Instruction Fetch using Multiple Sequencers

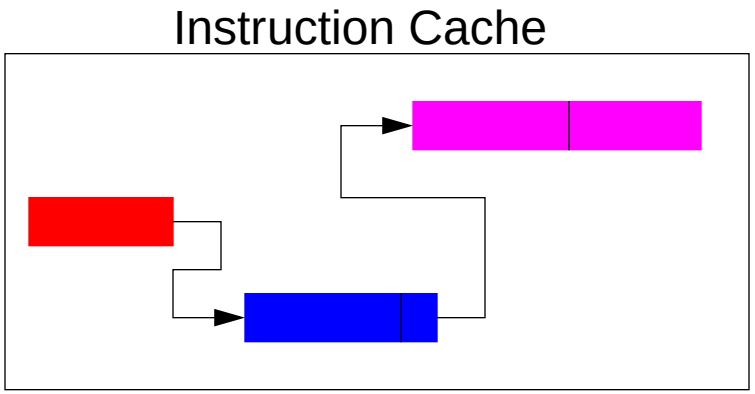
Paramjit Oberoi
Gurindar Sohi

University of Wisconsin - Madison

August 19, 2002

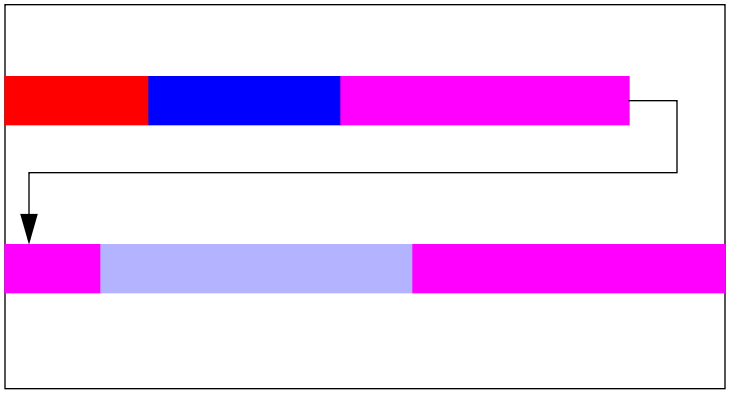
The 2002 International Conference on Parallel Processing
Vancouver, Canada

Problem: Instruction Stream Discontinuities

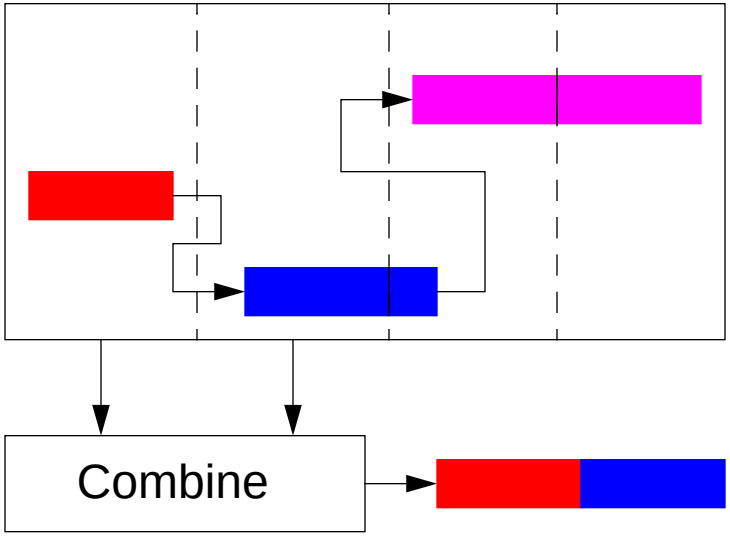


- Discontinuities in the instruction stream
  - Taken Branches
  - Cache Line Boundaries
- Discontinuities limit fetch throughput
- Solutions
  - Remove discontinuities
  - Handle discontinuities

Known Solutions



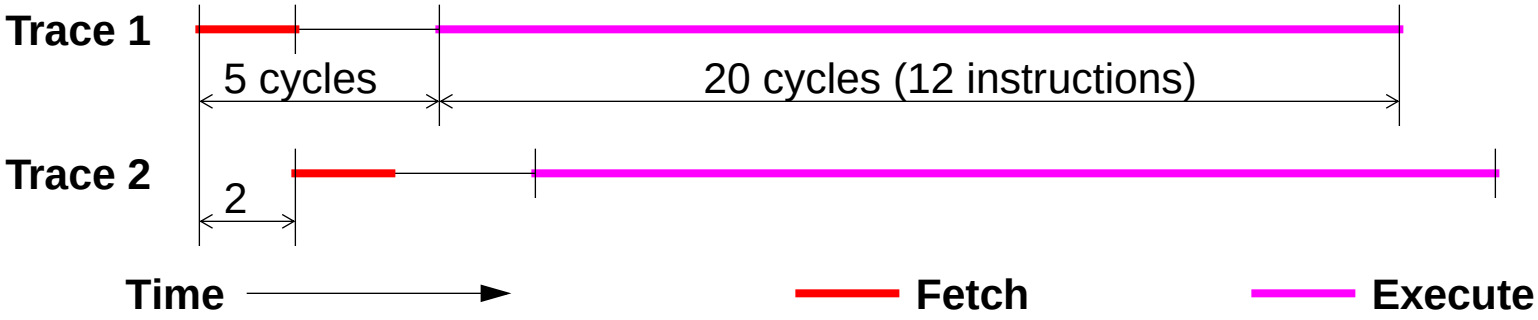
- **Remove Discontinuities**
  - Cache Layout
    Example: Trace Cache
  - Code Layout
    Example: Software TC, Dynamo



- **Handle Discontinuities**
  - Collapsing Buffer

Wide Fetch is Inefficient

- Example: Trace Cache



- Consecutive traces execute concurrently (almost)
  - Instructions are executed in dataflow order
  - But they are fetched serially
- A few critical instructions are needed first
  - Wide fetch also fetches the intervening instructions

Out-of-Order Fetch

- Instructions are not needed in sequential order
- Can we fetch instructions in a more efficient order?

