Survey

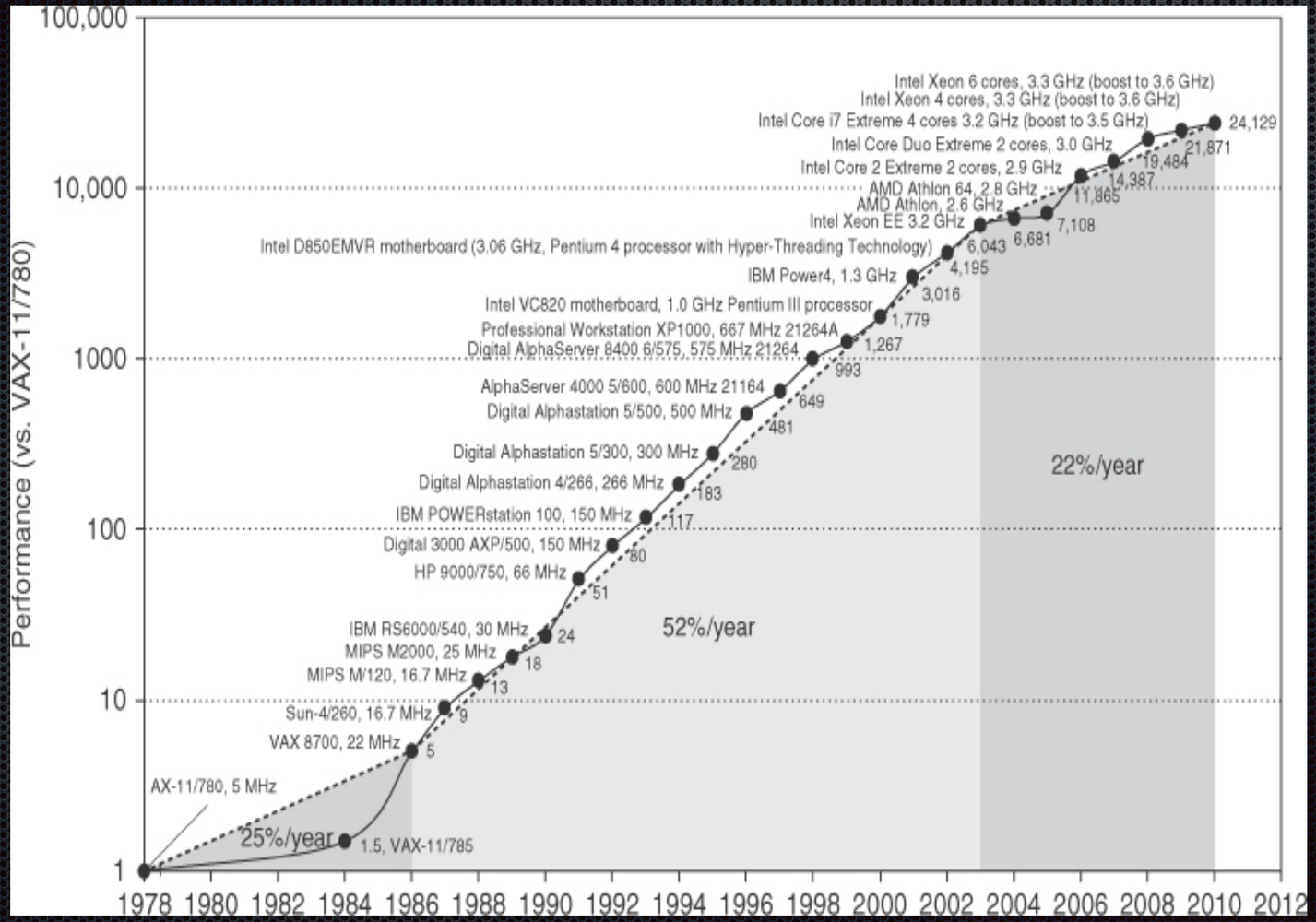
	Know well	Seen before	New
Pipeline speedup	2	16	6
Amdahl's law	1	10	13
Branch predictor	1	5	18
Victim buffer	3	8	13
RISC	6	9	9
Multicore	10	8	6
SMT	0	11	13

What is Architecture?

