Benchmarking:
You're Doing It Wrong

DEVNEXUS™

Aysylu Greenberg

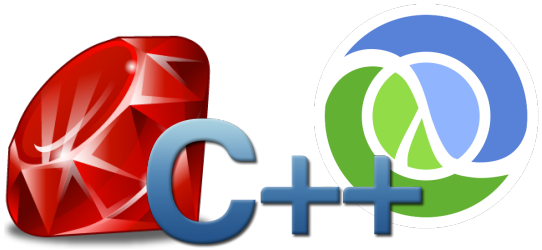
@aysylu22

Google

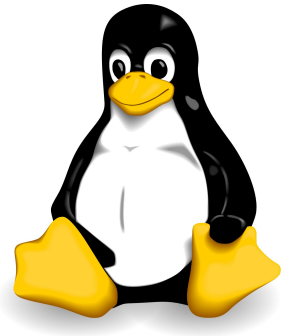
Loom

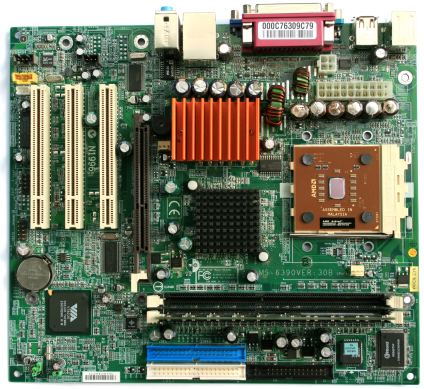

BENCHMARKING
YOU'RE DOING IT WRONG
imgflip.com

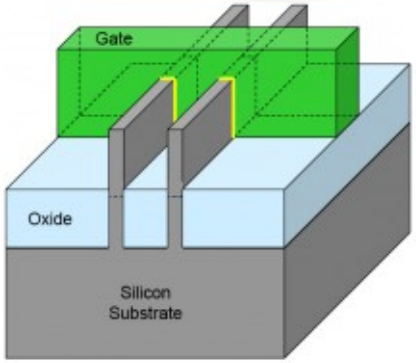
To Write Good Benchmarks…



Need to be Full Stack







Benchmark = How Fast?