Analogy:

Out-of-Order Execution	Out-of-Order Fetch
Uses execution resources more efficiently	Use fetch bandwidth more efficiently
Tolerates long latencies	Tolerate fetch delays
Higher execution throughput	Achieve high fetch throughput

Multiple Instruction Sequencers

- **Wide Fetch**
  - Fetch a large contiguous block of instructions

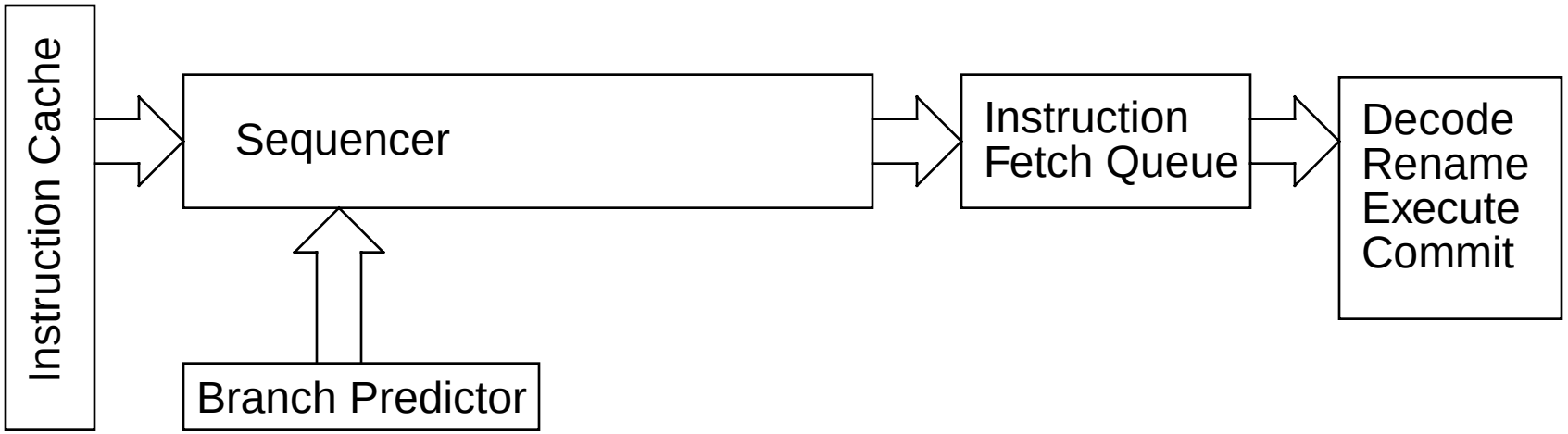
- **Multiple Fetch**
  - Fetch discontiguous blocks from multiple locations

Benefits of Multiple Sequencers

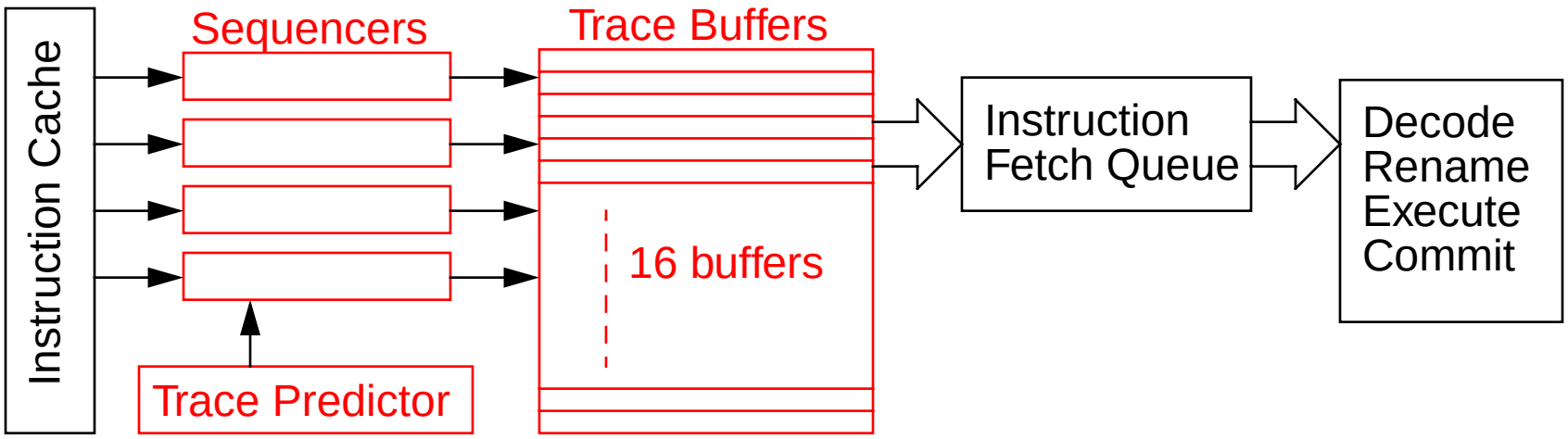
- Fetch throughput not limited by width of single sequencer
- Latency Tolerance
  - One cache miss does not block all sequencers
  - Overlap cache-miss latencies
- Flexibility
  - Not limited to sequential fetch
  - Fine-grain allocation of fetch bandwidth to threads
  - May ease implementation of many techniques
    - Dual-path execution
    - Speculative threads