No longer a simple answer

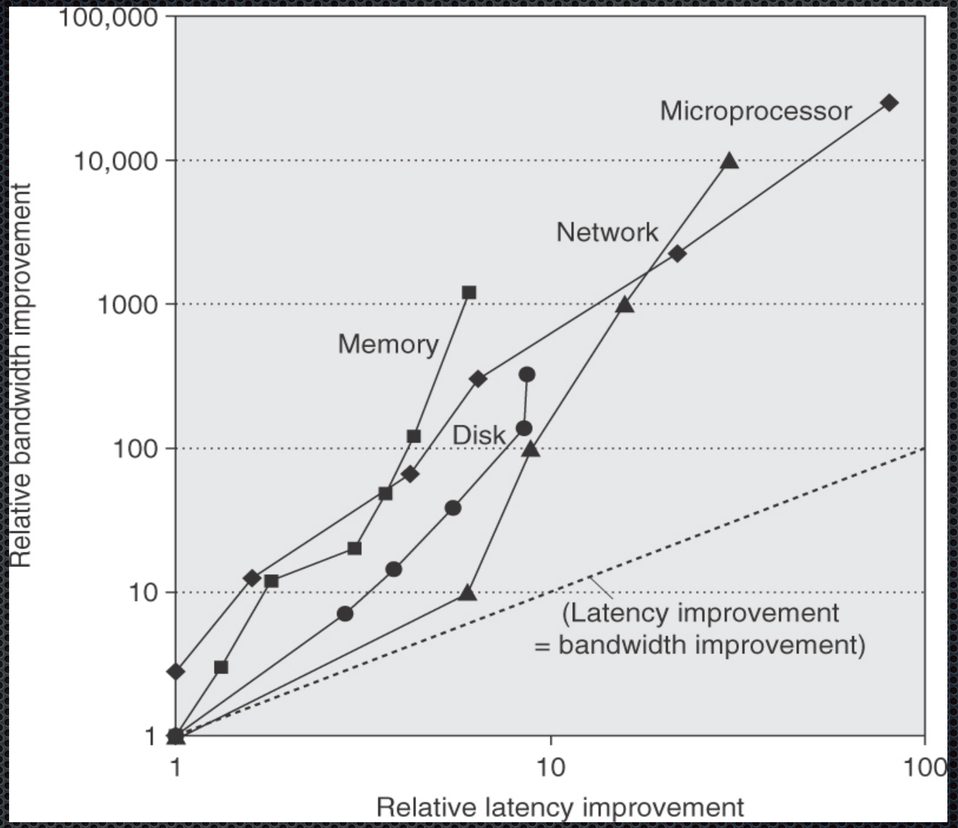
Performance Growth History

H&P Fig. 1-1



Bandwidth vs. Latency

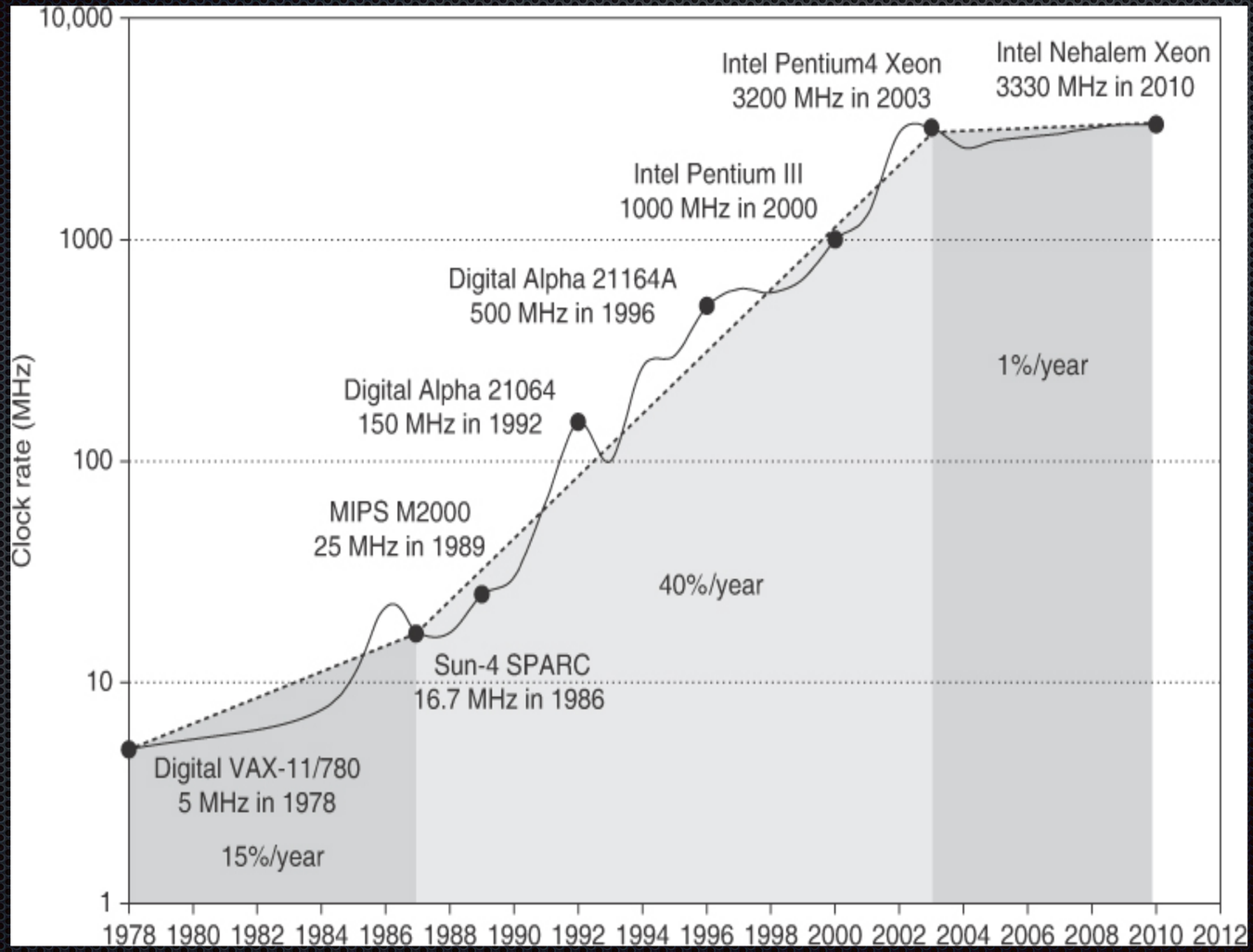
H&P Fig. 1-10



With each significant technology advance bandwidth has been easier to scale up

Clock Rate History

H&P Fig. 1-11



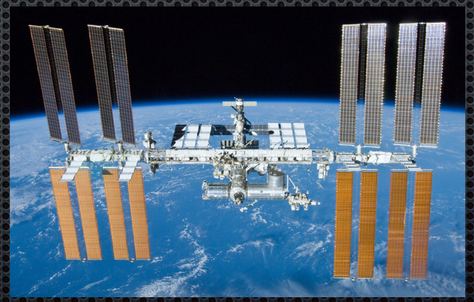
Architecture as Architecture







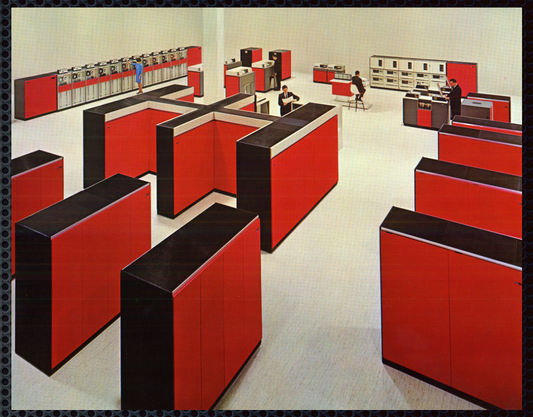


Instruction Set Architecture

- Assembly language view
- Execution contract
- Abstraction hiding a microarchitecture
- Abstraction hiding an abstraction
- Very little room today for innovation
  - Backwards compatibility

Microarchitecture



- Implementation of the abstraction
- Architecture family concept (IBM 360)
- Designed to meet goals
  - Cost, size, performance, power, thermal, reliability, lifetime, security, environment
- Targets future technology at start of design
    - Turning technology into a programmable computer

Performance Evaluation Methodology

Do you know what you're really measuring?

Methodology

- Computer science research is fast-moving
  - Driven by annual conference publication deadlines
- Sometimes cuts corners in methodology
- Architecture is no exception, and perhaps more so
- Why?

Nature of the Beast

- Architecture simulations are expensive
