Performance Issues in Correlated Branch Prediction

Nicolas Gloy, Cliff Young, Michael D.Smith

Division of Applied Sciences
Harvard University

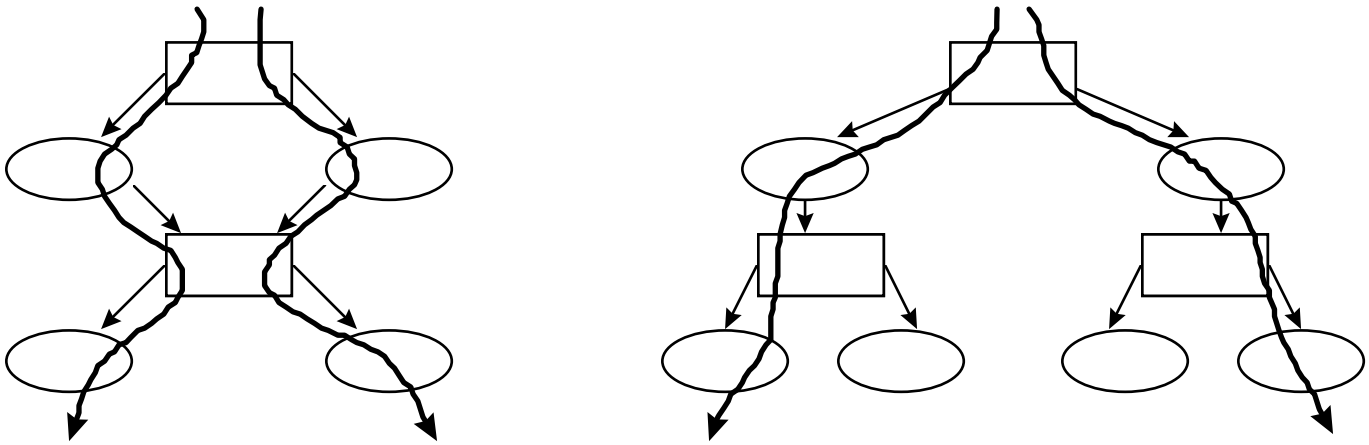
What is Branch Correlation ?

- Probability of branch directions may be influenced by neighboring branches

```
if ( foo % 2 == 0 )
     ++div2;
if ( foo % 3 == 0 )
     ++div3;
if ( foo % 5 == 0 )
     ++div5;
if ( foo % 6 == 0 )
     ++div6;
```