Design Details

Single Sequencer

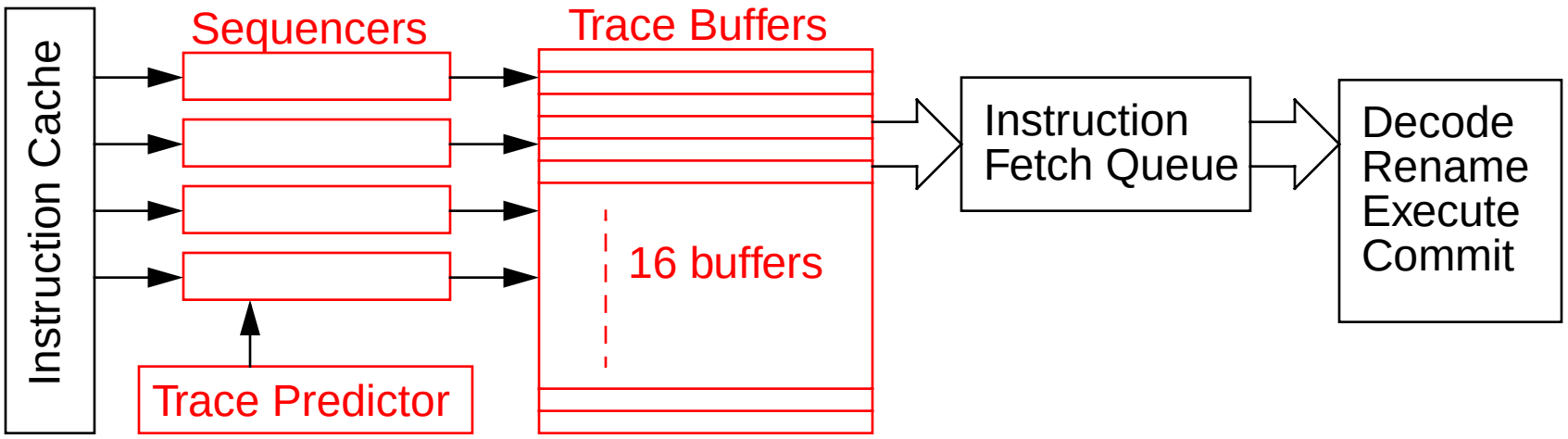


Multiple Sequencers



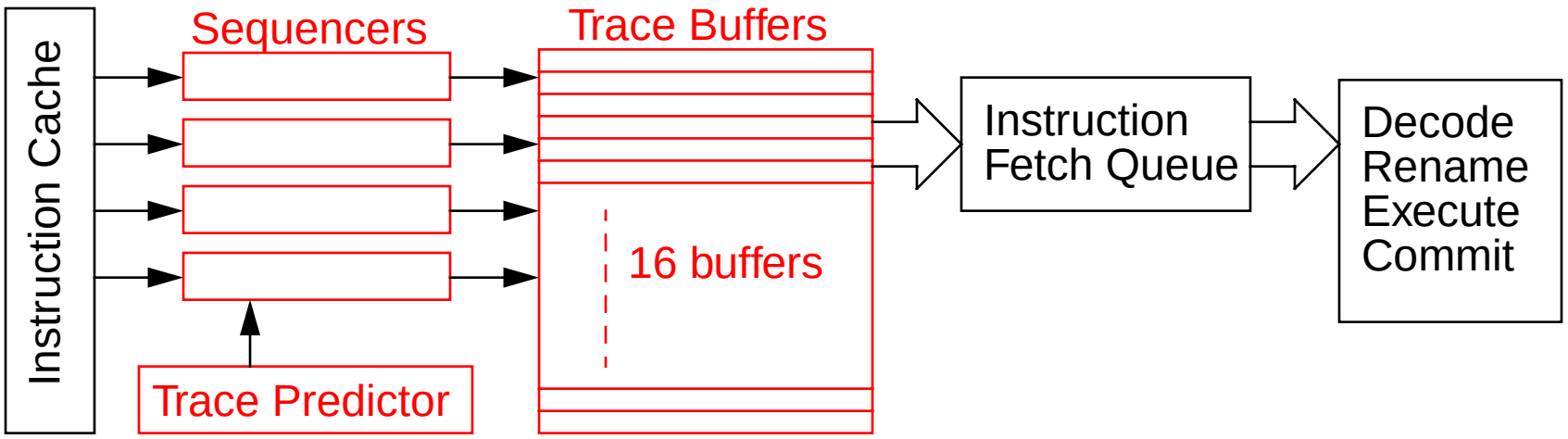
- Trace predictor predicts future traces
- Each sequencer is assigned a trace to fetch
- Multiple sequencers operate in parallel
  - I-Cache has multiple banks

Trace Buffers



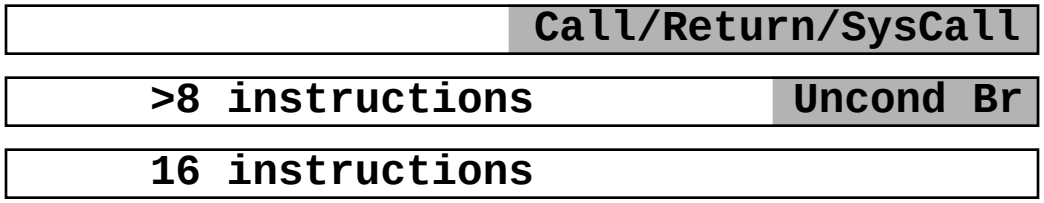
- Each sequencer can write to any trace buffer
- Instructions from oldest trace are placed in the IFQ
- Steady State: Just-in-time trace construction
  - Storage efficiency of Instruction Cache
  - Performance of Trace Cache

Trace Reuse



- Reuse traces instead of constructing them again
  - Like a small trace cache
- 20%-70% traces can be reused with only 16 trace buffers
  - Reduced I-cache traffic
  - Potential performance & power benefits

Trace Selection & Prediction



- Previously known techniques
  - Predictor: Breach [PhD Thesis], Jacobson et.al. [MICRO 1997]
  - 95% accuracy on average
- Trace selection heuristics
  - Maximum size is 16 instructions
  - End traces at Call/Return/SysCall
  - End traces at Unconditional Branchs if TraceSize > 8
    - Limit number of potential starting points - reduce working set
    - TraceSize > 8 increases average trace length