- Run thousands to millions of times slower
- Need to run big benchmarks for realism, significance
- Hard to validate (you never really know until you build it)
- Affected by all software tools (top of the food chain)

Example

- Early cache research used traces of about a million instructions
- Stone showed results were not significant
- 200K references, <10% miss rate => <20K misses
- For 1K line cache, average 20 events each, many lines are not even filled. Cache is still cold.
- Need 100 misses per line after warmup for significance

Proper Size

- 64K line cache (4MB, 64B/line)
- 100 misses per line = 6,400,000 misses
- 1% miss rate = 640,000,000 references
- 1/5 of instructions = 3.2B instructions (bare minimum)
- Given locality differences, could be 320B instructions to get significance
- x 10K slowdown = 3.2M sec at 1B IPS (~ 37 days)
- Obviously need to cut corners

Corners to Cut

- Use partial runs
- Use subsets of benchmarks
- Use traces instead of full simulation
- Use a simplified environment (faster simulation)
- Prewarm the system

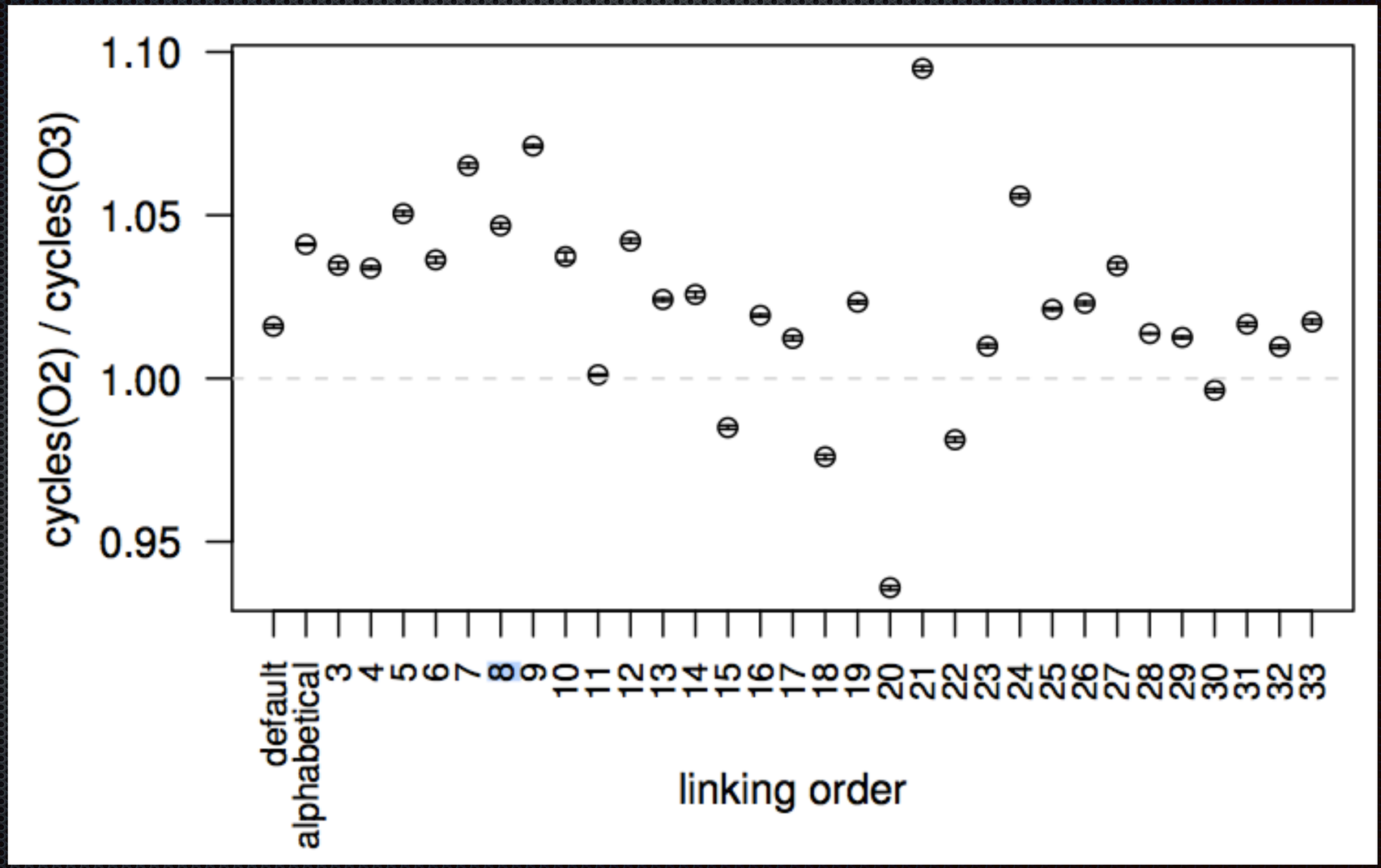
Diwan

- Measurement bias in systems research
- Uses SPEC 2006 C subset, both int and FP
- Compares O2 and O3 optimization
- Varies two factors:
  - Link order
  - Unix environment variable space