your process vs goal

your process vs best practices

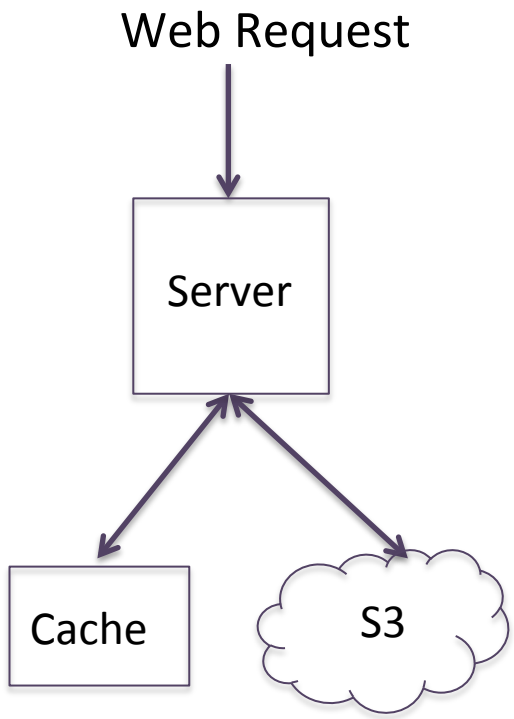
Today

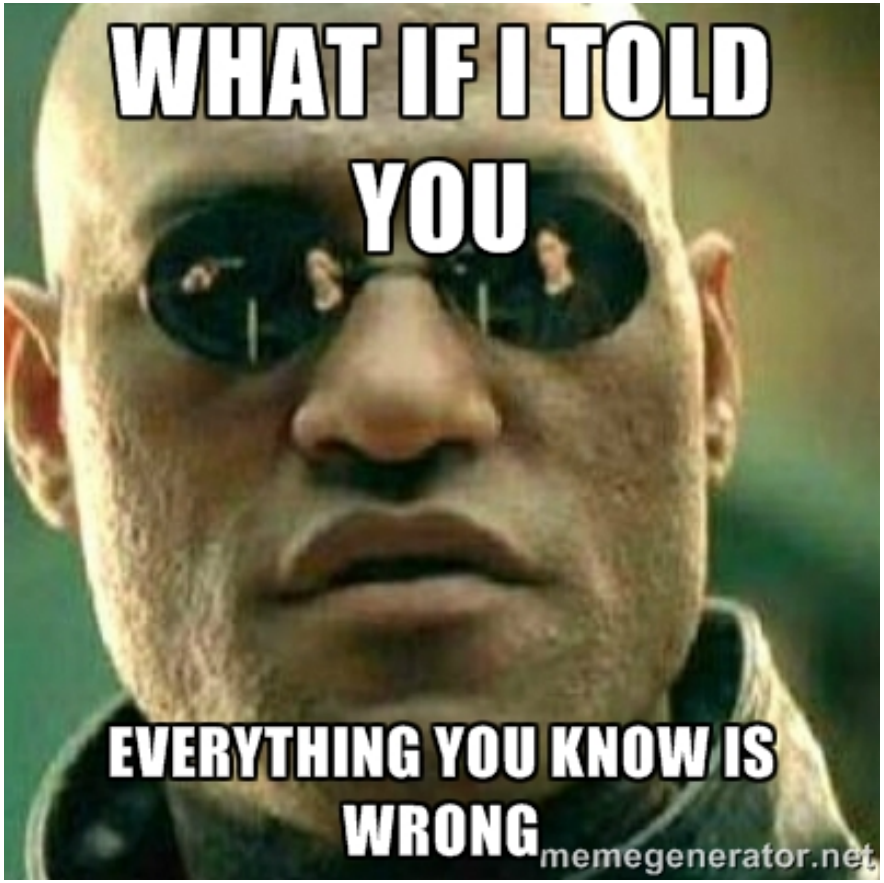
- How Not to Write Benchmarks
- Benchmark Setup & Results:
  - You're wrong about machines
  - You're wrong about stats
  - You're wrong about what matters
- Becoming Less Wrong

HOW NOT TO WRITE BENCHMARKS

Website Serving Images

- Access 1 image 1000 times
- Latency measured for each access
- Start measuring immediately
- 3 runs
- Find mean
- Dev environment



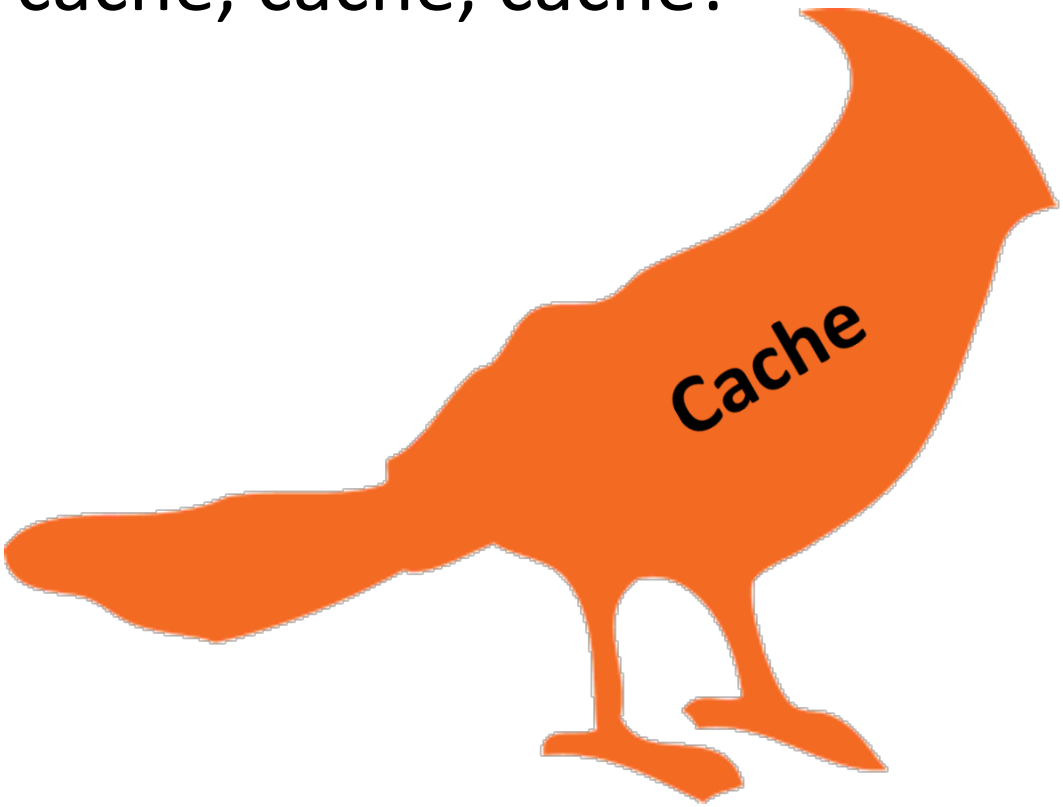


WHAT'S WRONG WITH THIS BENCHMARK?