Trace Characteristics

Benchmark	Dynamic Instructions	Traces	Average Trace Size	Dynamic Traces	Traces Contributing 95% instructions
Integer					
bzip2	8822 M	1819	**12.79**	690 M	**109 (6%)**
crafty	4265 M	7541	**12.02**	355 M	**909 (12%)**
gap	1246 M	9074	**10.70**	117 M	**972 (11%)**
gcc	2016 M	38180	**11.26**	179 M	**7165 (19%)**
gzip	3367 M	1942	**12.06**	279 M	**58 (3%)**
mcf	260 M	1424	**9.84**	26 M	**132 (9%)**
parser	4203 M	6496	**10.35**	406 M	**692 (11%)**
Floating Point					
ammp	5491 M	2932	**13.11**	419 M	**332 (11%)**
equake	1443 M	2182	**11.10**	130 M	**356 (16%)**
lucas	3689 M	1090	**15.68**	235 M	**130 (7%)**
mesa	2845 M	2543	**11.30**	252 M	**110 (4%)**

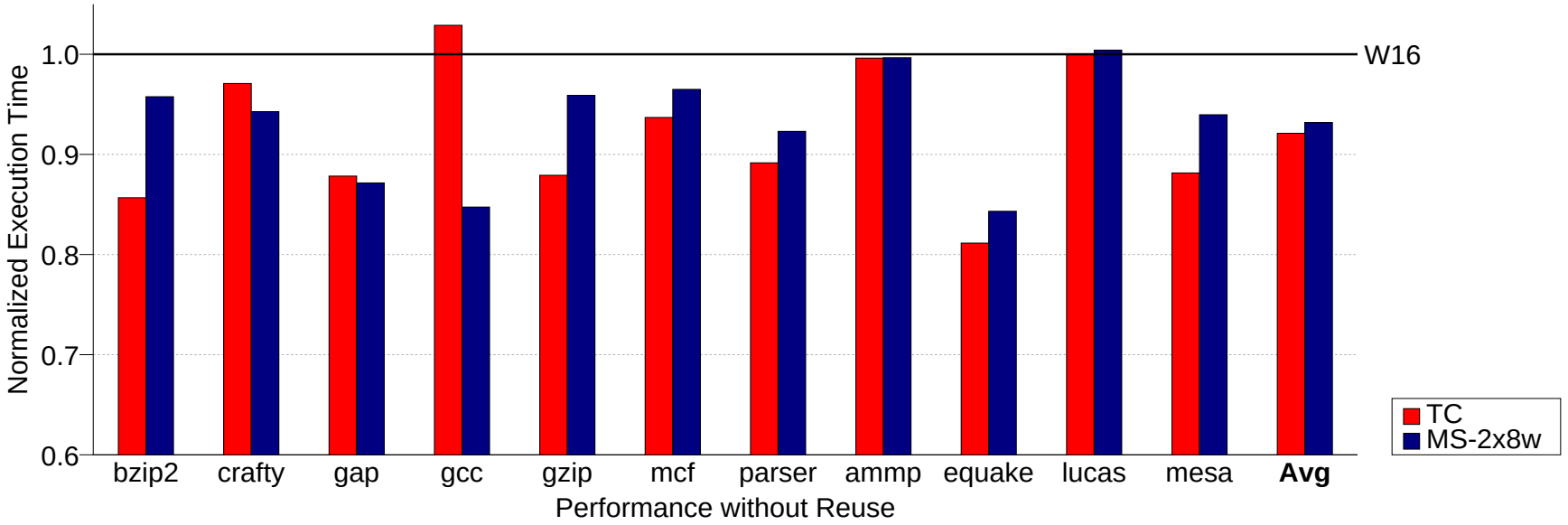
- Average Trace Size ~ 10-12 instructions
- Less than 1000 traces contribute to 95% dynamic instructions
  - except **gcc**

Performance Evaluation

Configurations

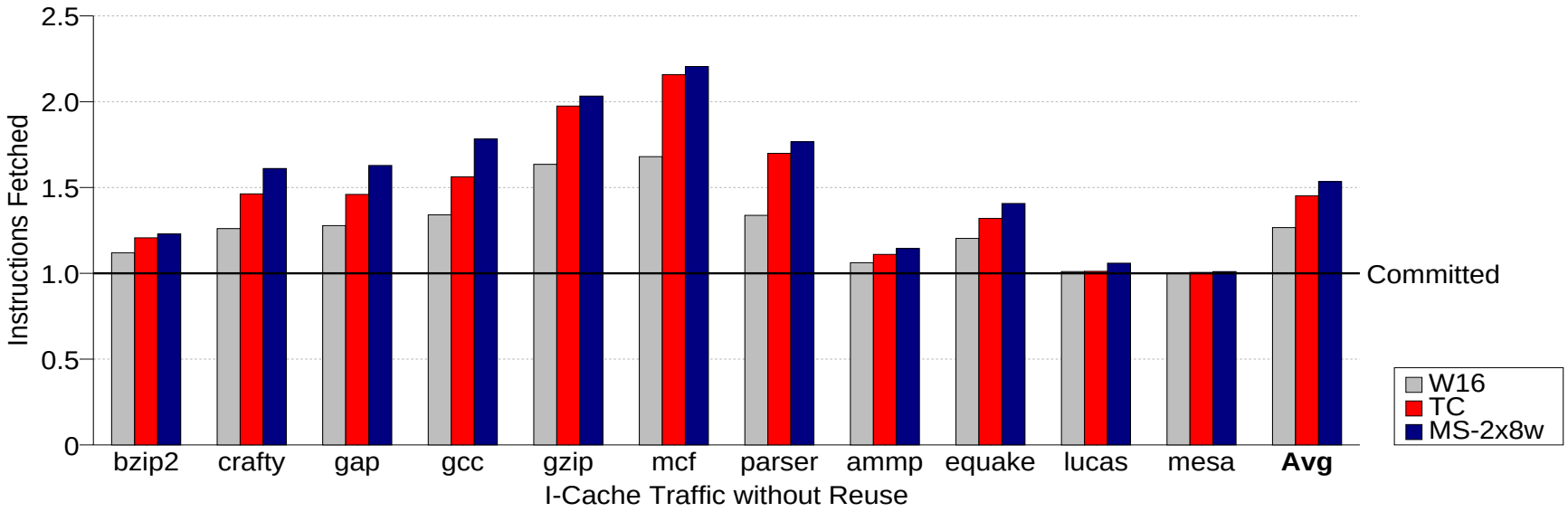
- 16-wide processor, 256 entry instruction window
  - 64K L1, 256K L2
- W16 — conventional 16-wide fetch
  - stop at taken branches and cache-line boundaries
- MS-2x8w — Multiple Sequencers
  - Two 8-wide sequencers
  - 16 trace buffers of 16 instructions each
- TC — Trace Cache
  - 2-way set associative, 16 instructions per trace
  - 32K (512 lines) + 32K I-cache
  - I-cache + TC performs better than just TC

