Improving Trace Cache Effectiveness with Branch Promotion and Trace Packing

Sanjay J. Patel Marius Evers Yale N. Patt

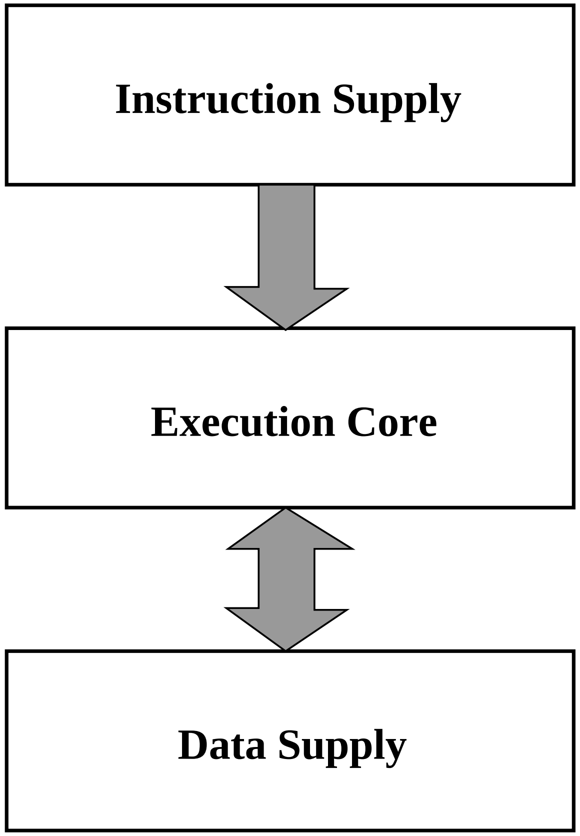
High Performance Systems Group
Advanced Computer Architecture Laboratory
University of Michigan
Ann Arbor, MI 48109-2122

http://www.eecs.umich.edu/HPS/

Outline

- High Bandwidth Fetch with a Trace Cache
- Experimental Model
- Branch Promotion
- Trace Packing
- Performance Implications
- Conclusions

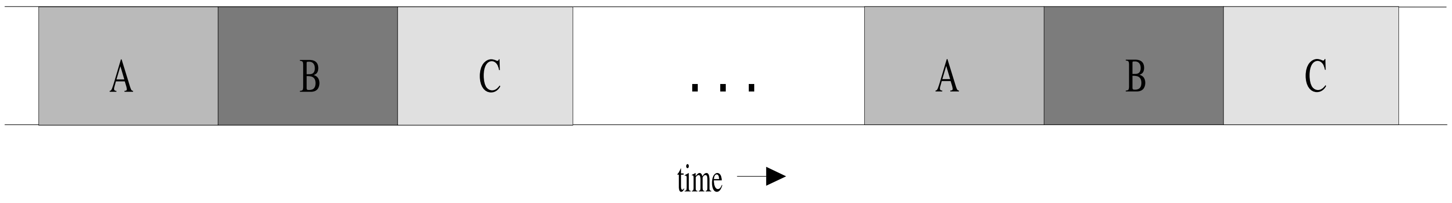
High Performance Processor Design



Maximizing Instruction Bandwidth

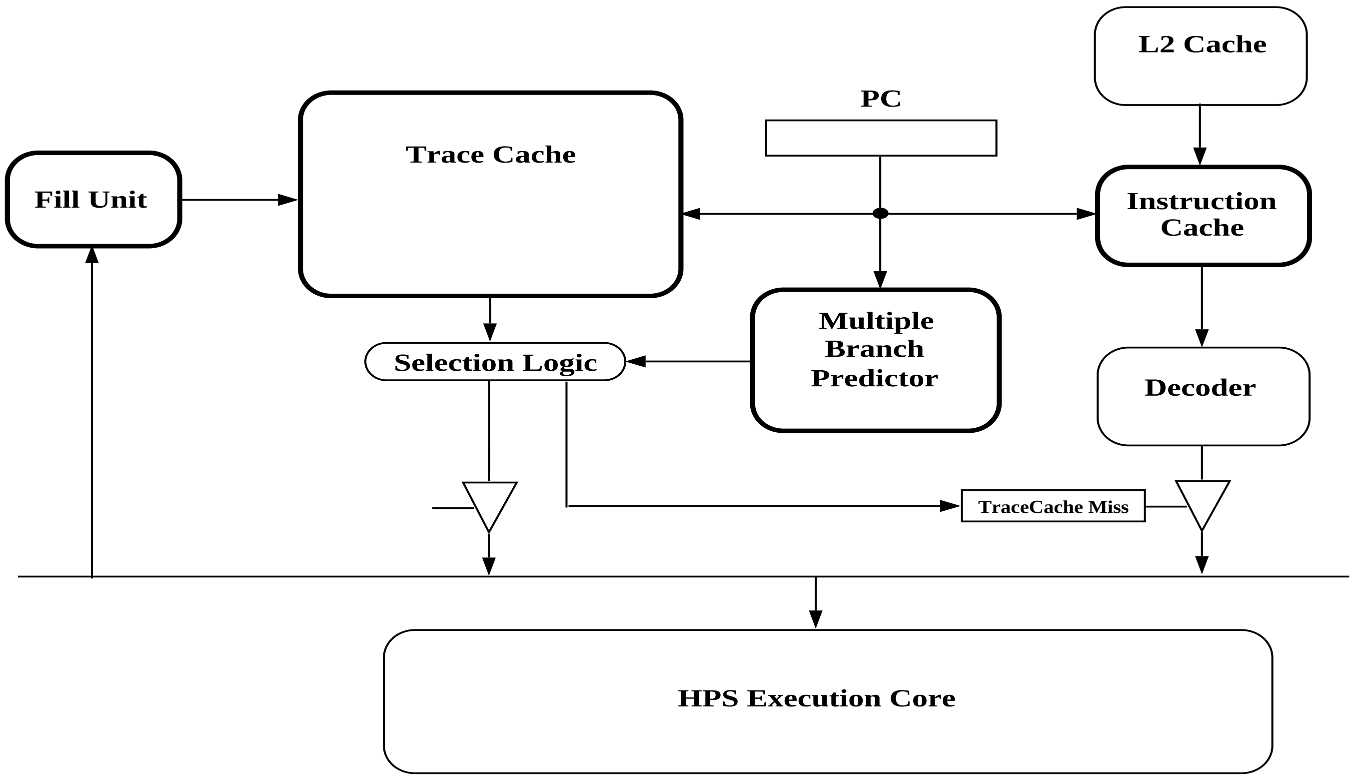
- No empty fetches (caused by cache misses)
- No fetches thrown away (caused by mispredictions)
- No partial fetches (caused by branches)