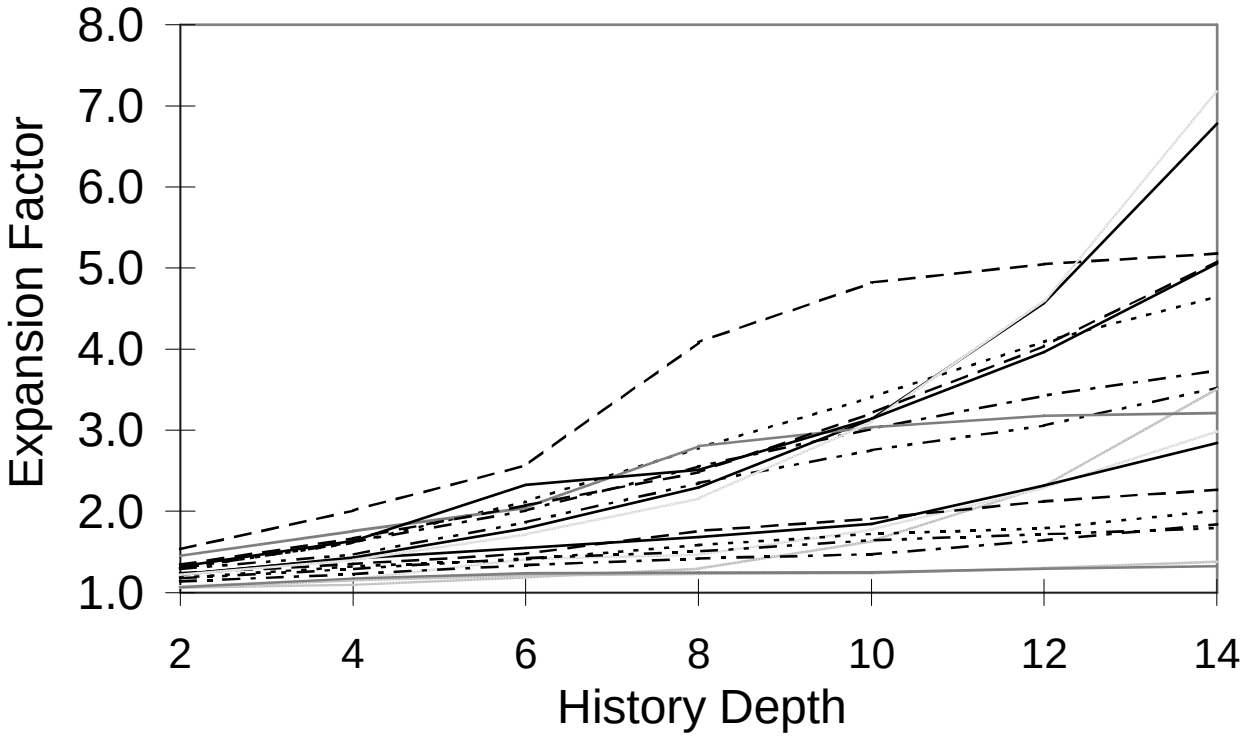
Our Static Scheme

- Use static prediction bits in branch instructions
- Profile to collect correlation data
- Encode correleation into the program counter by *duplicating* program basic blocks:



- See ASPLOS 94 paper for more details

Code Expansion (executed code only)



- Increasing history depth increases code expansion
- Performance tradeoff:

  fewer prediction misses for more I-cache misses

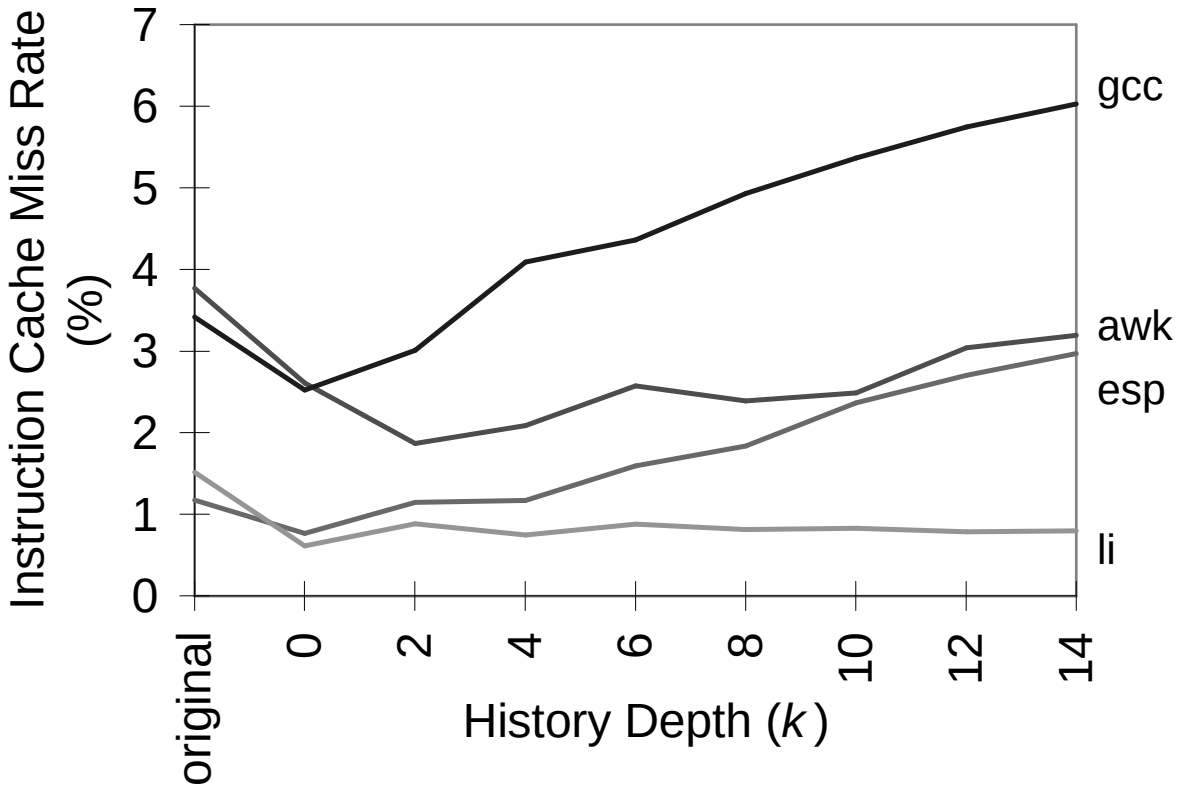
Outline

- Code layout and effects on instruction cache miss rate
- Performance metric
- Processor models
- Performance results

Code Layout

- Optimize layout in memory to
  - reduce conflict misses in the instruction cache
  - avoid misfetch penalties
- Necessary to allow comparison between different versions of the same program
- Pettis & Hansen [1990] reorganize the program text through
  - procedure positioning
  - basic block reordering / branch alignment
  - procedure splitting
- Techniques based on profile information

Effects on Instruction Cache Miss Rate



- Cache parameters: 8KB, direct-mapped, 32 byte line size
- Overall trends:
  - Miss rate increases as history depth increases
  - miss rate at large history depth can be less than original miss rate

Measuring Impact on Performance

- Metric: avg. cycles saved per 1000 instr. executed

  $=$ misprediction_penalty $\times$ (Δ no_of_mispredictions)
  + misfetch_penalty $\times$ (Δ no_of_misfetches)
  + L1_miss_penalty $\times$ (Δ no_of_L1_icache misses)

- Baseline: identical machine microarchitecture with
  - profiled branch prediction
  - no code layout

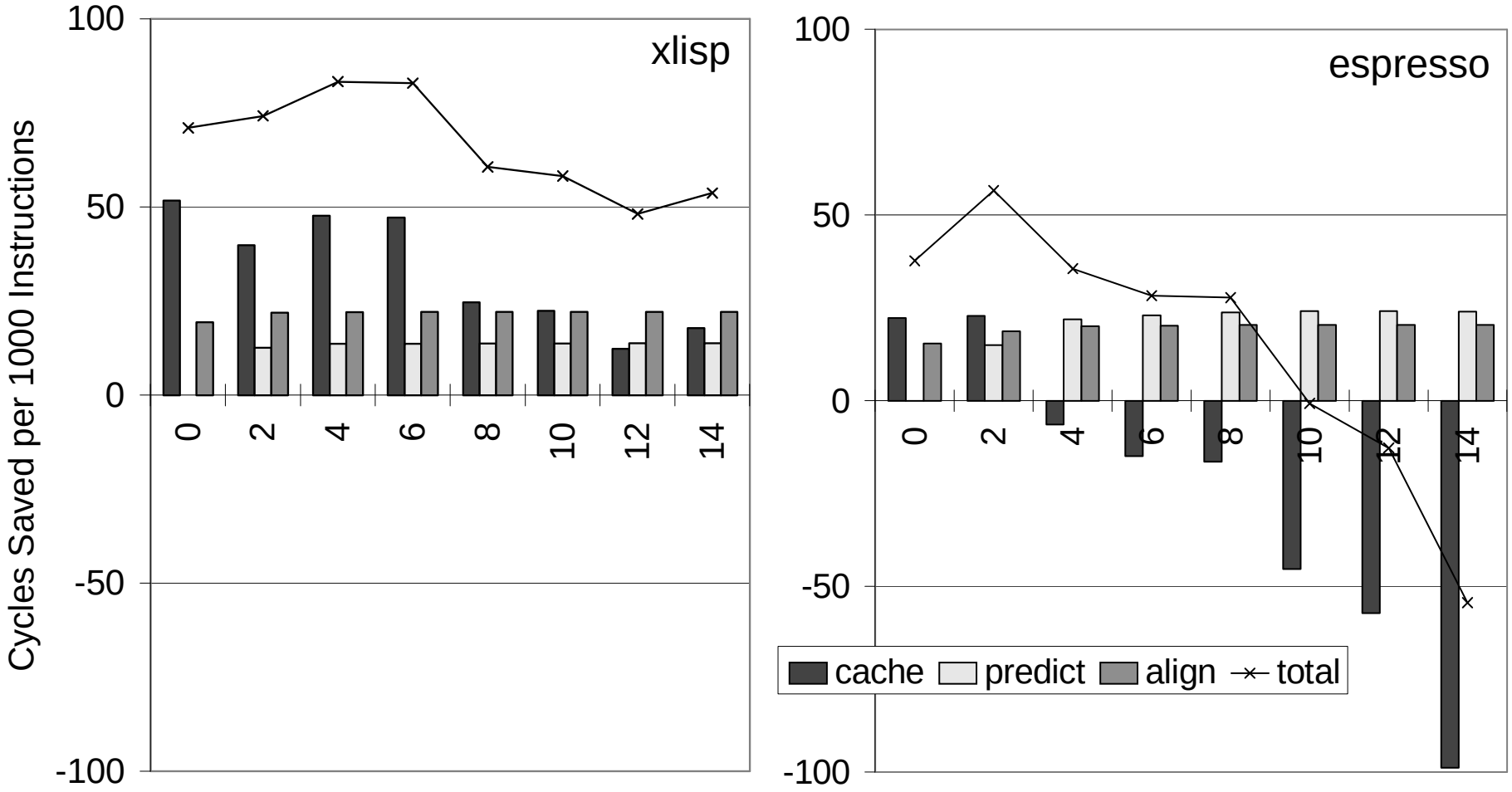
- Our metric
  - does not represent overall performance
  - does capture important components of performance affected
  - is independent of the rest of the processor organization