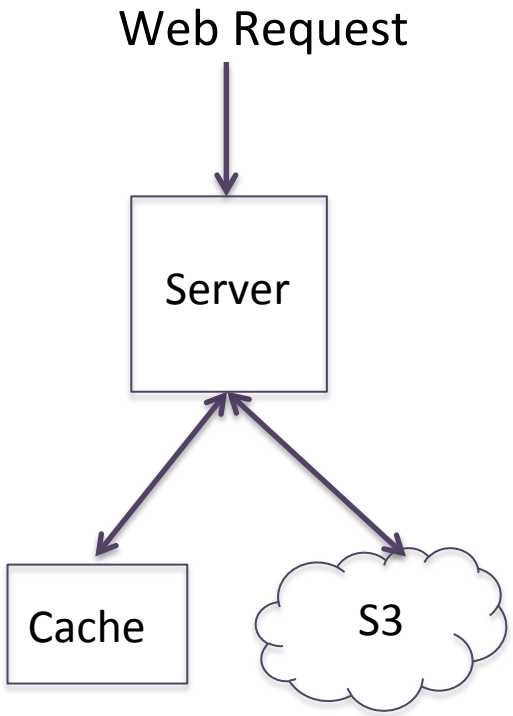
YOU'RE WRONG ABOUT THE MACHINE

Wrong About the Machine

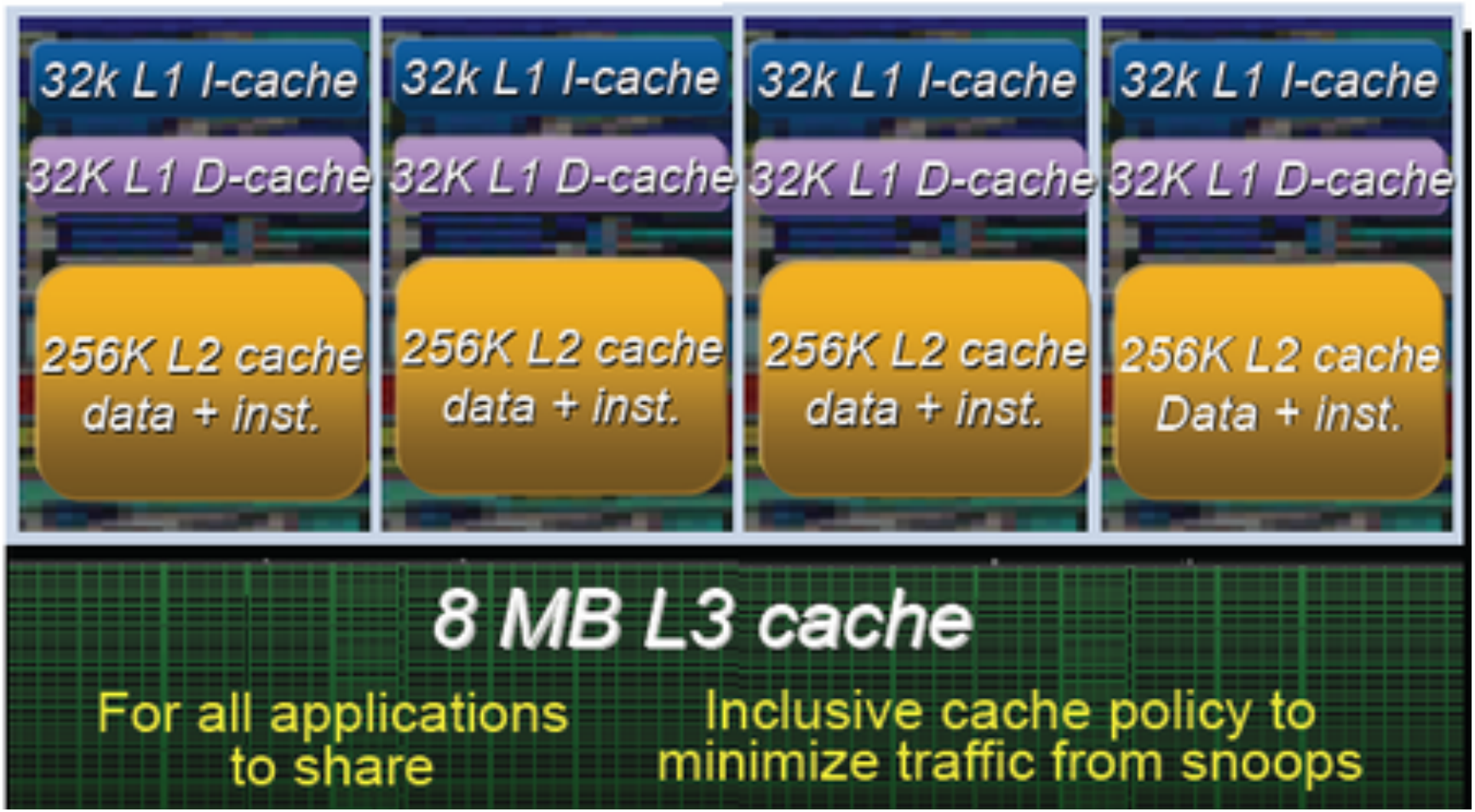
- Cache, cache, cache, cache!