Performance (without reuse)



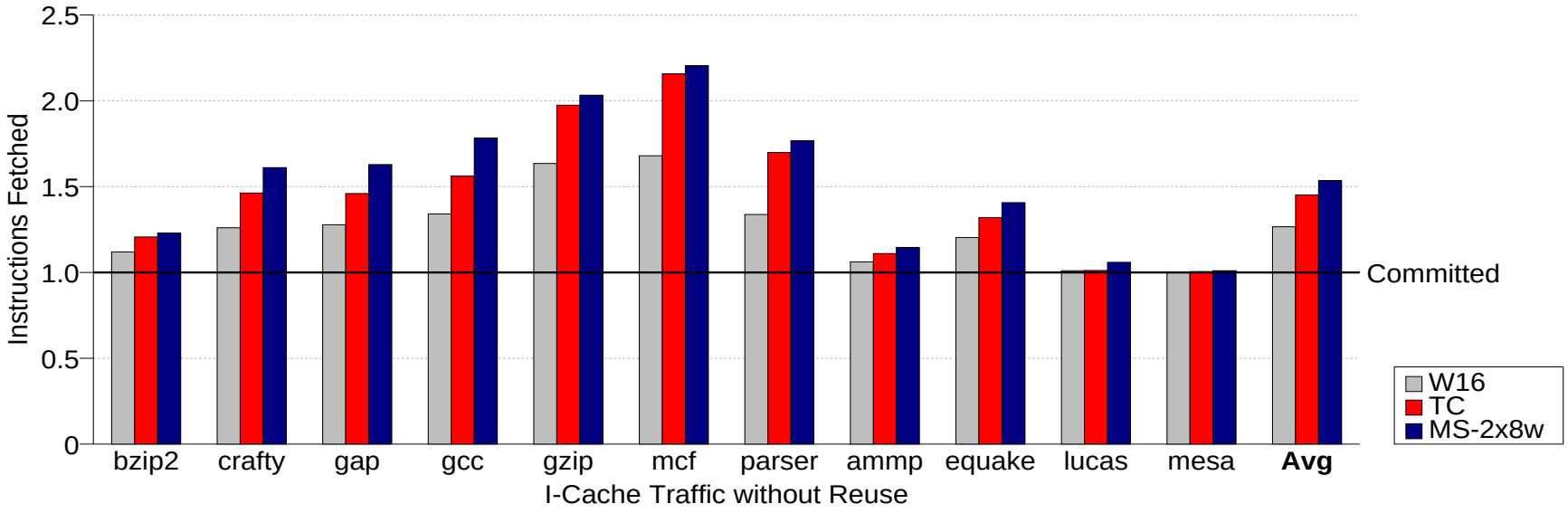
- 5% - 15% speedup over W16 on most benchmarks
- Performance similar to Trace Cache
- More efficient utilization of cache space (**gcc**)

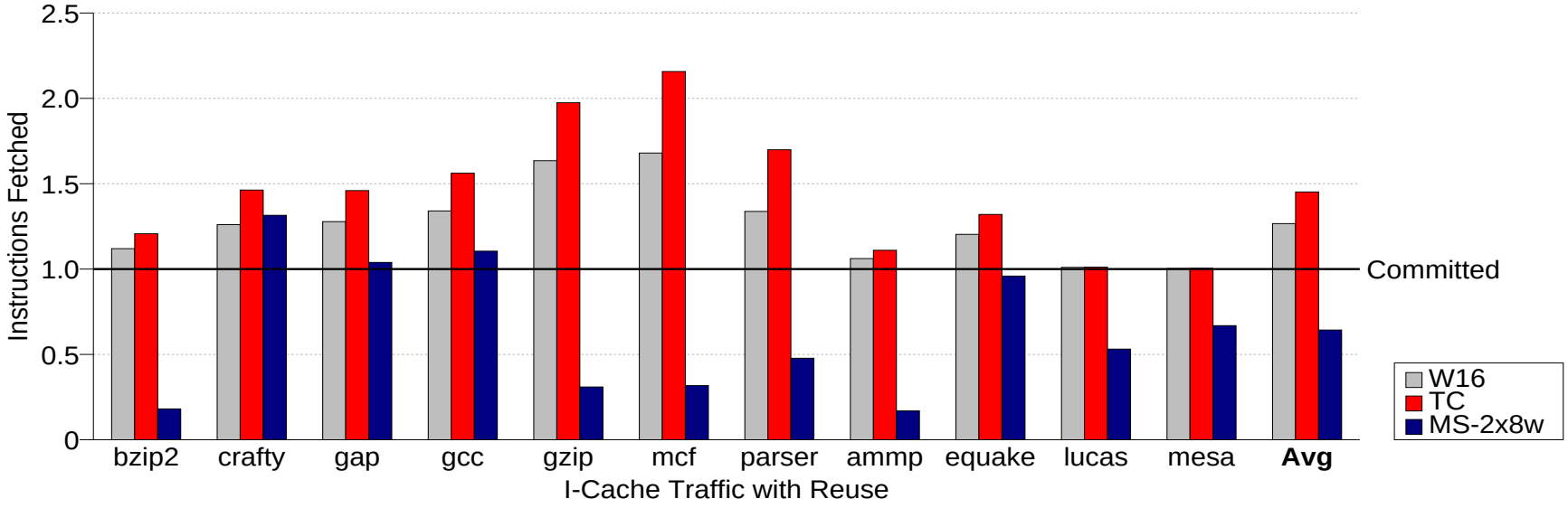
I-Cache Traffic (without reuse)