Processor Models Used in Simulations

Processor model	mispredict penalty	misfetch penalty	L1 cache miss penalty	size of L1 I-cache
DEC Alpha 21164-like	5	1	6	8KB
Intel P6-like	≥11	0	3	16KB
HP PA-8000-like	5	2	30	256KB

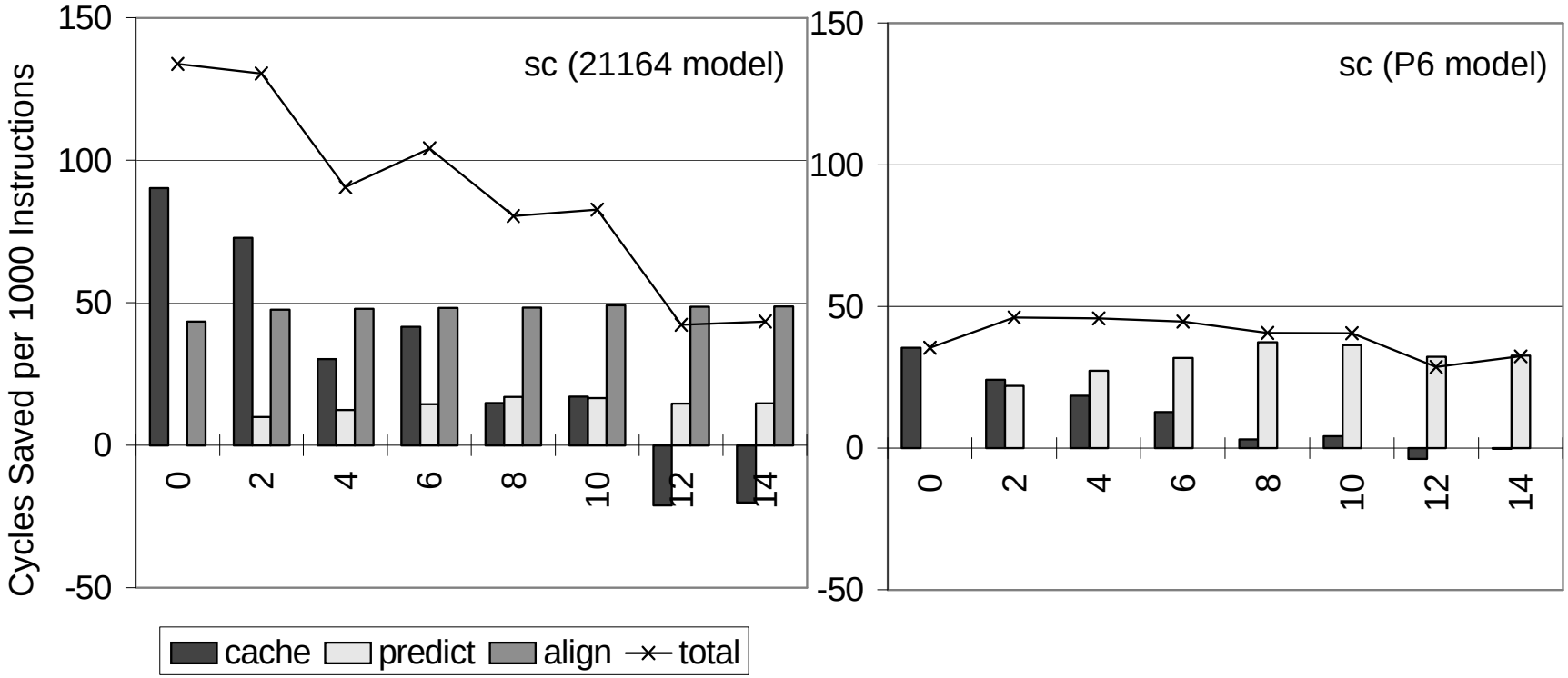
- Chosen because high misprediction penalties demand aggressive branch prediction schemes