It’s Caches All The Way Down



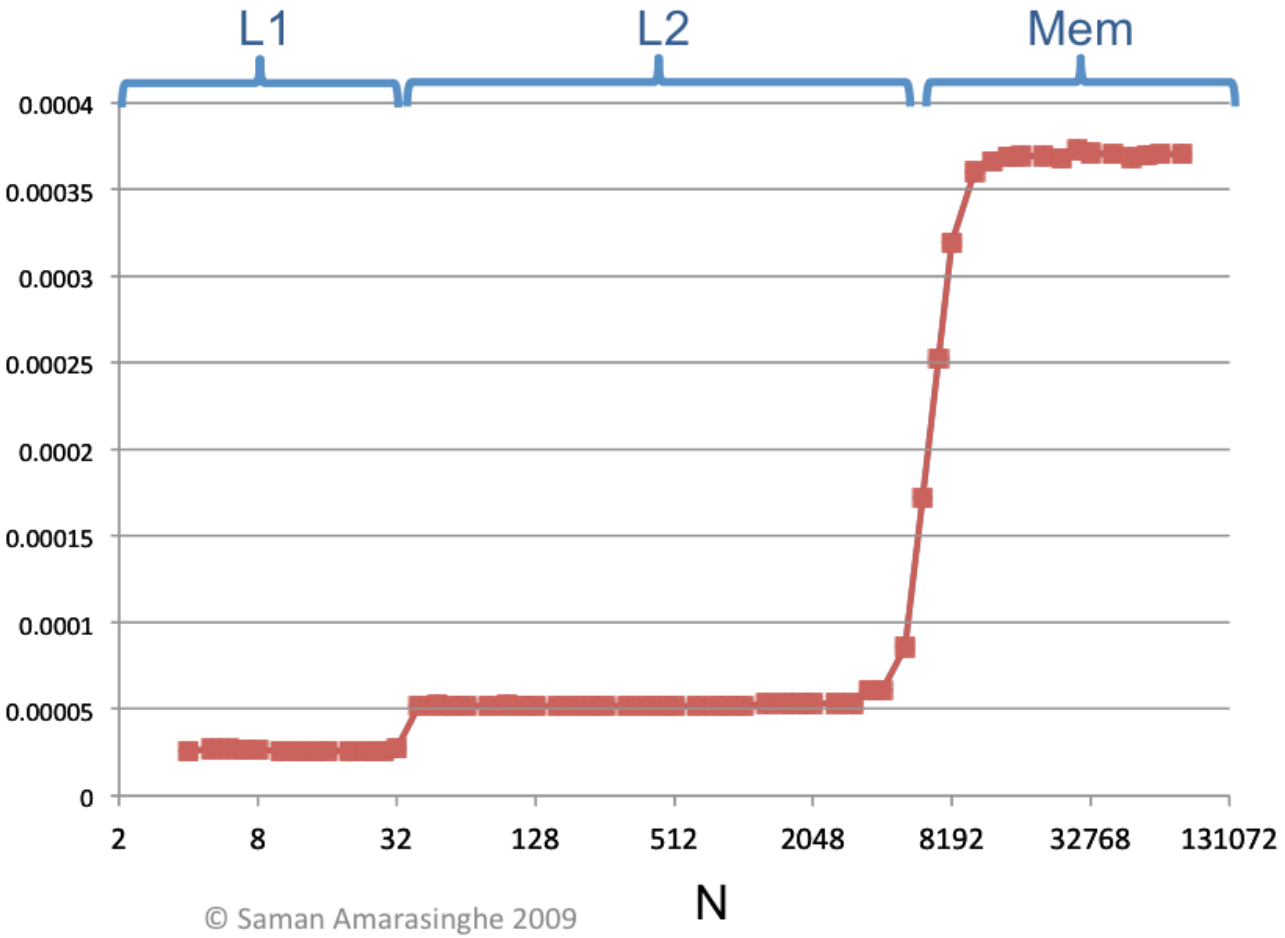
It’s Caches All The Way Down



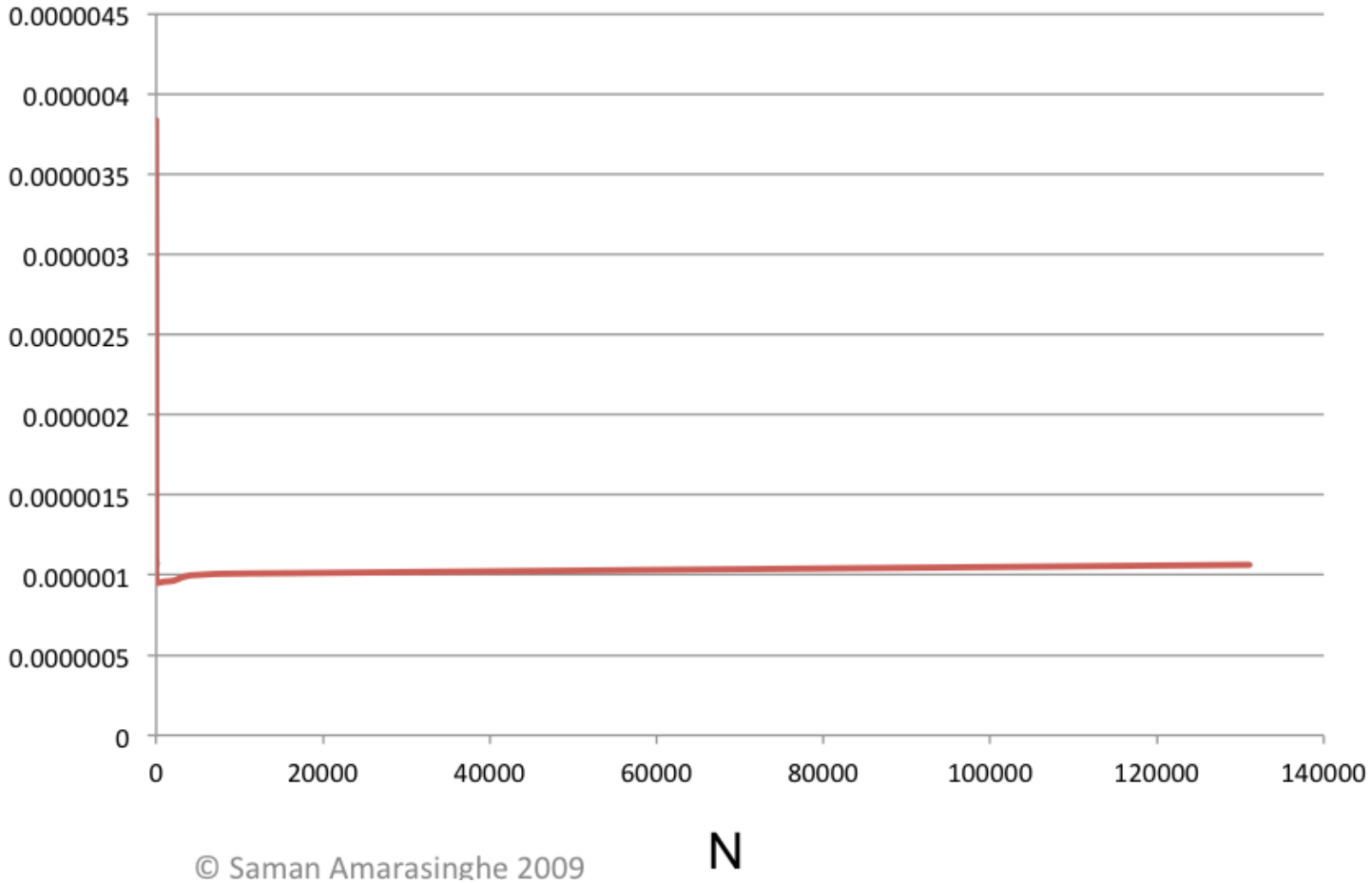
Prefetching: Program

```
for(rep=0; rep < REP; rep++)
    for(a=0; a <  N ; a++)
        A[a] = A[a] +  1;
```