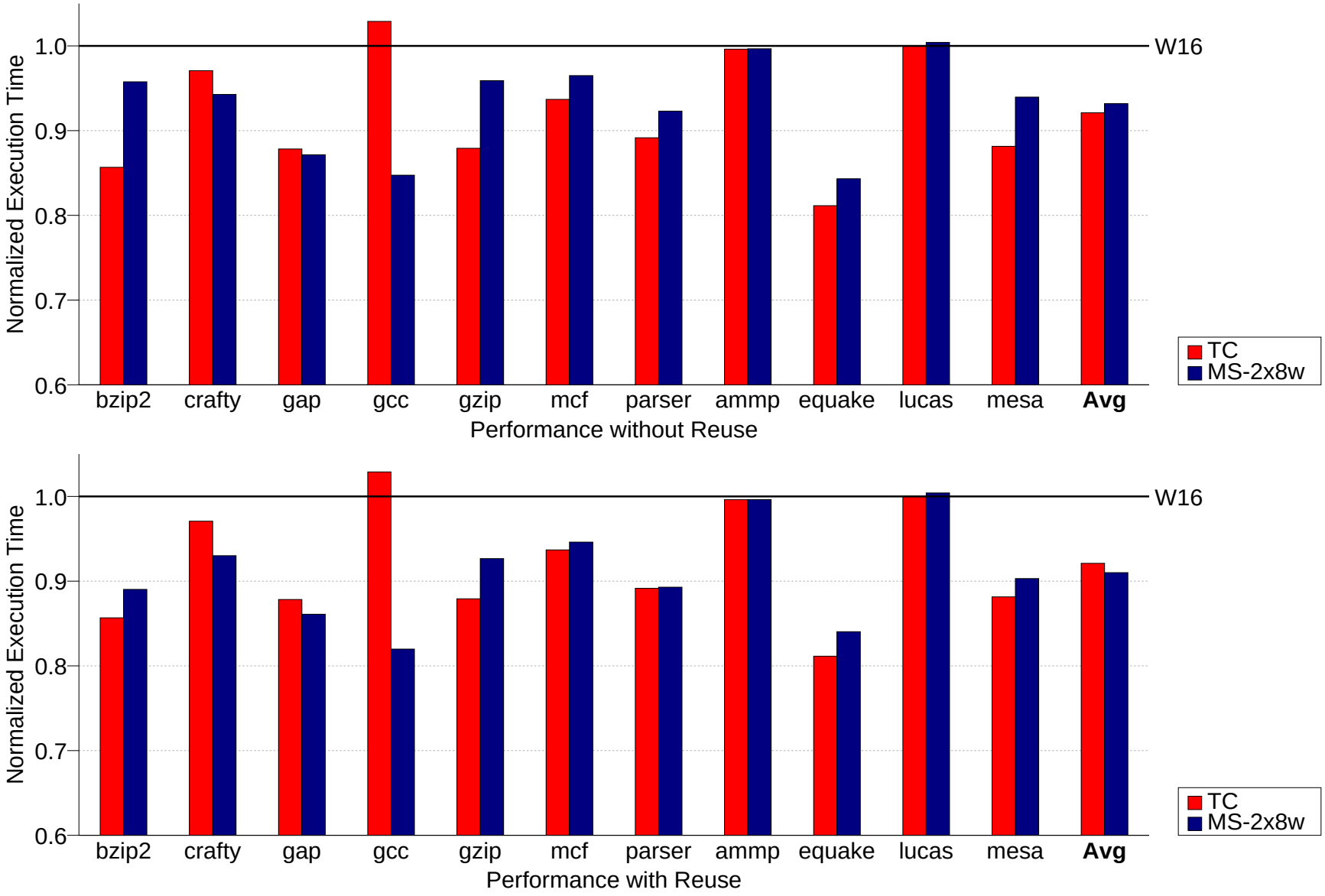
- Number of instructions fetched from the I-Cache
  - Independent of cache-line size
- 20% increase over W16

