branch prediction, perfect caches

E: real branch prediction, prob. caches

F: prob. branch prediction, prob. caches

Errors in D,E,F are generally caused by program "phases", and "clustering" of miss events -- not modeled in simple model

program	A	B	C	D	E	F
gcc	+0.1	+2.3	+2.5	+2.3	-68.6	-80.3
compress	-14.5	+2.8	-4.0	+22.7	-1.2	-20.7
go	-0.2	-4.7	+2.6	+2.7	-48.2	-52.4
ijpeg	-0.0	+0.0	-0.0	+0.0	+0.8	-27.2

Initial Results, Synthetic Traces

A: real trace, dependence matrix, perf. branch prediction, perf. caches

B: synthetic stream, perfect prediction and caches
- using mixes contributes more error than dependence matrix

C: synth. stream, prob. prediction, perf. caches
- little additional error introduced
- in fact, compensates for some error

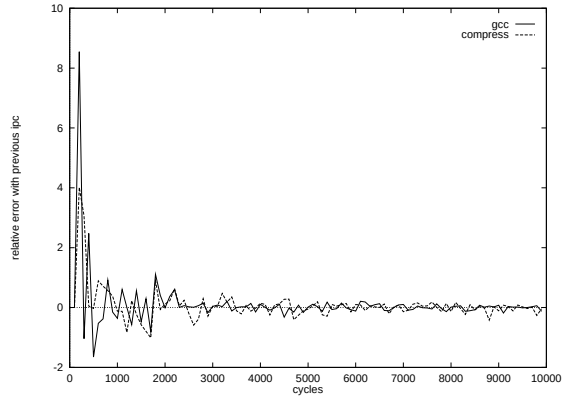
D: synthetic stream, prob. prediction, prob. caches
- rel. large errors probably due to clustering of cache misses and branch mispredictions

=> should look at higher-order models for branches and caches

program	A	B	C	D
gcc	+1.6	-2.0	-2.5	-82.1
compress	-7.4	-20.3	-26.2	-21.7
go	+7.3	-3.5	-3.7	-69.9
ijpeg	+6.4	-14.4	-14.2	-12.8
li	-5.8	-9.3	-19.0	-12.1
m88ksim	+0.0	-18.9	-10.4	-23.7
applu	-2.0	-21.6	-13.2	-13.2
tomcatv	-11.8	-34.2	-10.8	-43.5

Convergence characteristics

- Consider change in relative performance
- Terminate when change goes below some threshold
- Convergence in a few thousand cycles



Related Work

- Noonburg and Shen, HPCA ’97

  trace generated stats,

  analytical model => simple processors

  target appears to be processor design

- Dubey, Adams, and Flynn, April ’94 Trans. on Computers

  analytical model

  focuses on dependence modeling

  target appears to be processor design

- Sorin et al., ISCA ’98

  target is system design

  analytical model based on mean value analysis

  processor’s memory interface characterized via fast simulation

- Smith and Taylor, ICS ’92

  uses programmed automata to generate synthetic traces

  target is vector system design

- Agarwal, Horowitz, Hennessy, May ’89 TOCs

  analytical cache model

  could be useful for eliminating implementation dependence of cache stats.

Future Work

- Work toward greater accuracy
  - consider clustering of misses/mispredicts
  - deal with problems of consistency in synthetic generation
- Make cache/mispredict behavior more implementation independent
- Consider program phasing
  - as sources of error
  - to implement deterministic sampling
- System simulations
  - - independent jobs -- transaction processing
  - - parallel jobs -- cache coherence events
- Benchmark characterization

This work was funded by an IBM Partnership Award