Performance results (1)



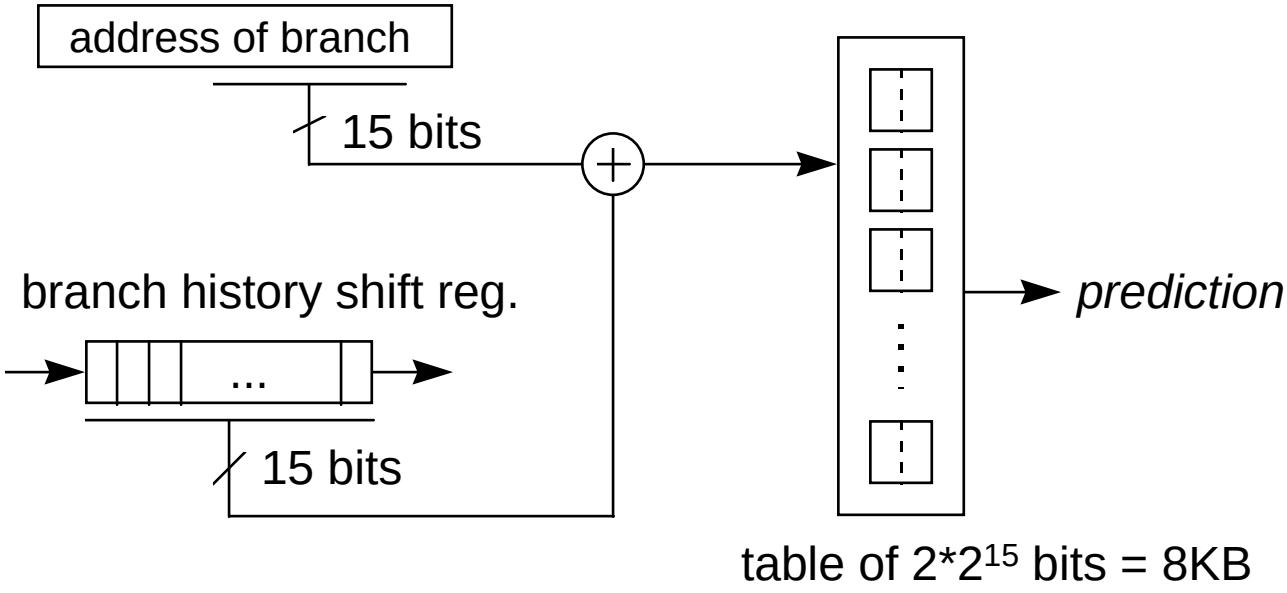
- two representative results for 21164-like machine model
- in general, observe a bell-shaped curve for metric vs. history depth

Performance results (2)