Effect of Reuse on I-Cache Traffic

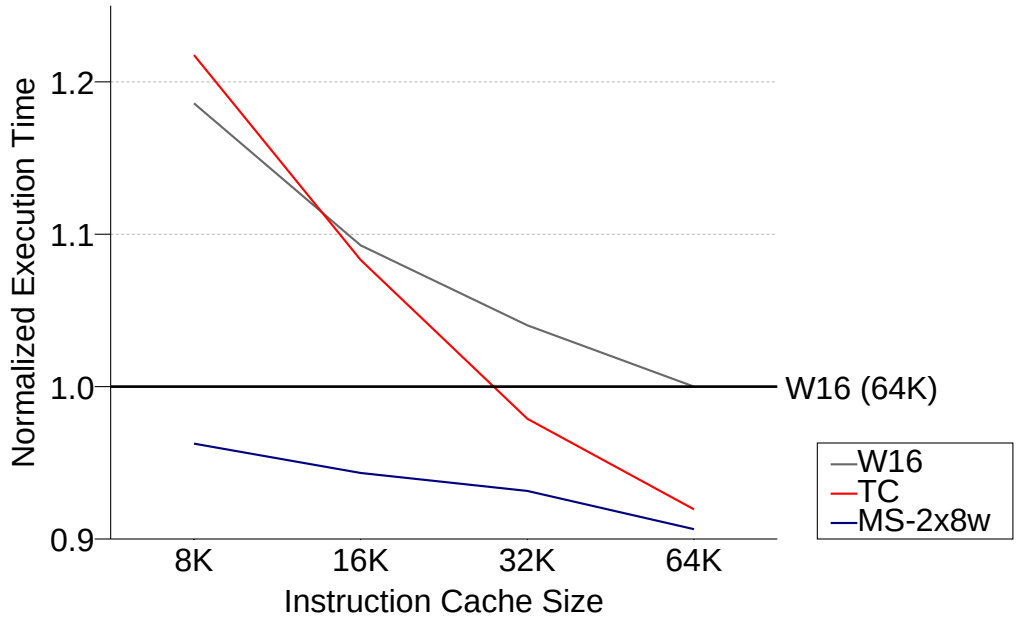




Effect of Reuse on Performance



Latency Tolerance

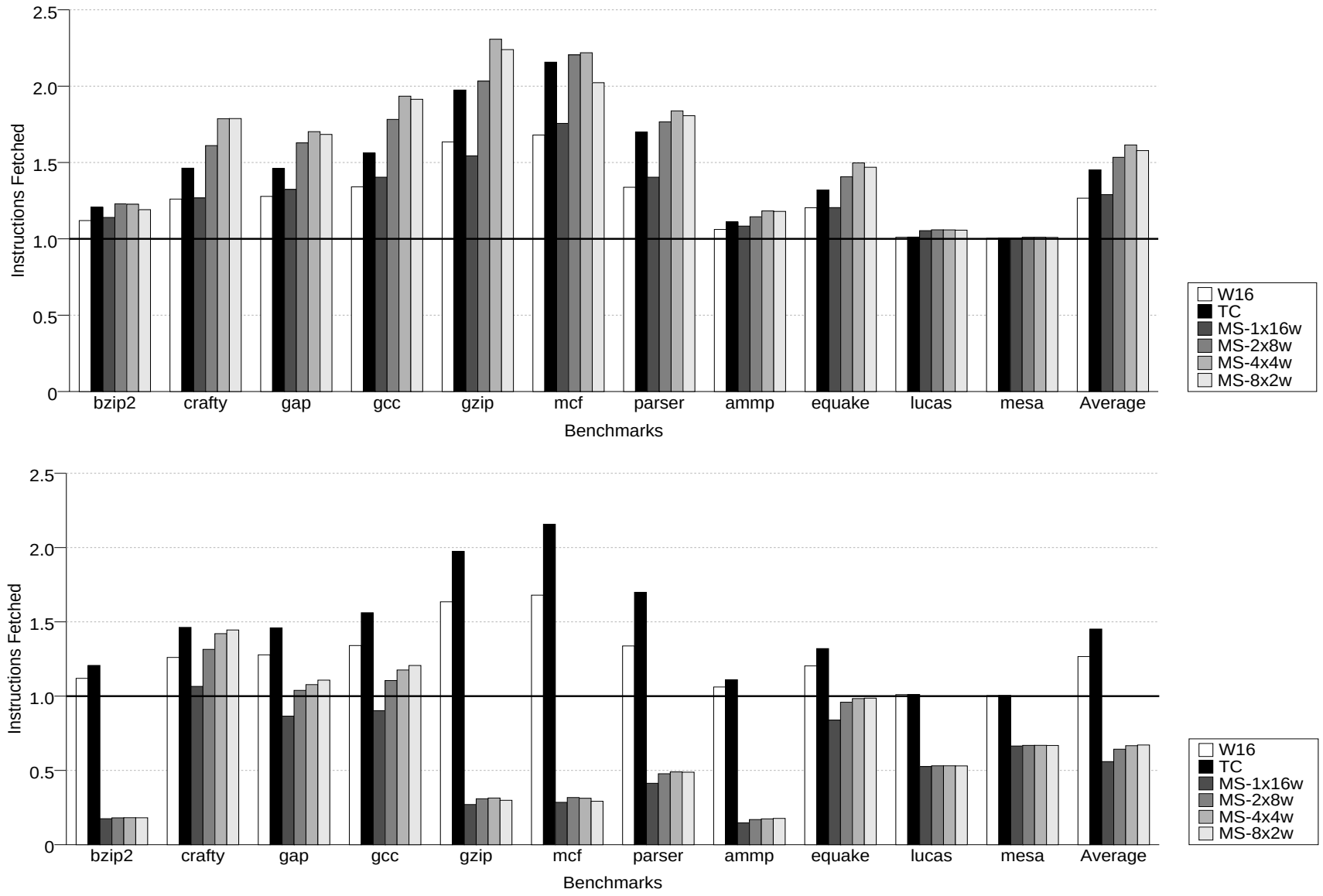


- Increase cache misses by reducing cache size
  - W16 and TC - performance deteriorates rapidly
  - MS performs robustly across a range of sizes
- MS is 20% faster on average for small cache sizes

