Statistical Simulation of Superscalar Architectures using Commercial Workloads

Lieven Eeckhout and Koen De Bosschere

Dept. of Electronics and Information Systems (ELIS)
Ghent University, Belgium

CAECW'01, January 21, 2001

Outline

- Introduction
- Statistical Simulation
  - Statistical profiling
  - Synthetic trace generation
- Methodology
- Evaluation
- Conclusion

Introduction

- Architectural simulation
  - trace-driven or execution-driven
  - accurate
  - long simulation times
  - long traces to be stored
- Need for fast simulation techniques
  - take part of a full trace
  - analytical modeling
  - trace sampling
  - statistical simulation

Goal

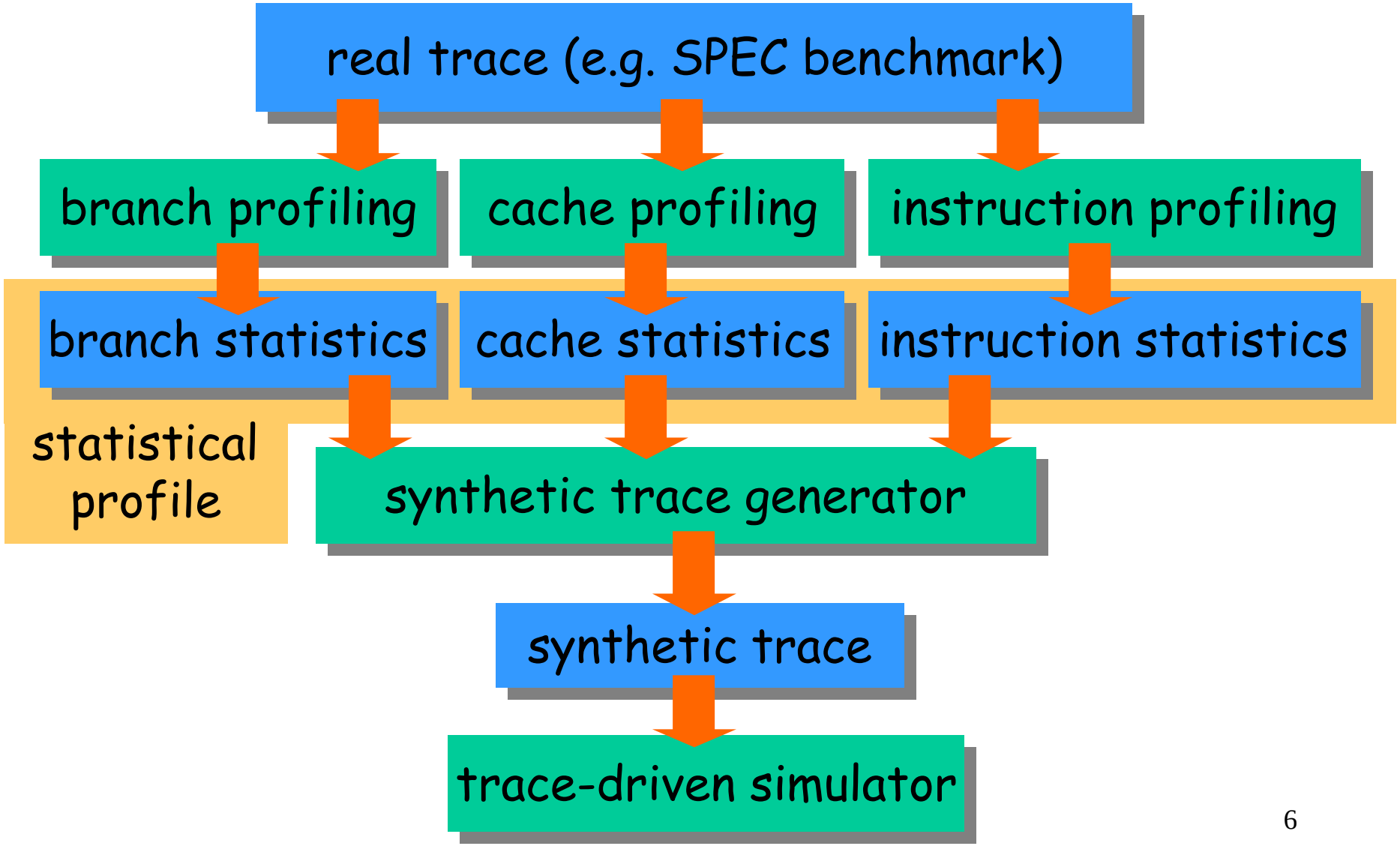
- Previous work used SPEC benchmarks to evaluate statistical simulation
- In this talk we use both commercial and scientific workloads
  - SPECint, SPECfp, system traces, multimedia, X graphics, database