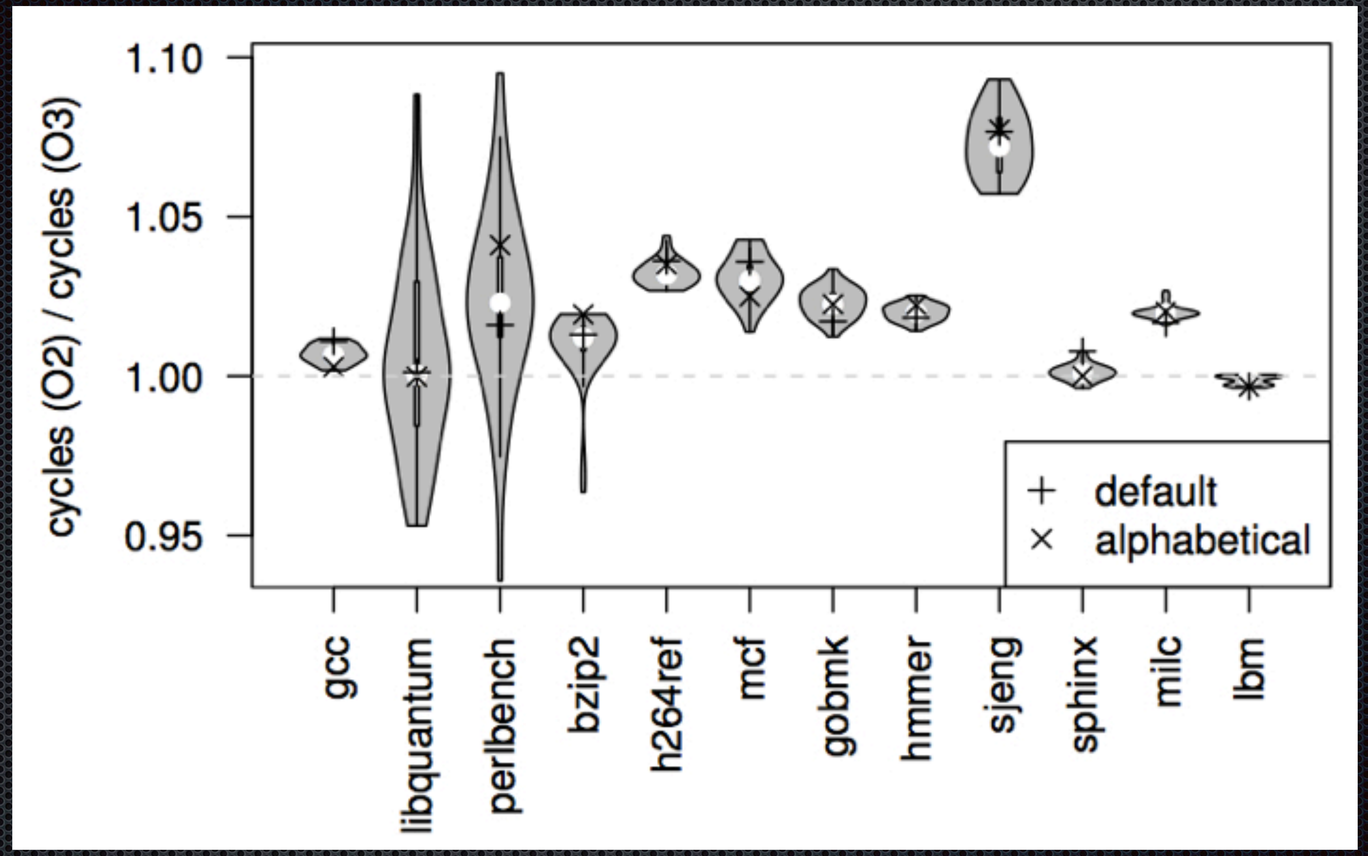
Benchmarks Used

Suite	Benchmark	Cycles	Seconds
CINT	gcc	3,142,640,332	1.3
	libquantum	7,690,324,238	3.2
	perlbench	12,752,930,486	5.3
	bzip	13,840,302,589	5.8
	h264ref	46,785,473,888	19.5
	mcf	59,250,579,566	25.0
	gobmk	180,491,870,344	75.2
	hmmer	246,974,548,791	102.9
	sjeng	419,892,937,630	175.0
CFP	sphinx3	35,238,100,682	14.7
	milc	57,414,647,507	23.9
	lbm	232,213,767,693	96.8

Effect of Link Order
(Perlbench only)