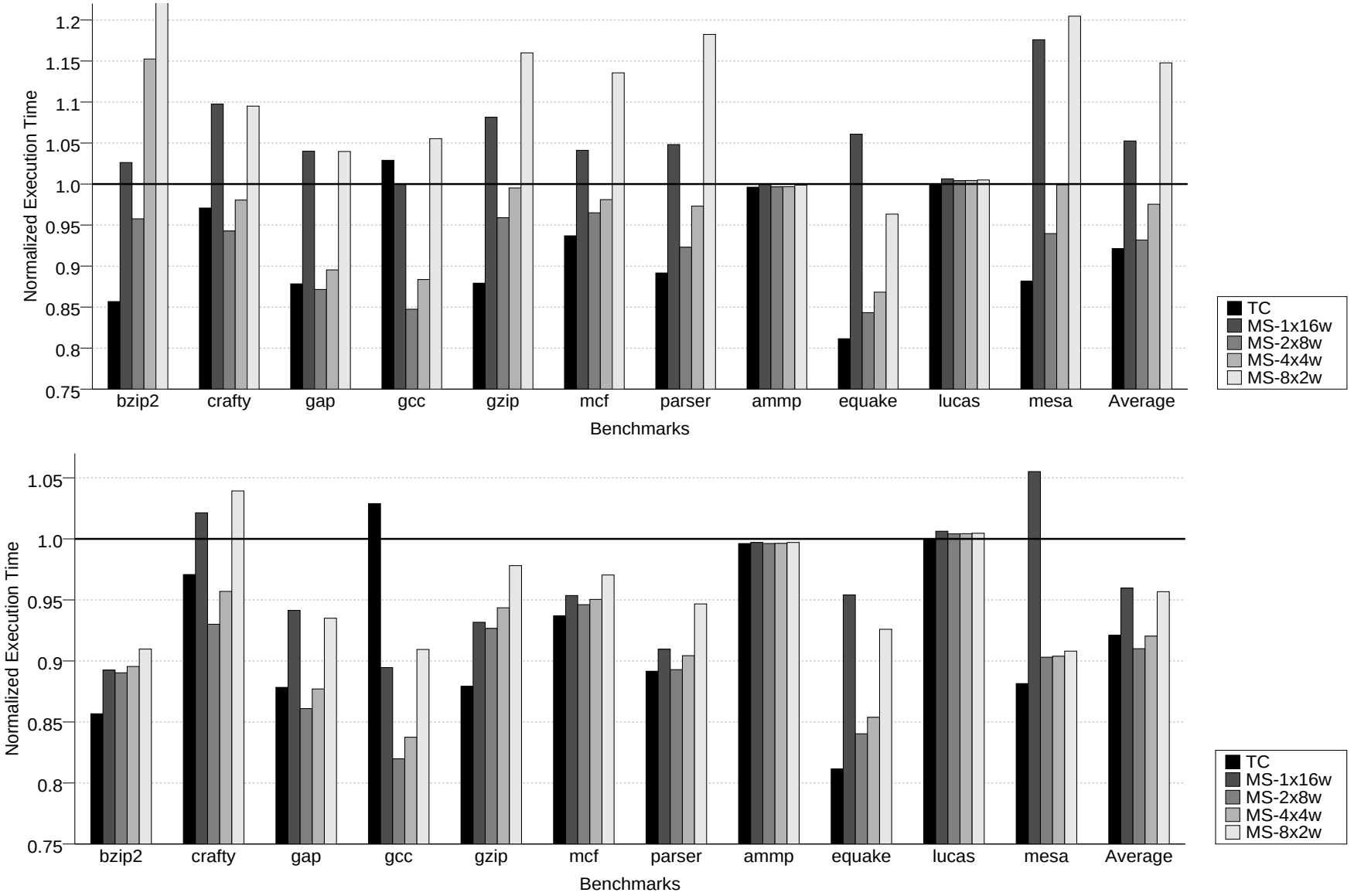
Conclusion

- High bandwidth fetch
  - Existing solutions attempt wide fetch
  - Multiple fetch is also an alternative
- Multiple Sequencers
  - Not limited by instruction stream discontinuties
  - Performance of Trace Cache, without wasted storage
  - More latency tolerant than existing techniques
- Better fit for the future
  - Flexible allocation of fetch resources to threads
  - Robust across a wide range of cache miss rates
    (Smaller caches, Large working sets)
  - May make other optimizations easier
    (Dual-path execution, Speculative threads)

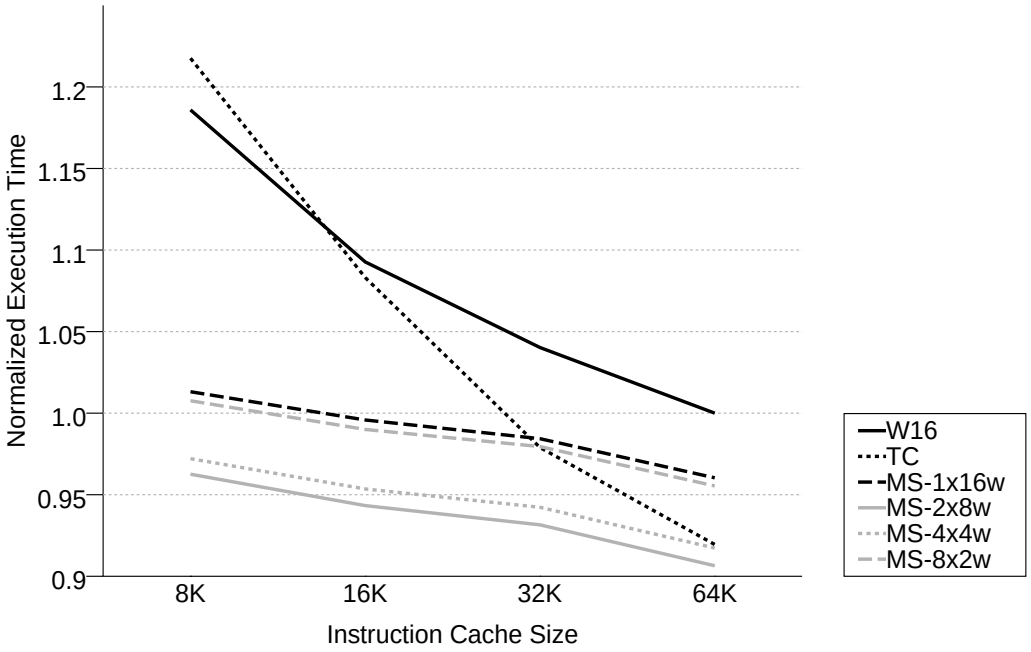
Backup: I-Cache Traffic



Backup: Performance



Backup: Latency Tolerance



- Increase cache misses by reducing cache size
  - W16 and TC - performance deteriorates rapidly
  - MS performs robustly across a range of sizes
- MS is 20% faster on average for small cache sizes

Backup: Simulation Parameters

Width	Fetch, decode and commit at most 16 instructions per cycle
Functional Units	16 integer ALUs, 4 integer multipliers, 4 floating point ALUs, 1 floating point multiplier, 4 load/store units
In-flight Instructions	256 entry instruction window 128 entry load/store queue
L1 Caches (Insn & Data)	64K, 2-way set-associative, 1 cycle access time, 64b blocks
L2 Cache (Unified)	256K, 4-way set-associative, 10 cycle access time, 128 byte blocks
Memory	100 cycle access time
Trace Predictor	64K entry primary table 16K entry secondary table D=9, O=4, L=7, C=9