Statistical Simulation

- Three steps:
  - extract statistical profile from a program execution
  - generate synthetic trace from it
  - simulate on a trace-driven simulator
- Two major advantages:
  - statistical profile is more compact than full trace
  - fast simulation due to statistical nature

  $\Rightarrow$ design space exploration in limited time

Statistical Simulation



Statistical Profiling

- Microarchitecture-independent statistics
  - instruction statistics
- Microarchitecture-dependent statistics
  - branch statistics
  - cache statistics
- Result: statistical simulation only to explore design options of processor core (cache and branch predictor are fixed)

Statistical Profiling
Instruction Statistics

- Instruction mix (13 classes)
- Number of register operands
- Age of register operands
  - probability that register operand was produced δ instructions before it in the trace (only RAW)
- Memory dependencies
  - probability that load is memory-dependent on the δ-th store before it in the trace (only RAW)

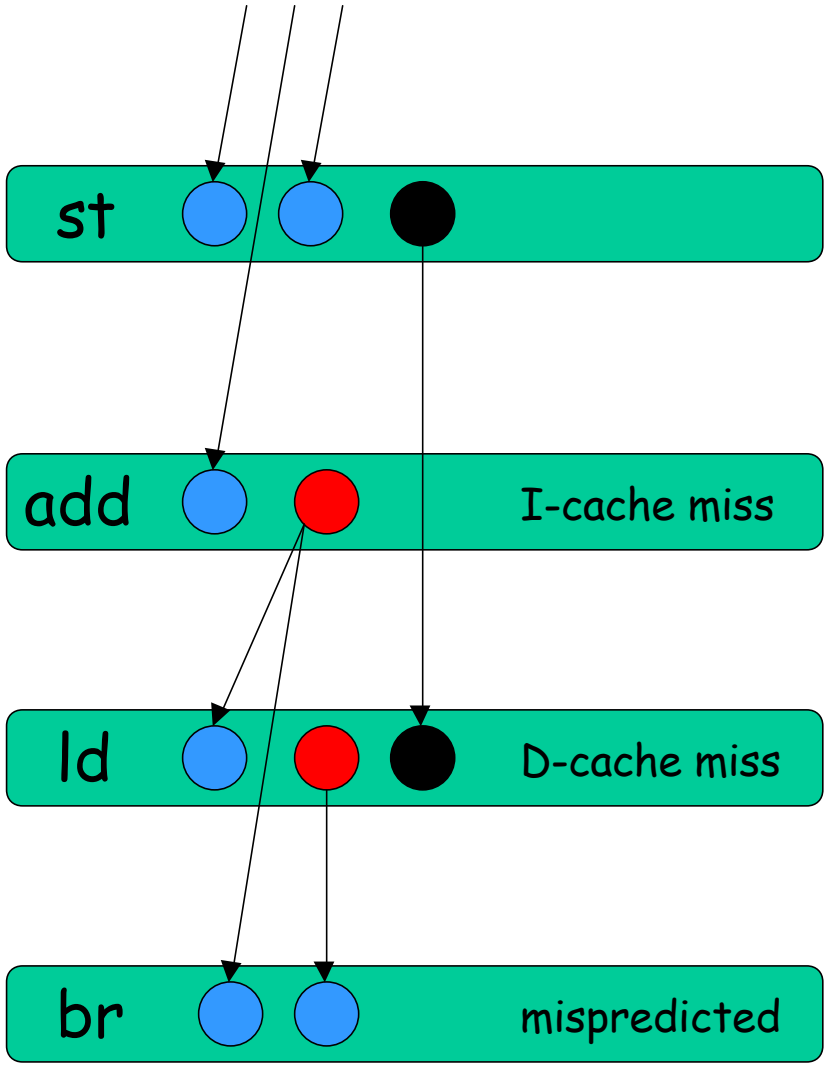
Statistical Profiling
Branch Statistics

- Six branch types
  - conditional branch, unconditional branch, call with offset, indirect jump, indirect call, return
- Distinction
  - branch prediction accuracy: refill pipeline on branch misprediction