Caching the Dynamic Instruction Stream



Dynamic Instruction Stream

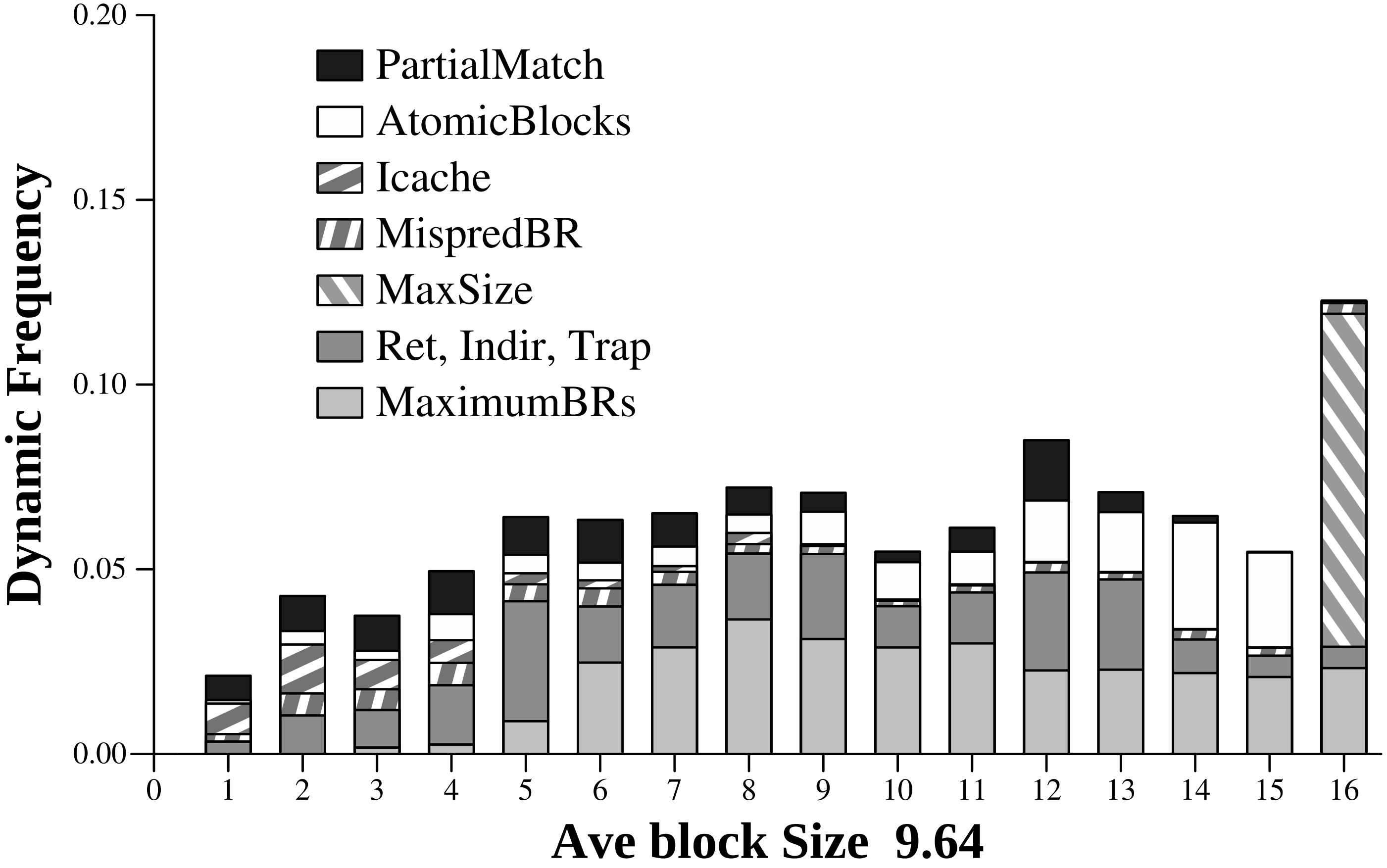
Trace Cache Fetch Mechanism



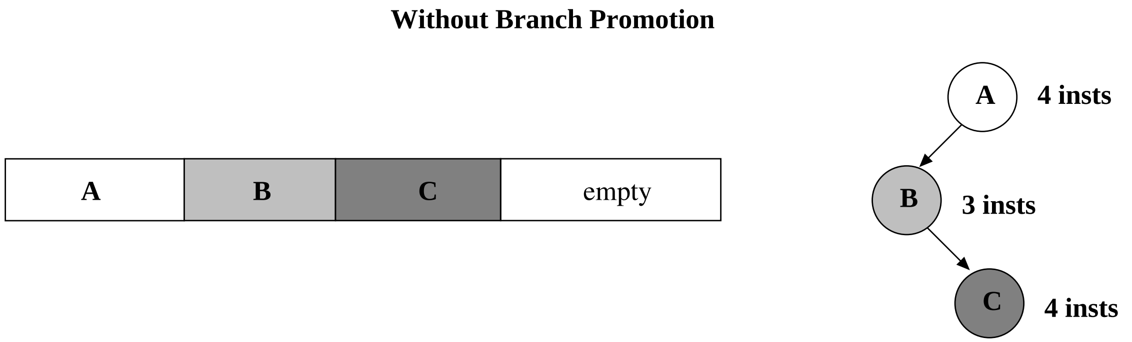
Experimental Model

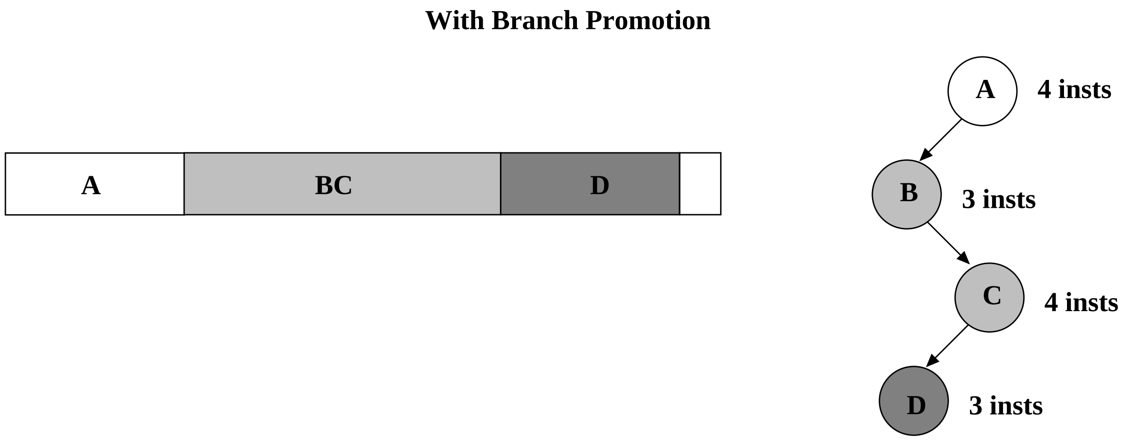
- Developed on simplescalar-2.0
- Executable-based, wrong path effects
- 5 stages : Fetch, Issue, Schedule, Execute, Retire
- 128KB Trace Cache, 4KB ICache
- 32KB gshare multiple branch predictor
- Partial Matching, Inactive Issue, Atomic Blocks