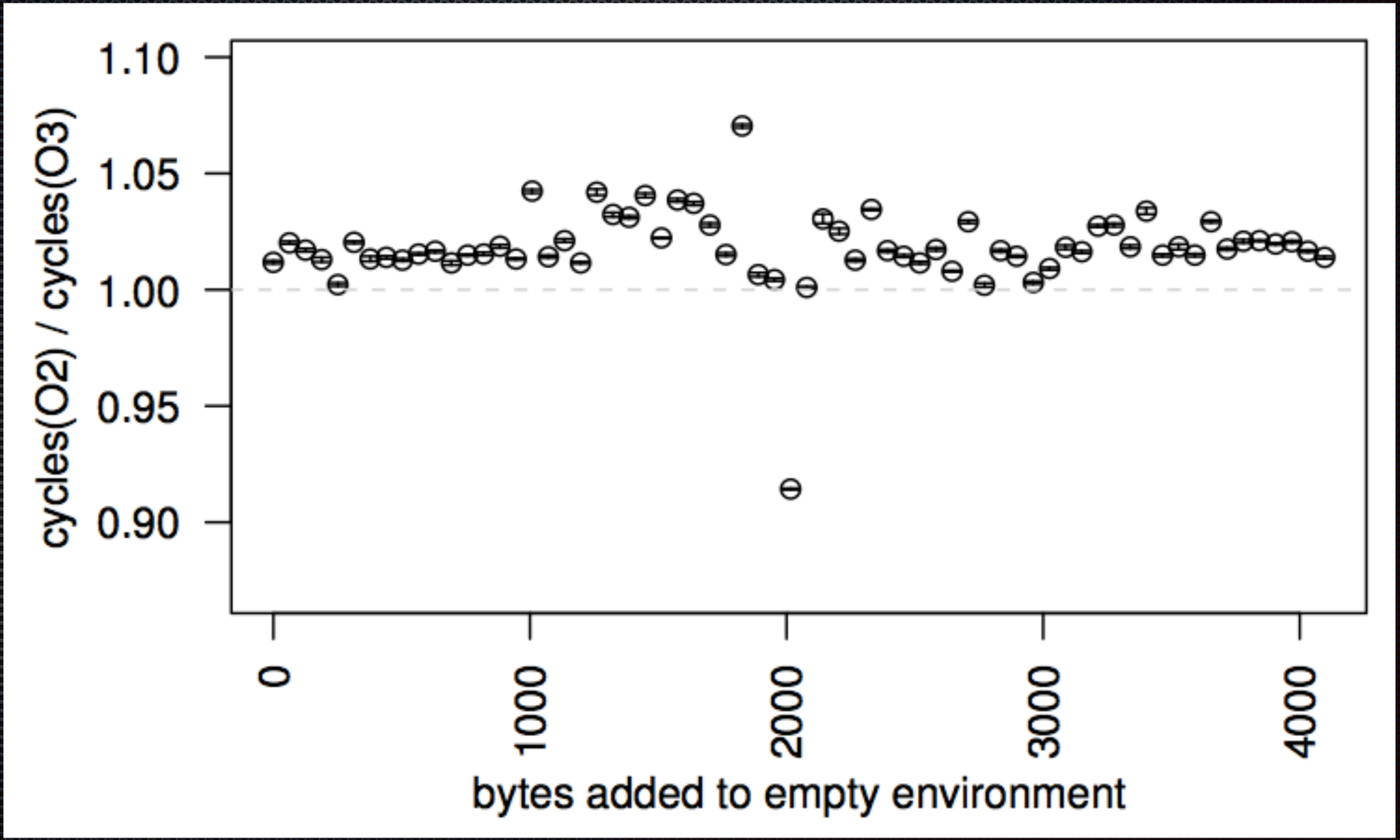


Effect of Link Order

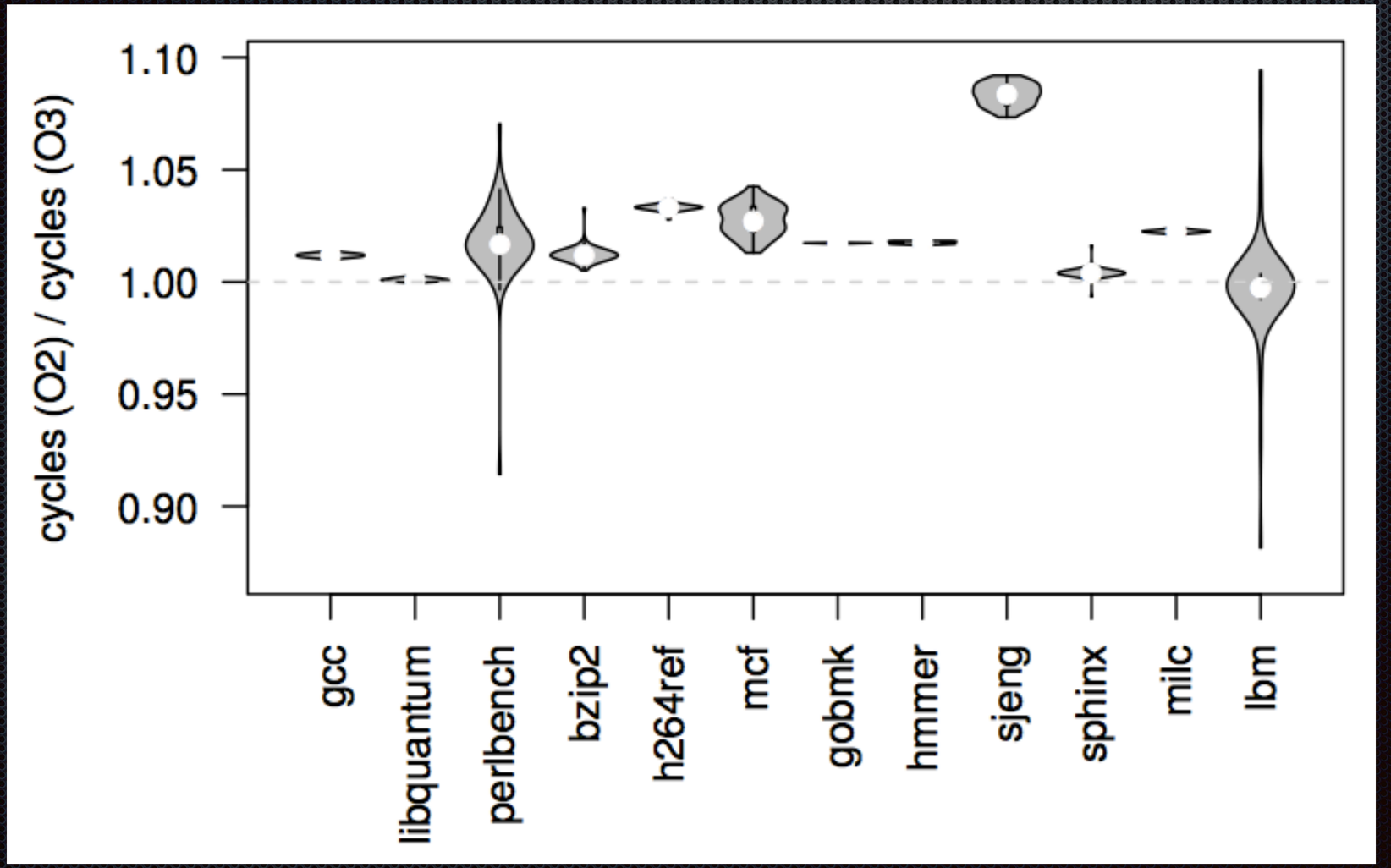


Width = #samples, dot = median, thick