  - branch target prediction accuracy: single-cycle bubble in pipeline on correct branch prediction but target misprediction

Statistical Profiling
Cache Statistics

- D-cache statistics
  - L1 D-cache miss rate
  - L2 D-cache miss rate
- I-cache statistics
  - L1 I-cache miss rate
  - L2 I-cache miss rate

Synthetic Trace Generation



Instruction-by-instruction through random number generation

Determine
- instruction type
- number of operands
- age of register operands
- memory dependency
- branch behavior
- D-cache behavior
- I-cache behavior

Methodology: microarchitecture

- Out-of-order processor
  - 8 and 16 issue
  - windows of 64 and 128 instructions
- McFarling branch predictor
- 'small' cache configuration
  - 8KB DM L1 I-cache, 8KB DM L1 D-cache, 64KB 2WSA unified L2 cache
- 'large' cache configuration
  - 32KB DM L1 I-cache, 64KB 2WSA L1 D-cache, 512KB 4WSA unified L2 cache
- Access time
  - L1 I-cache (1 cycle), L1 D-cache (2 cycles), L2 cache (10 cycles), main memory (80 cycles)

Methodology: benchmarks

- 8 SPECint95 benchmarks
- 5 SPECfp95 benchmarks (hydro2d, su2cor, swim, tomcatv, wave5)
- 8 IBS system traces (mpeg, jpeg, gs, verilog, gcc, sdet, nroff, groff)
- 4 MediaBench applications (g721, gs, gsm, mpeg2)
- 4 X graphics benchmarks (DooM, POVRay, Xanim, Quake)
- 2 TPC-D queries running on Postgres 6.3