Limitations on trace size: gcc

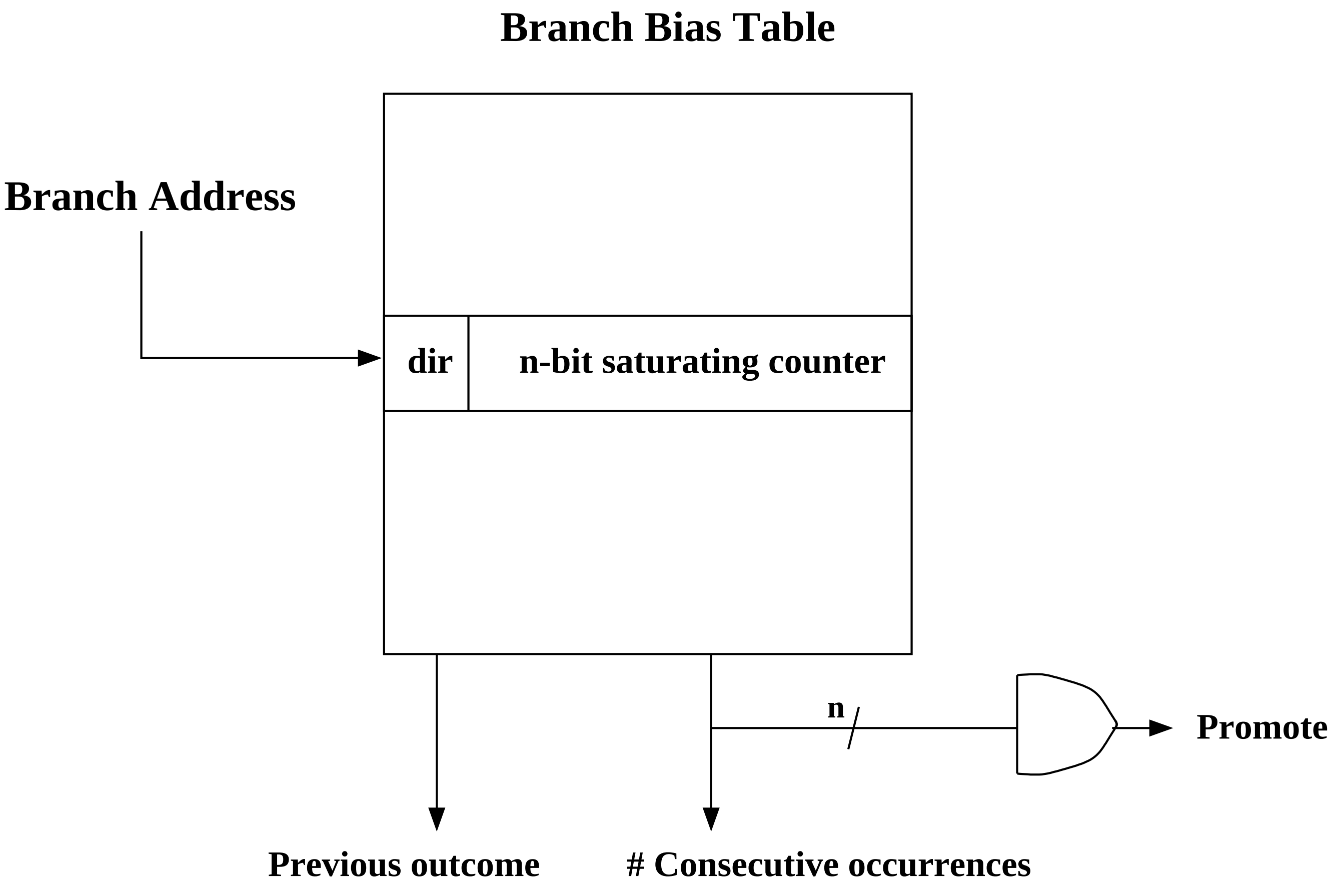


Branch Promotion

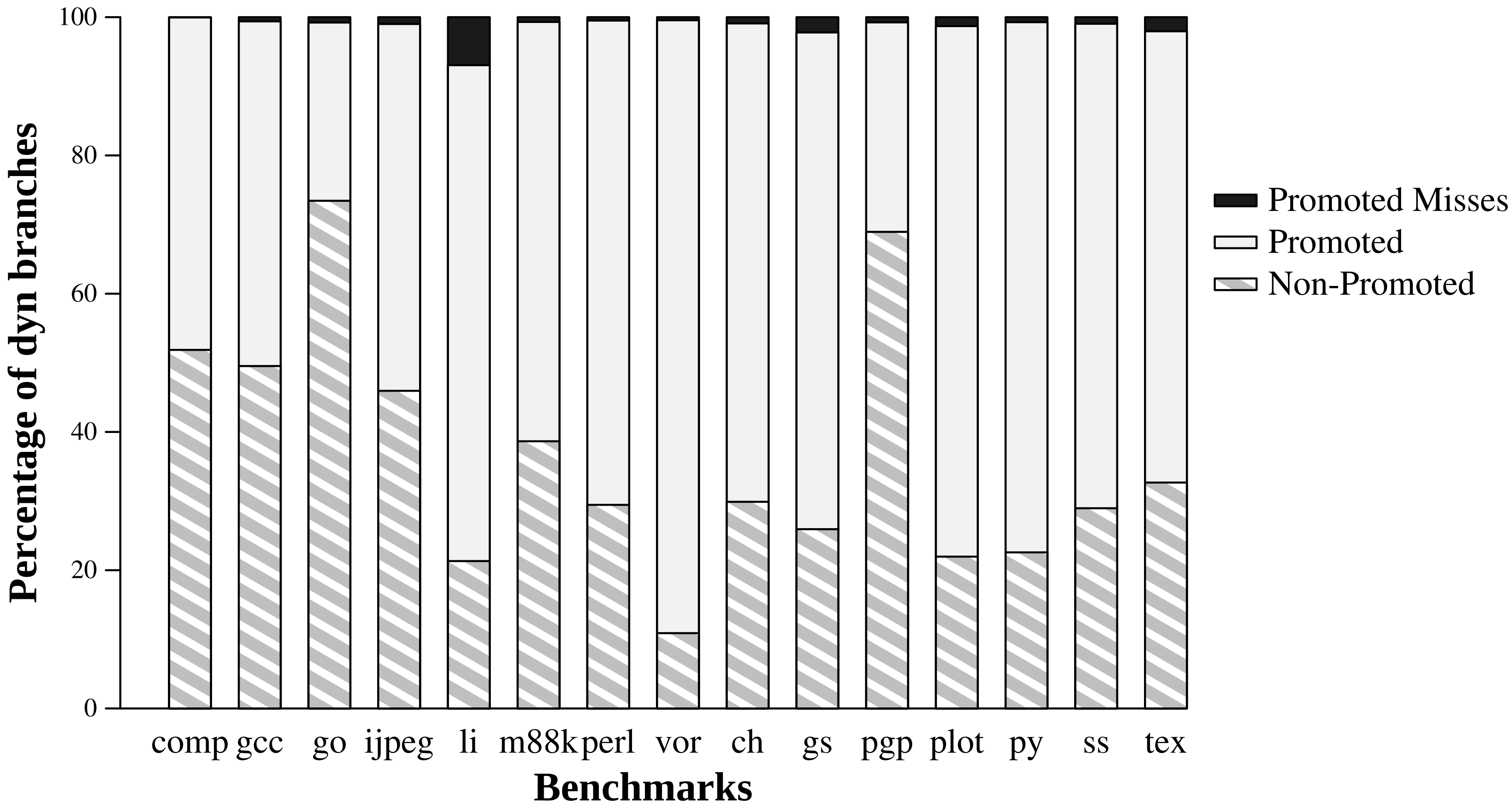




Implementing promotion



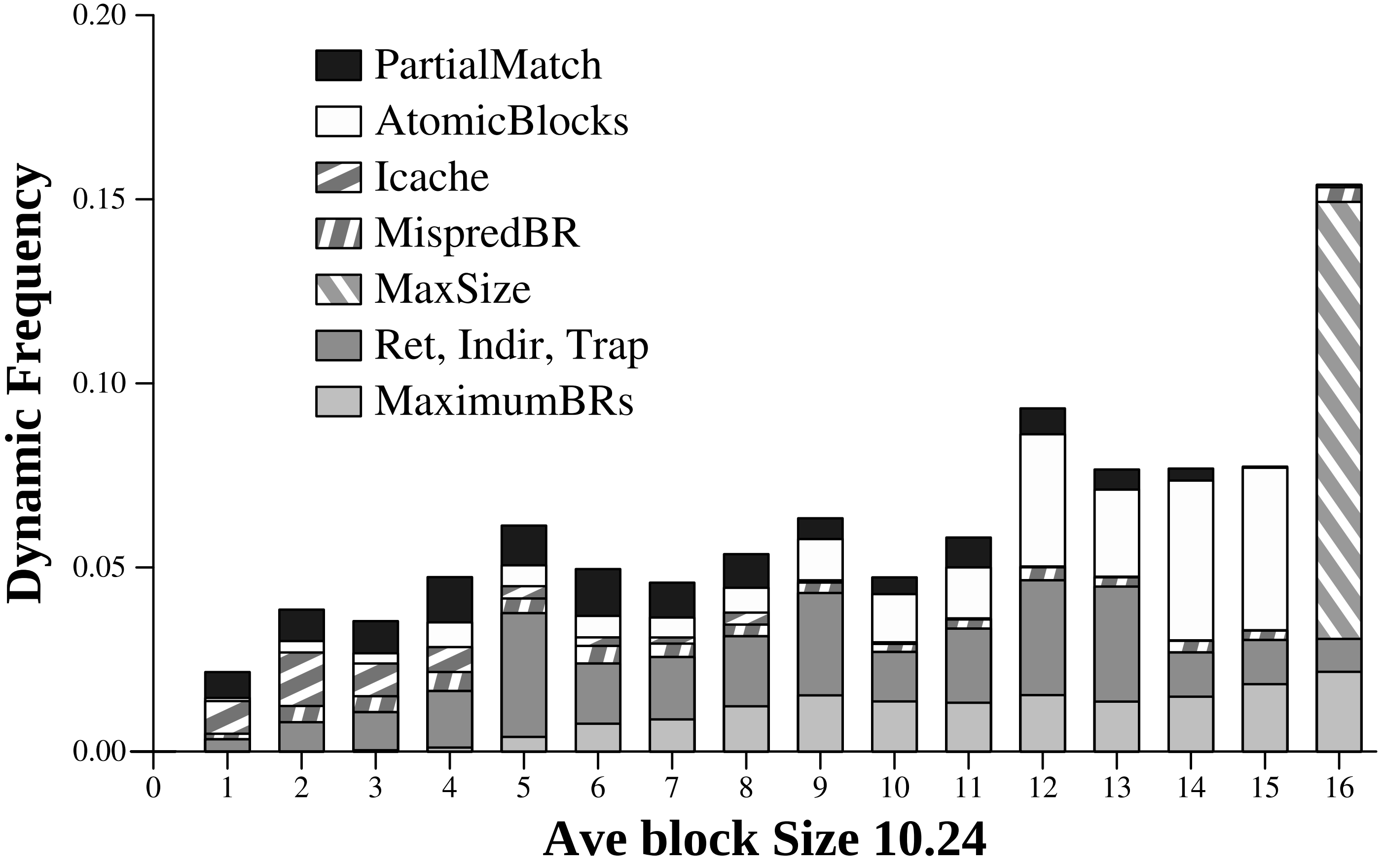
Promoted Branch Frequency



Number of dynamic predictions per fetch

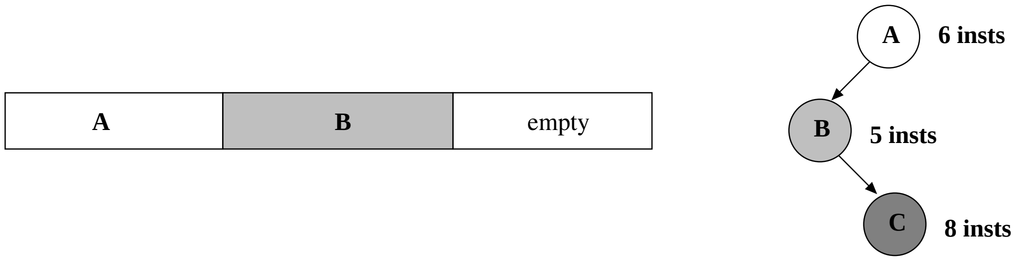
Configuration	0 or 1 predictions	2 predictions	3 predictions
w/o promotion	54%	18%	28%
w promotion	85%	12%	3%