bar = interquartile range, + = default, X = alphabetical

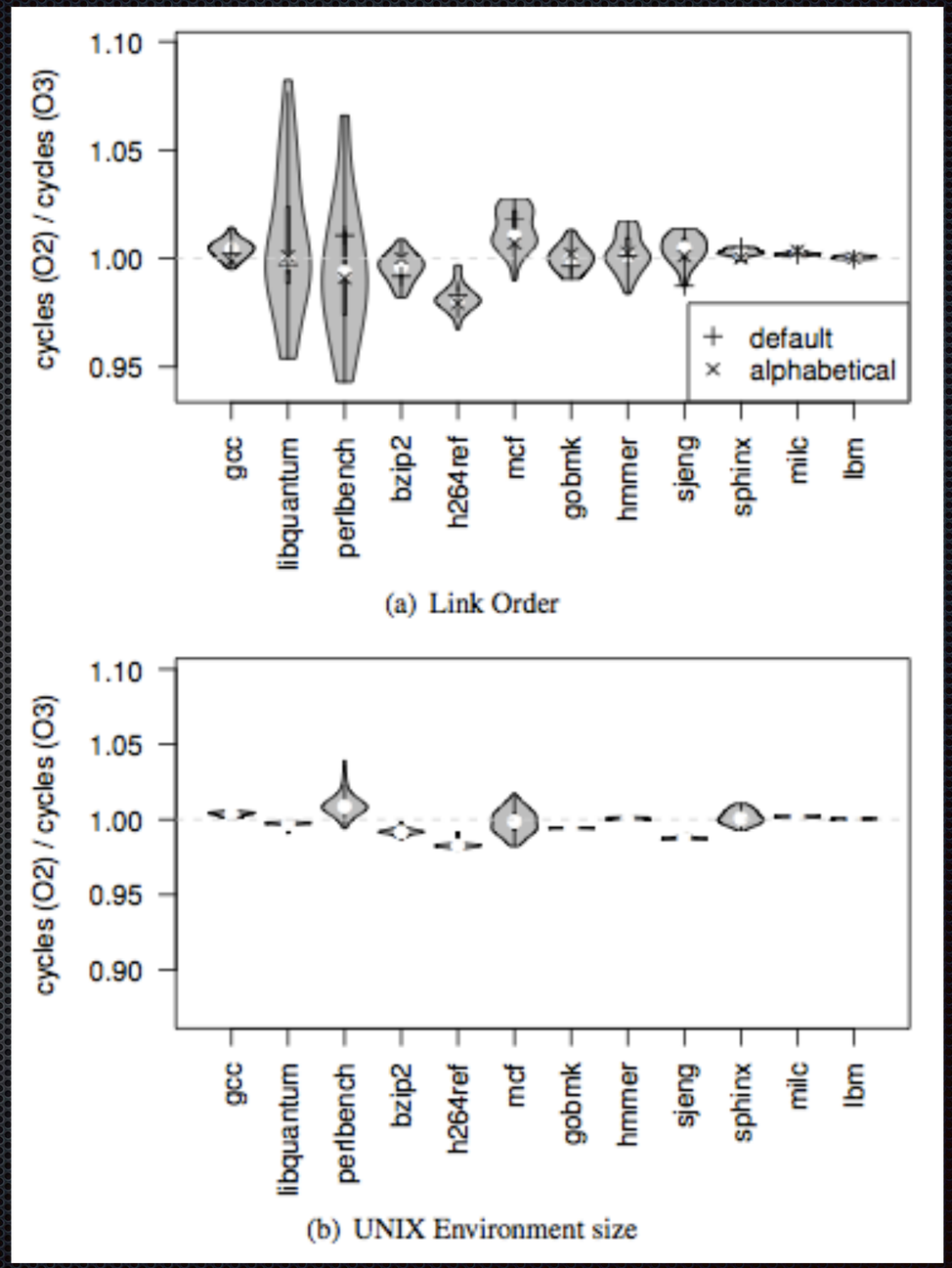
Effect of Environment
(Perlbench only)