$\Rightarrow$ ~ 200 million instructions / trace

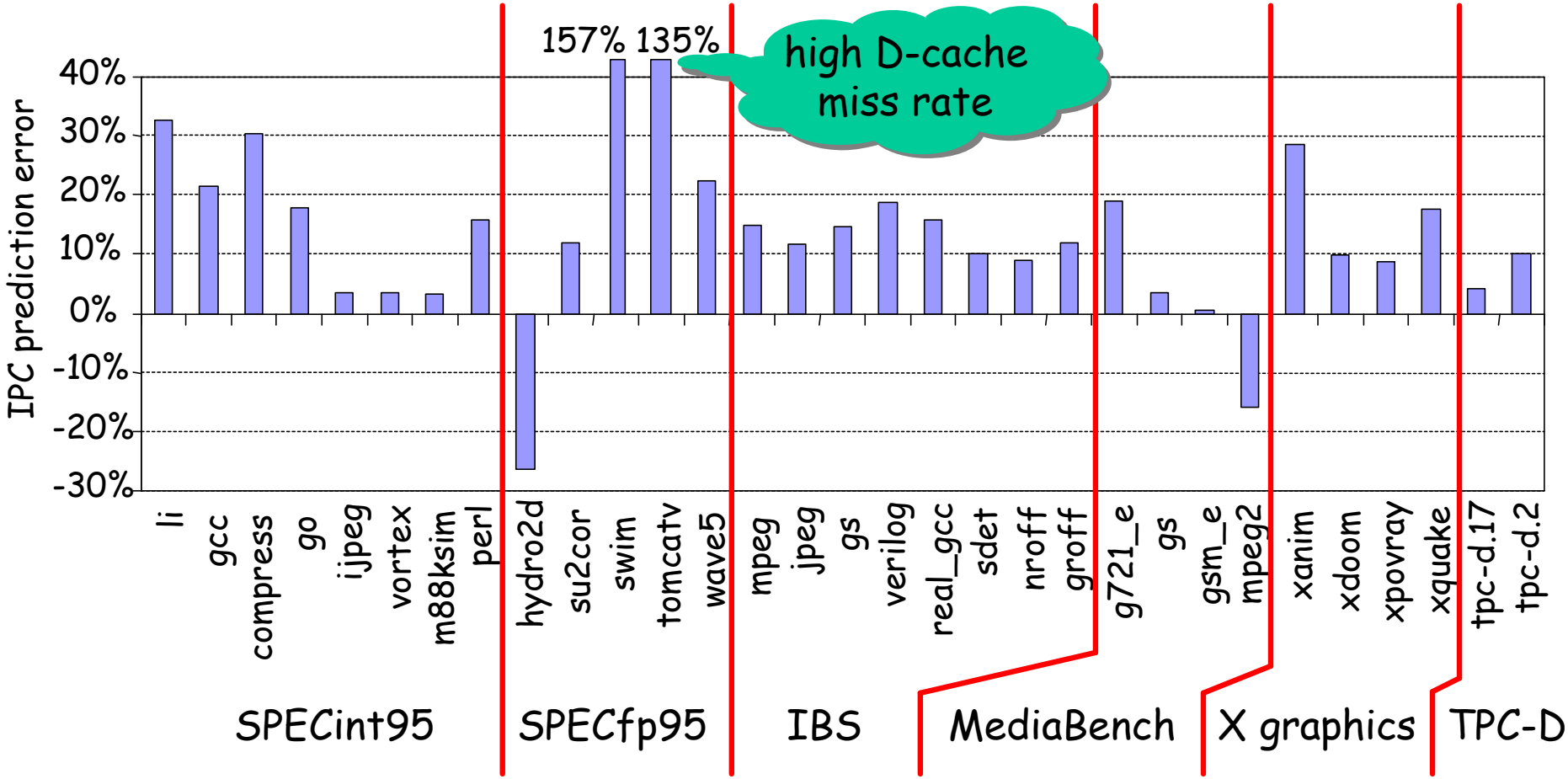
Evaluation

- IPC prediction error =

$$\frac{\text{IPC real trace - IPC synthetic trace}}{\text{IPC real trace}}$$

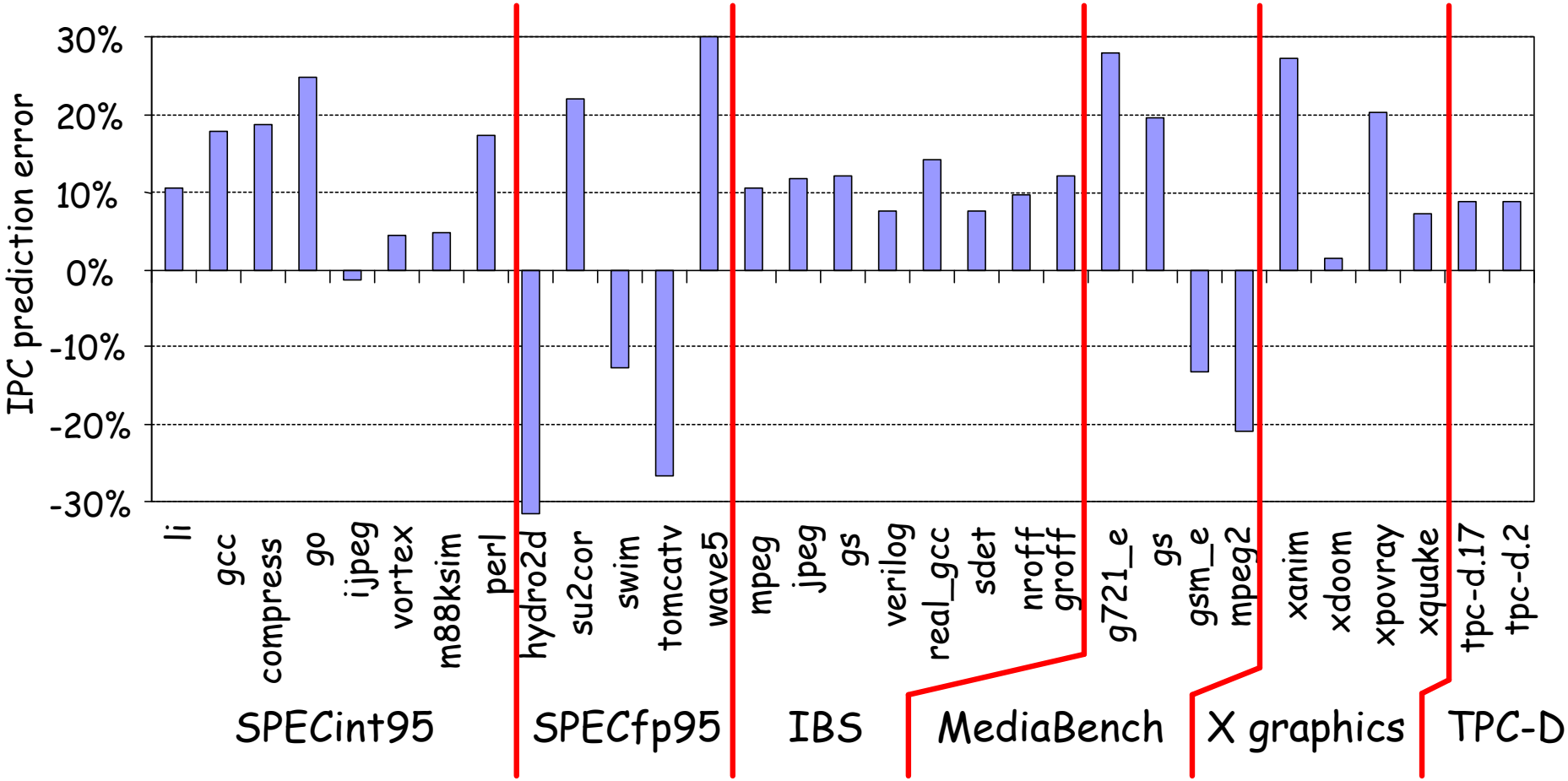
- IPC real trace = IPC when running real trace on trace-driven simulator
- IPC synthetic trace = IPC when running synthetic trace generated from the statistical profile of the real trace
- Simulation speed: $s_{IPC}/\bar{x}_{IPC}$ less than 1% after simulating 1 million instructions

IPC prediction error (1)