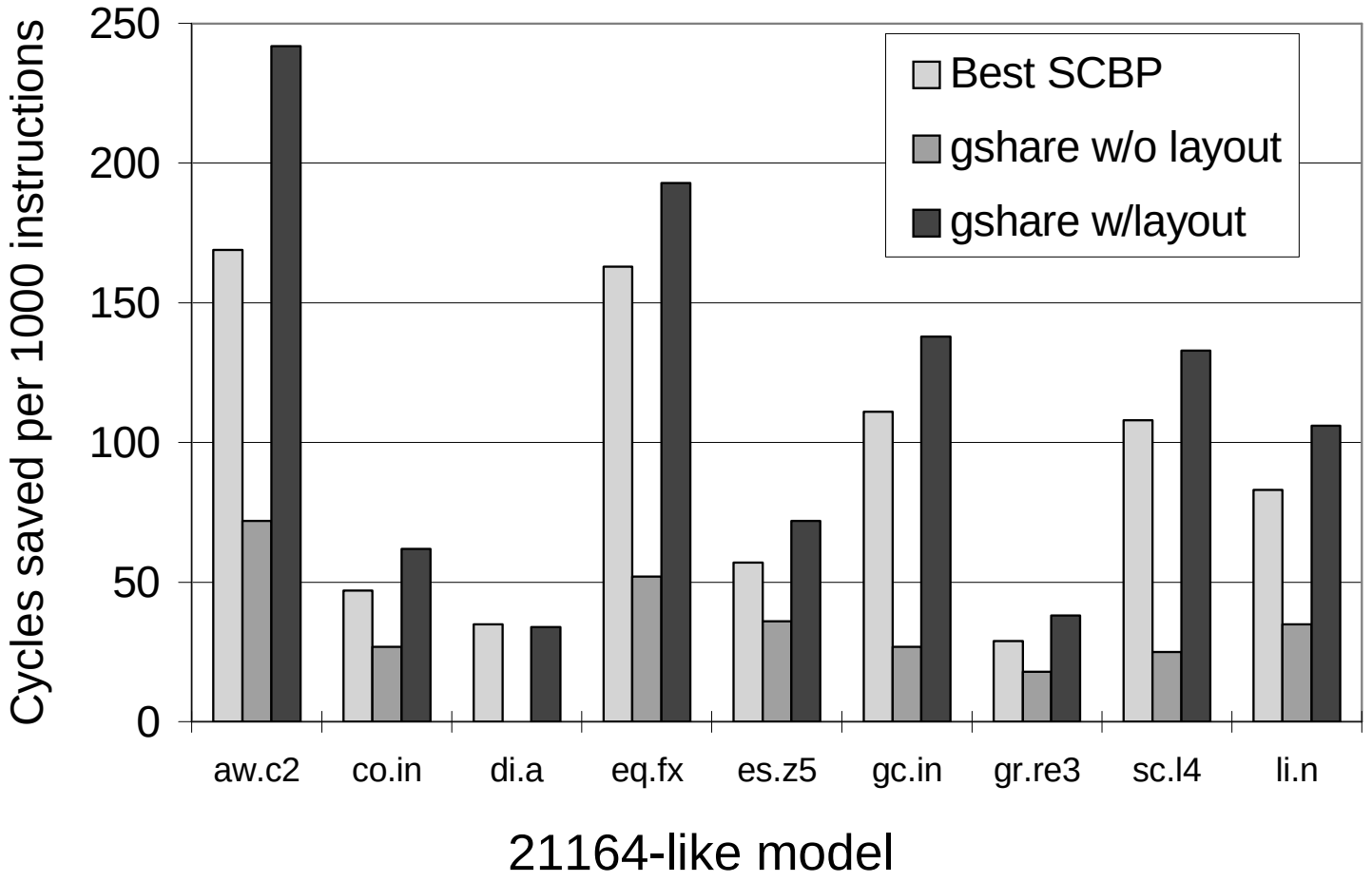
- Best history depth value gets bigger as
  - misprediction penalties increase
  - cache miss rates and/or miss penalties decrease

A Dynamic Prediction Scheme: *gshare*



- gshare -- McFarling [1993]
- best accuracy for global history schemes under constraint of fixed-size branch history table