Effect of Environment

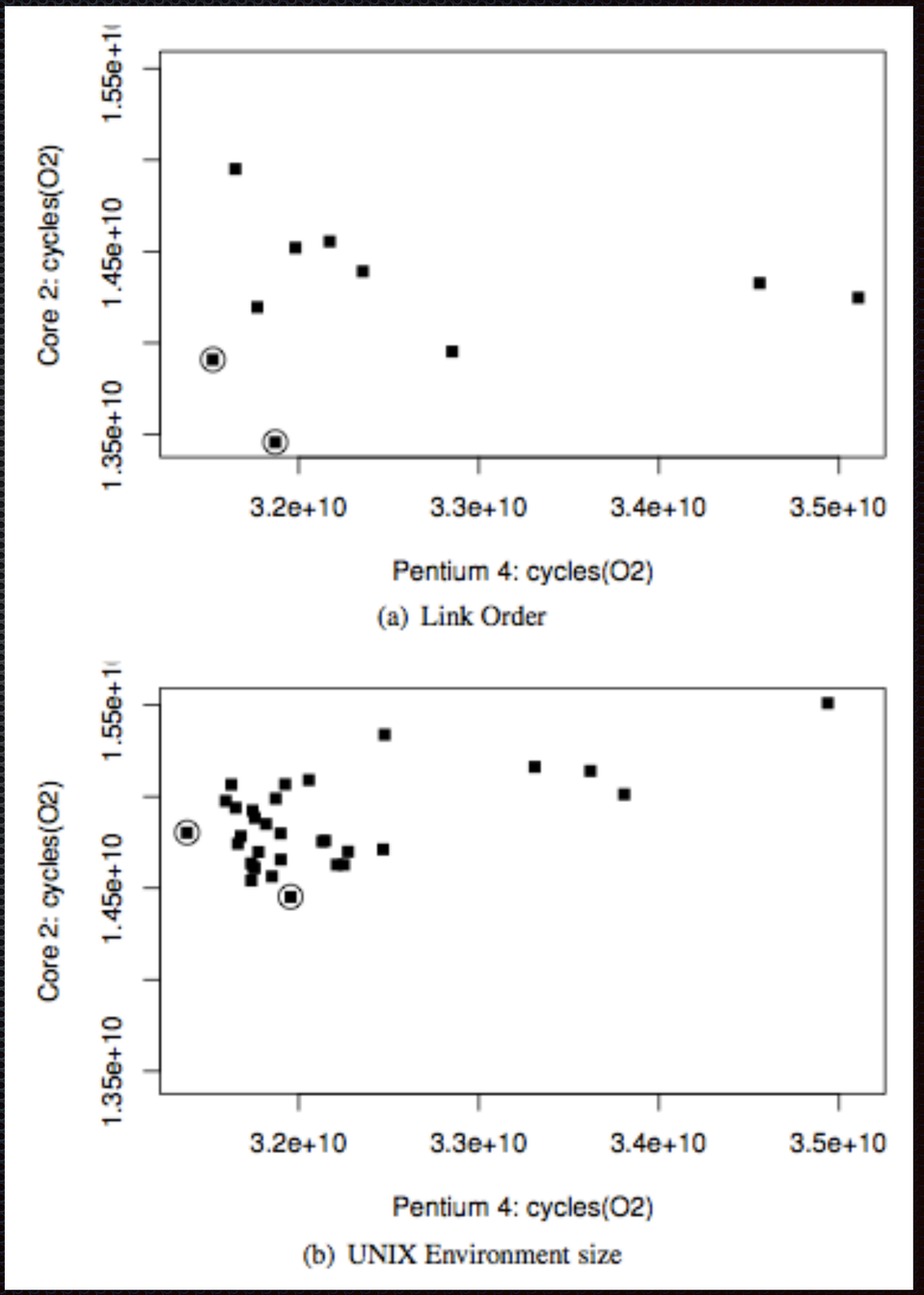


Intel Compiler



Core 2 vs. Pentium 4

Best for one doesn't predict for other



Literature

- ASPLOS, PACT, CGO, PLDI
- 88 of 133 include method (66%)
- Median reported speedup = 10%

Testing Diversity

7% variation for entire suite indicates that it is not diverse enough to eliminate bias