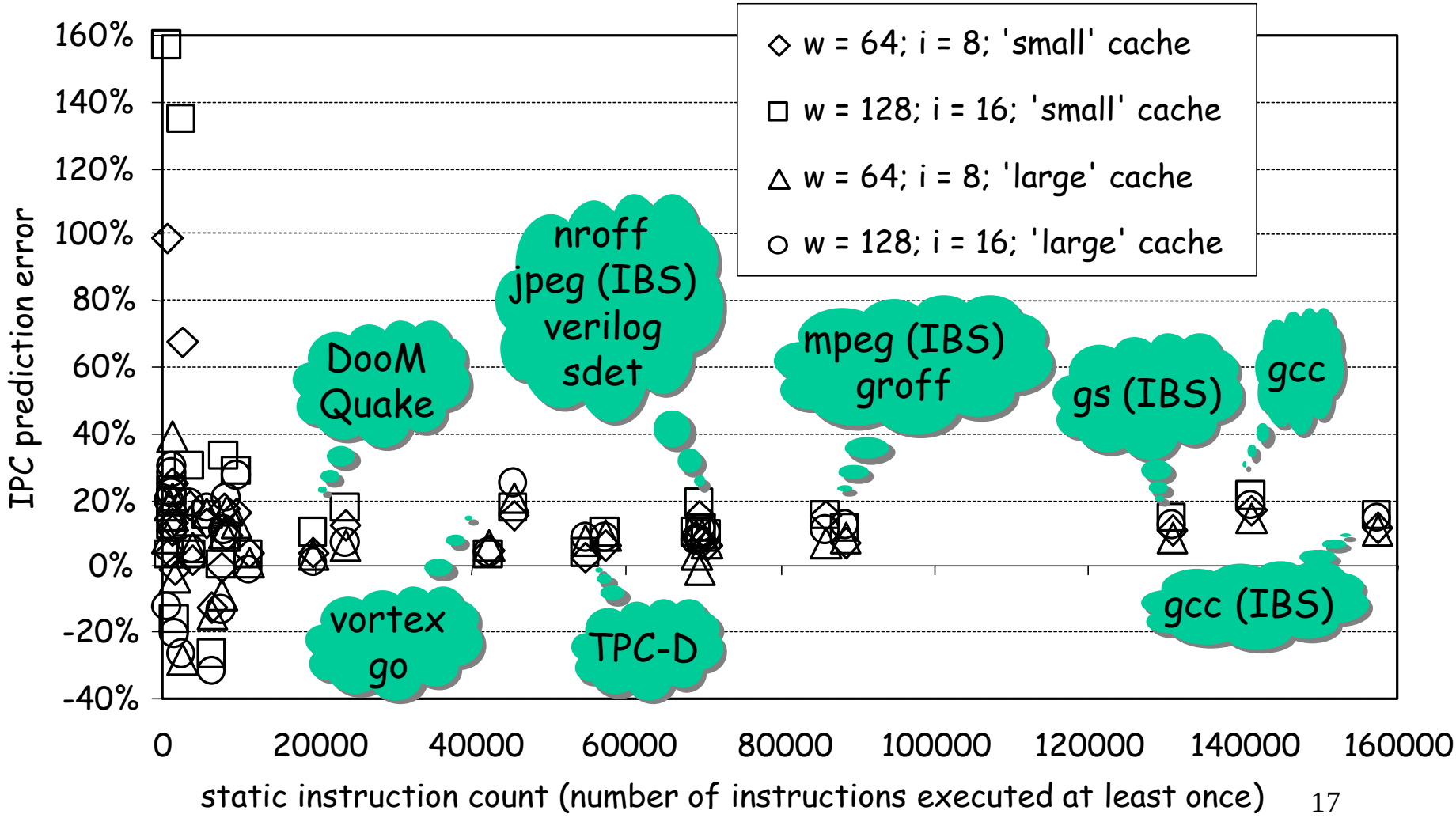
16-issue, 128-entry window, 'small' cache configuration

IPC prediction error (2)



16-issue, 128-entry window, 'large' cache configuration

IPC prediction error vs. static instruction count



Conclusion (1)

- Higher IPC prediction errors for applications with smaller static instruction count:
  - MediaBench applications
  - SPECfp95 benchmarks
  - 2 X graphics benchmarks (POVRay and Xanim)
  - 5 SPECint95 benchmarks

Conclusion (2)

- Smaller IPC prediction errors for applications with larger instruction footprint:
  - IBS system traces
  - TPC-D traces
  - 2 X graphics benchmarks (DooM and Quake)
  - 3 SPECint95 benchmarks (go, gcc, vortex)

  $\Rightarrow$ IPC prediction error between -1% and 25%

Conclusion (3)

- Statistical simulation is a useful fast simulation technique for commercial workloads
  - due to higher variability in instructions
  - since commercial workloads have larger instruction footprint
  - which makes a statistical technique more powerful