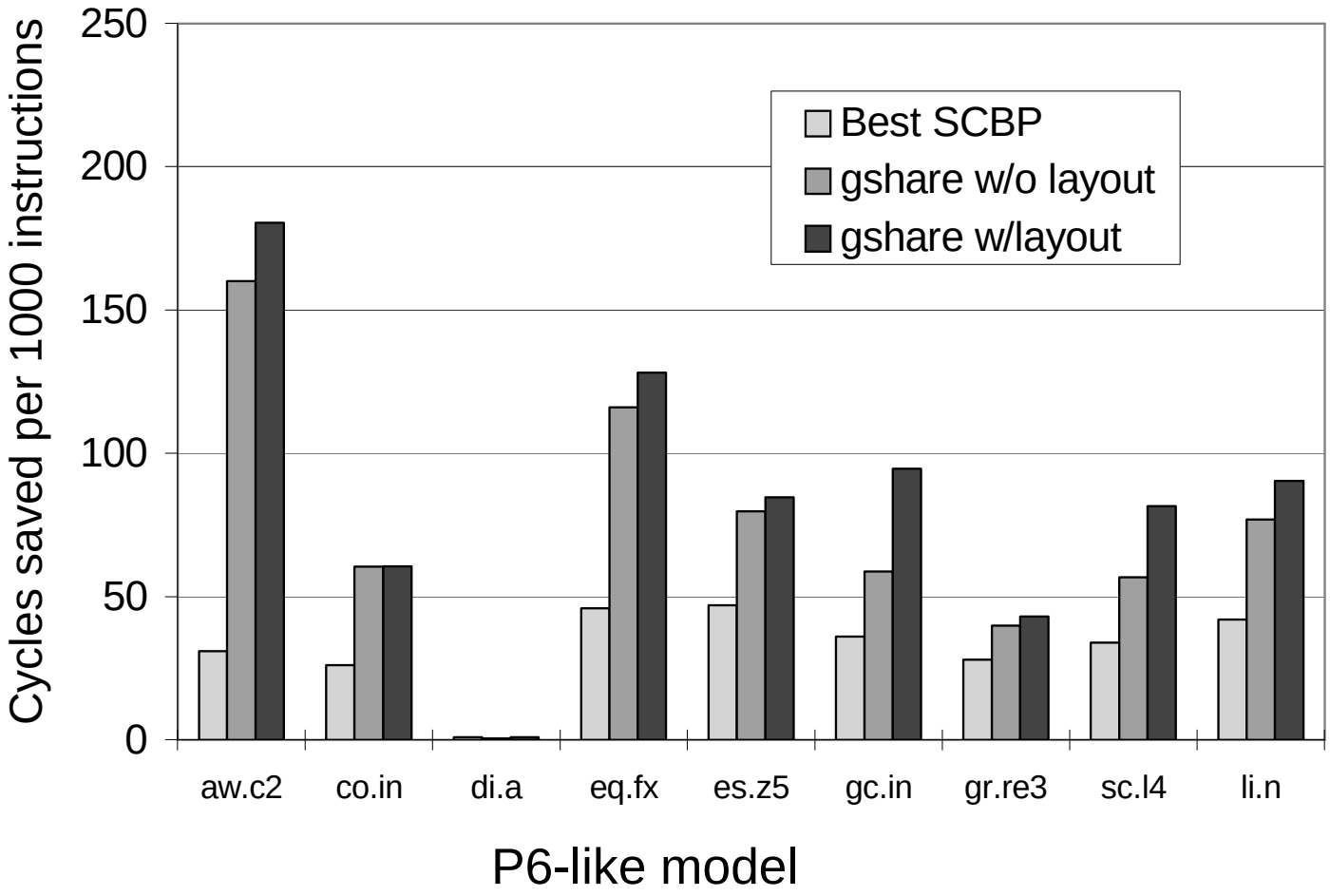
Comparison with *gshare*



gshare:
8KB table

- Code layout is important for dynamic schemes, too

Comparison with *gshare* (2)



gshare:
8KB table

- Prediction accuracy is more important under high misprediction penalty and smaller miss penalty

Summary

- SCBP can improve performance over simple static prediction
- Dynamic schemes can benefit from code layout
- Further performance benefits achievable from better compile-time prediction accuracy

For more information...

HUBE Research Group:
`http://www.eecs.harvard.edu/~hube`

Nicolas Gloy	`ng@eecs.harvard.edu`
Cliff Young	`cyoung@eecs.harvard.edu`
Michael D. Smith	`smith@eecs.harvard.edu`