Limitations on trace size: gcc with promotion

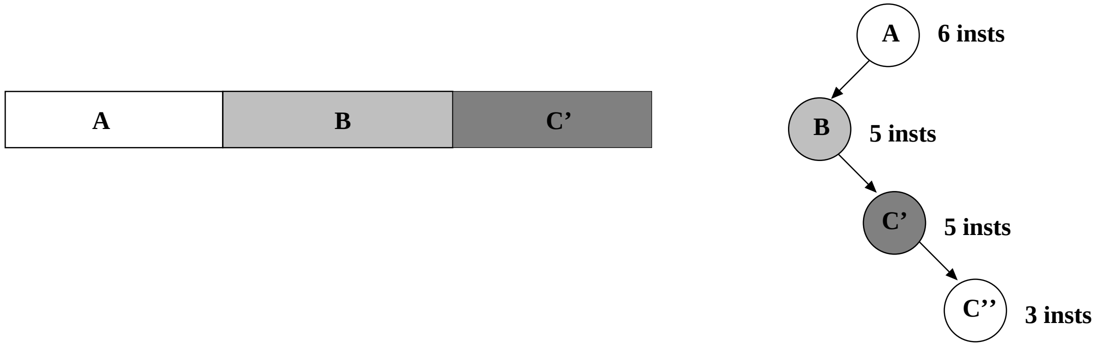


Trace packing

Without Trace Packing



With Trace Packing



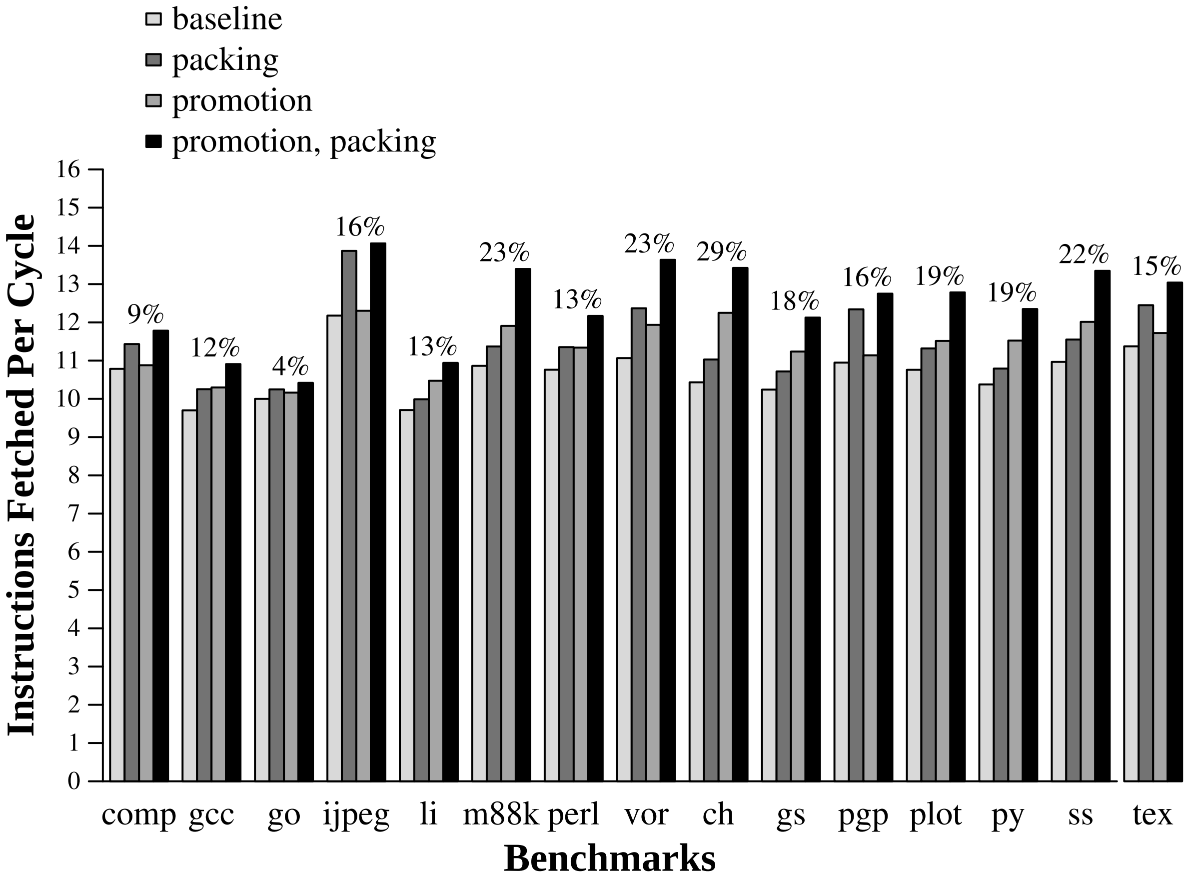
Two Packing Schemes

- n-Regulated
  - only pack chunks of n instructions
- Cost Regulated
  - only pack if unused space is large OR line contains backwards branch