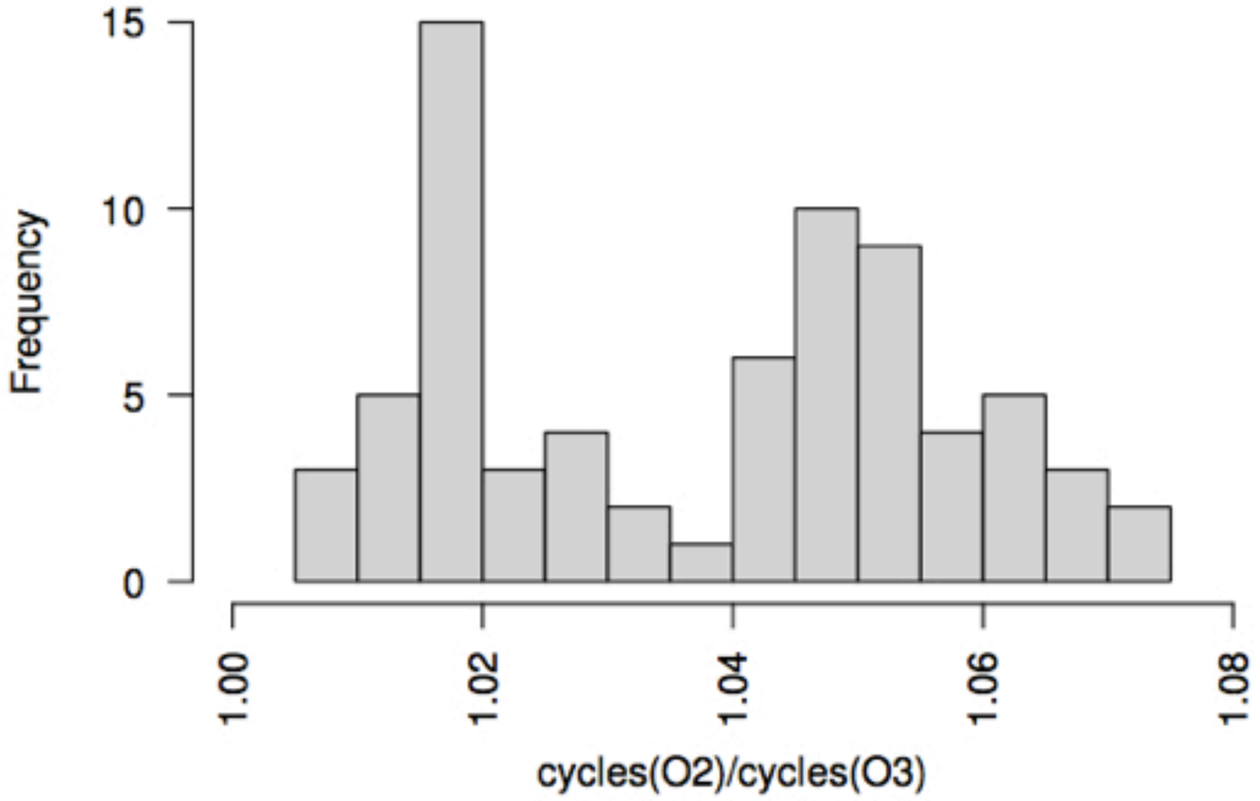


Figure 6. Distribution of speedup due to *O3* as we change the experimental setup.

Setup Randomization

(For variation in just the two factors)

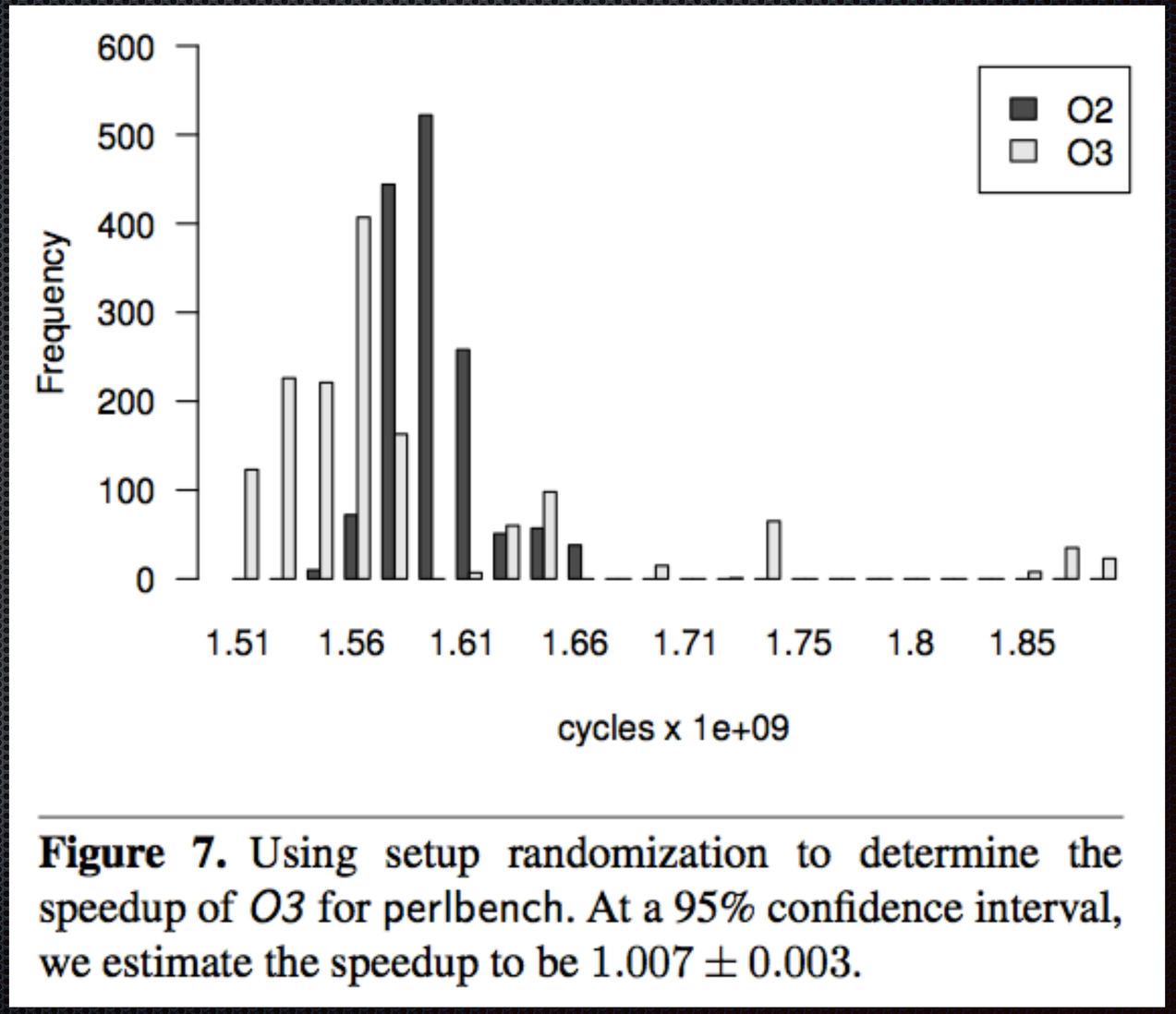


Figure 7. Using setup randomization to determine the speedup of *O3* for perlbench. At a 95% confidence interval, we estimate the speedup to be 1.007 ± 0.003.

Discussion