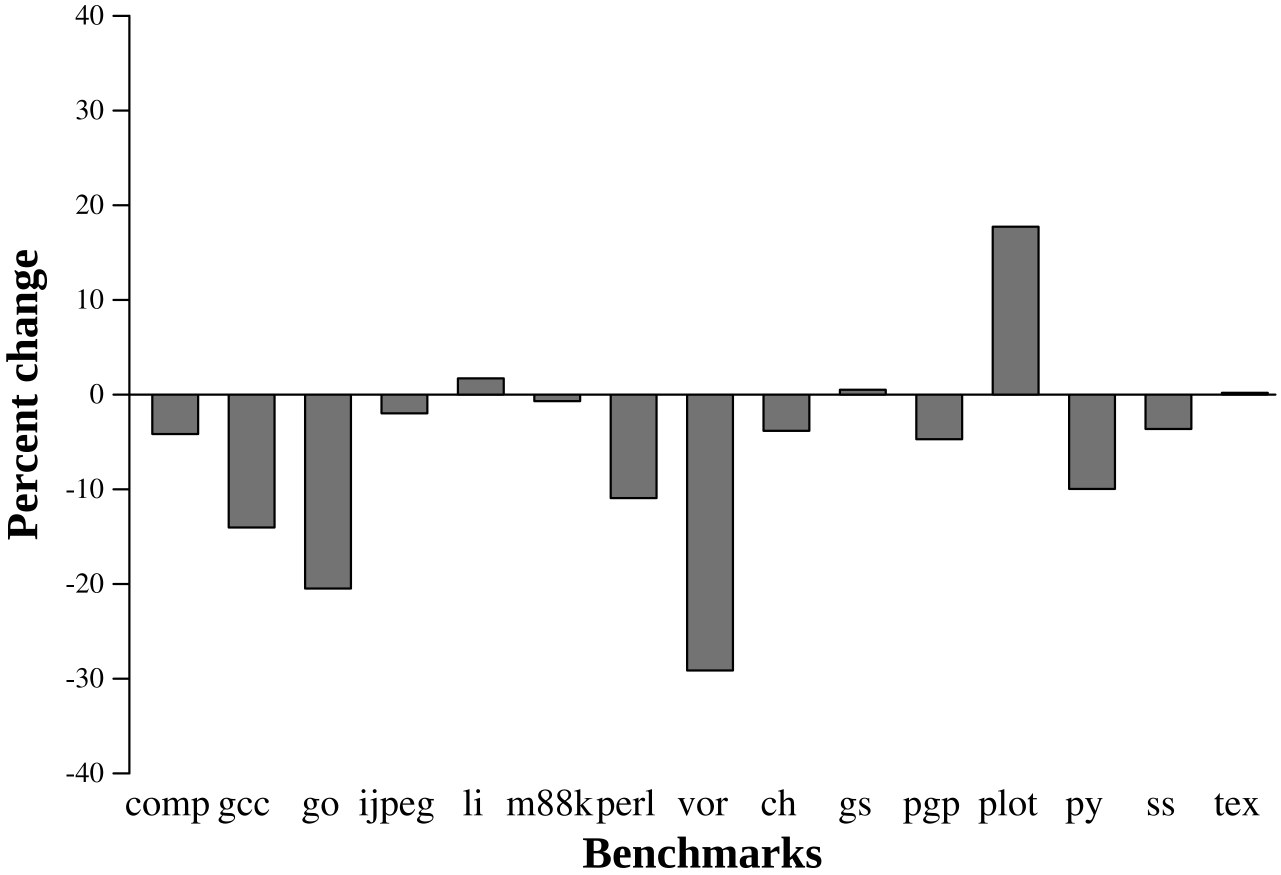
Percent Change in Cache Miss Cycles versus baseline

Benchmark	unreg	cost-reg	n = 2	n = 4
gcc	26.9%	13.2%	22.3%	15.8%
go	28.4%	11.6%	23.9%	15.9%
vortex	18.1%	15.0%	11.1%	4.5%
gs	29.5%	16.2%	22.8%	14.1%
python	38.9%	1.5%	18.2%	13.0%
tex	95.6%	39.5%	74.6%	52.8%
Ave Eff Fetch Rate	12.47	12.23	12.42	12.18

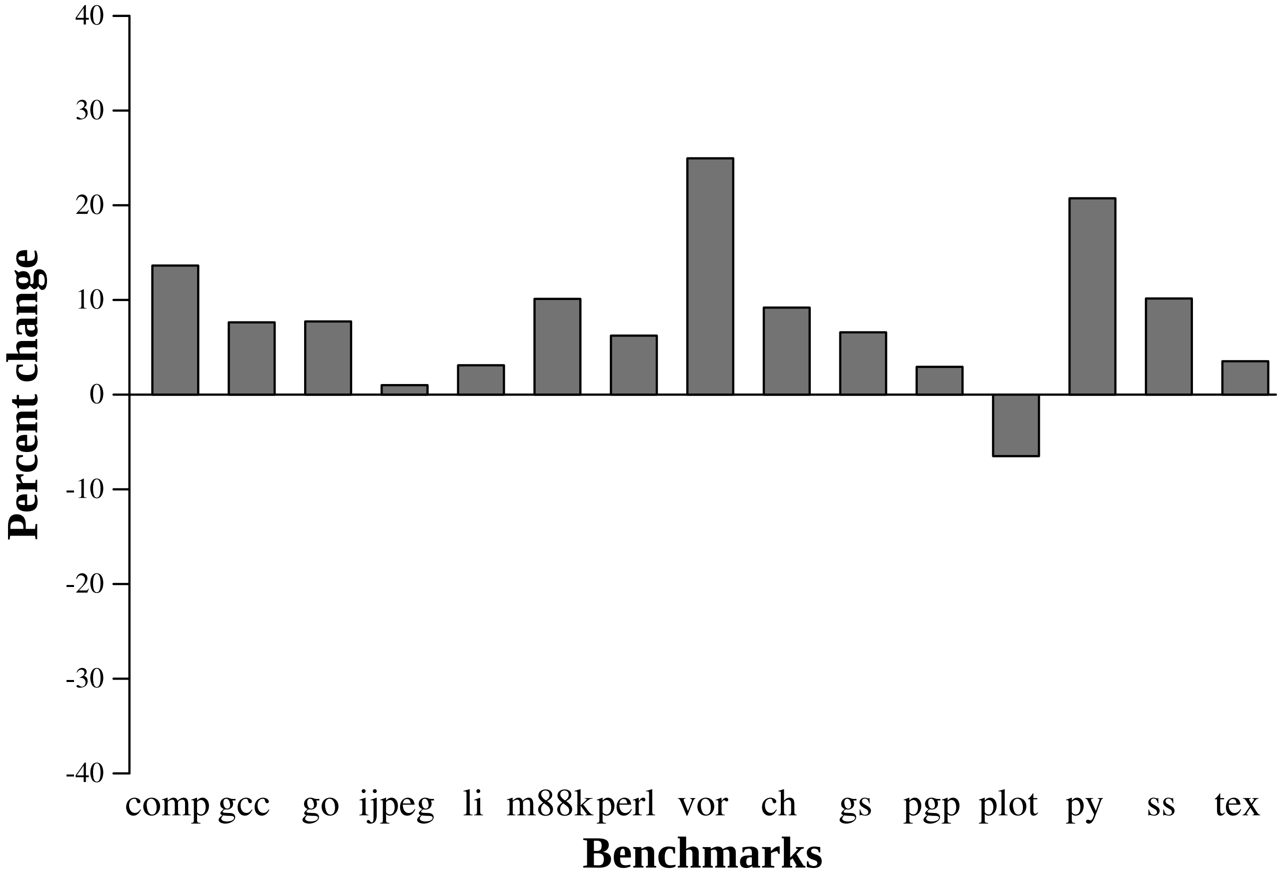
Effective Fetch Rate



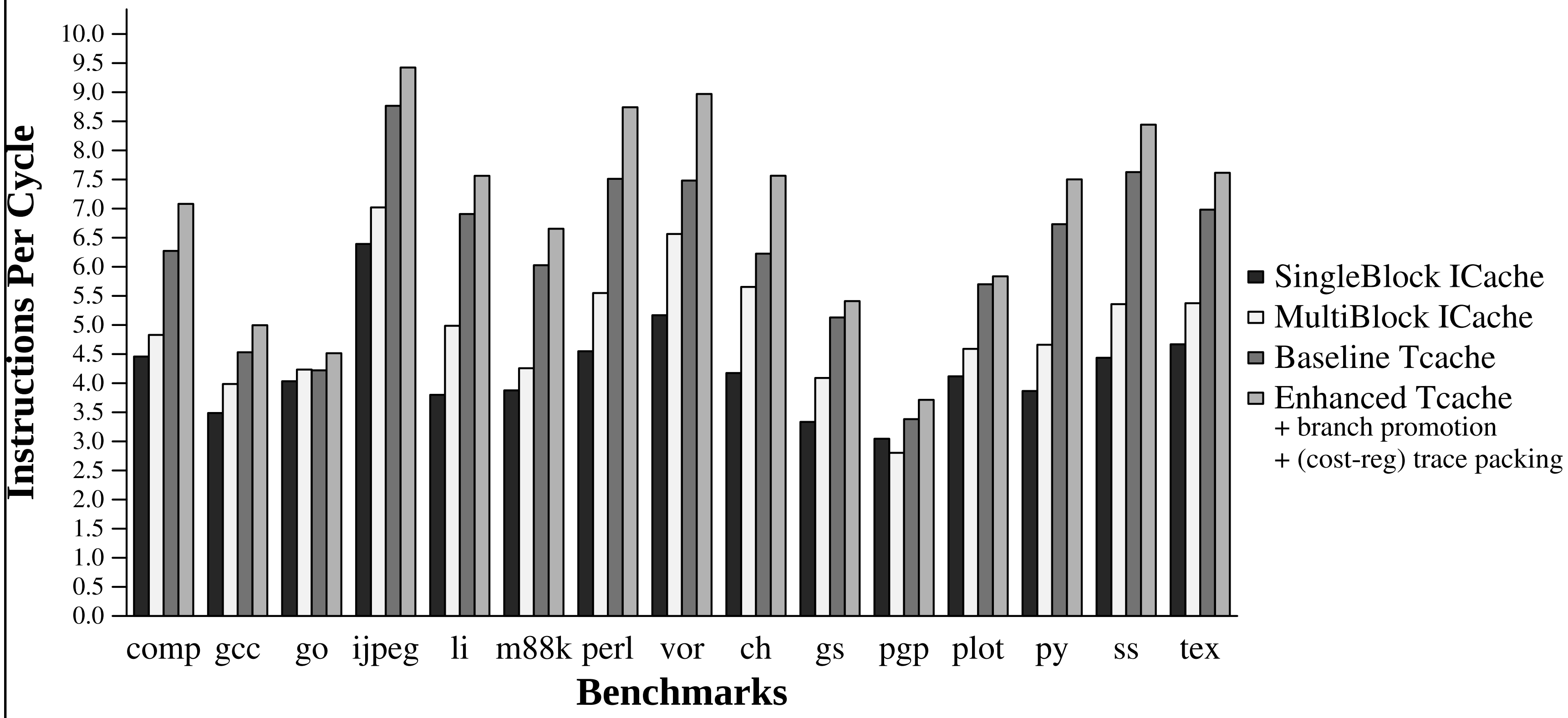
Percent Change in Conditional Branch Mispredictions



Percent Change in Average Branch Resolution Time



Overall Performance (aggressive core)



Conclusions

- Promotion reduces demands on branch predictor
- With Packing, enables high fetch rates
- Aggressive fetch demands agressive core
- 11% performance boost with both