Prefetching: Disabled

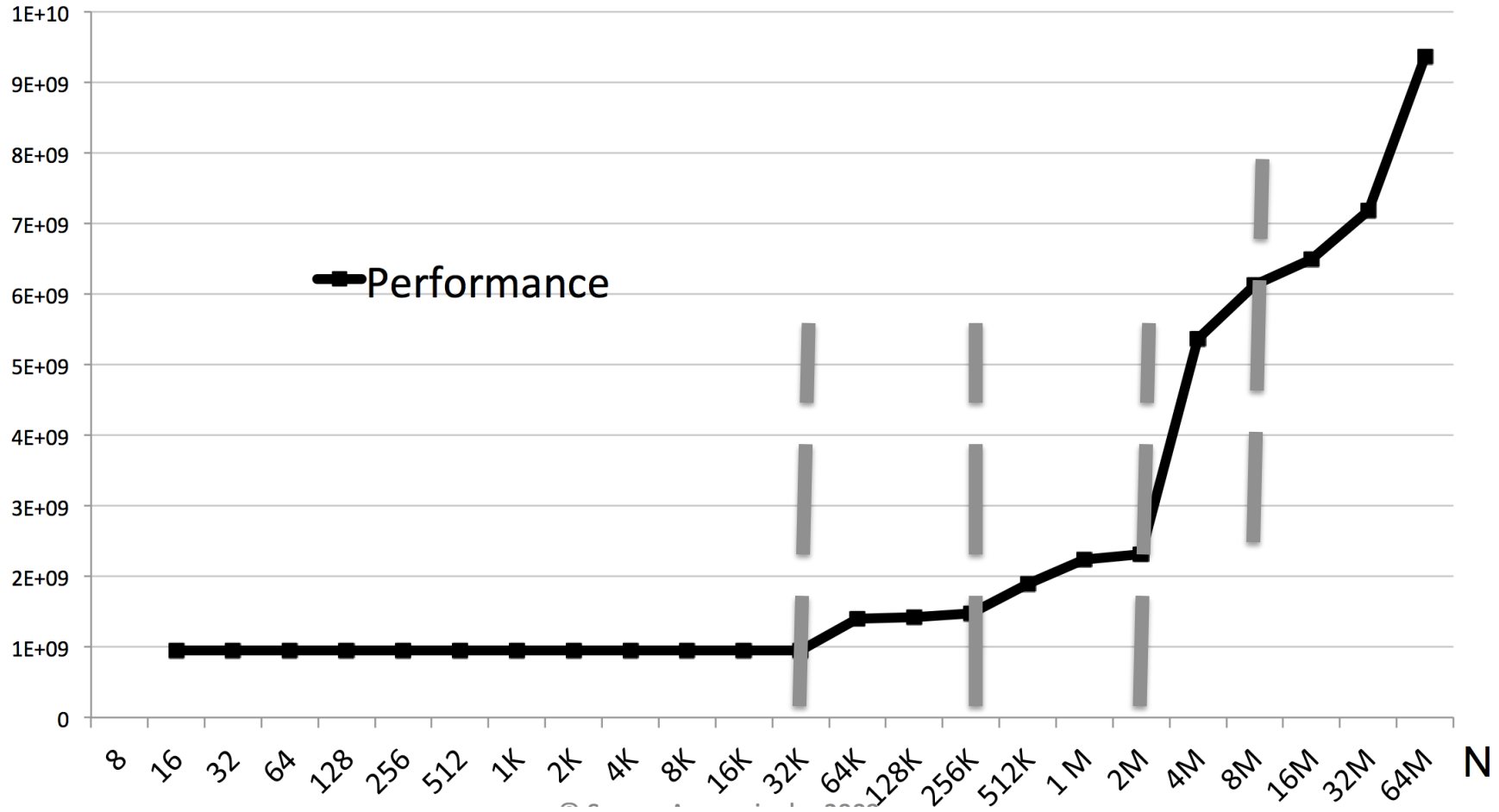


Prefetching: Enabled



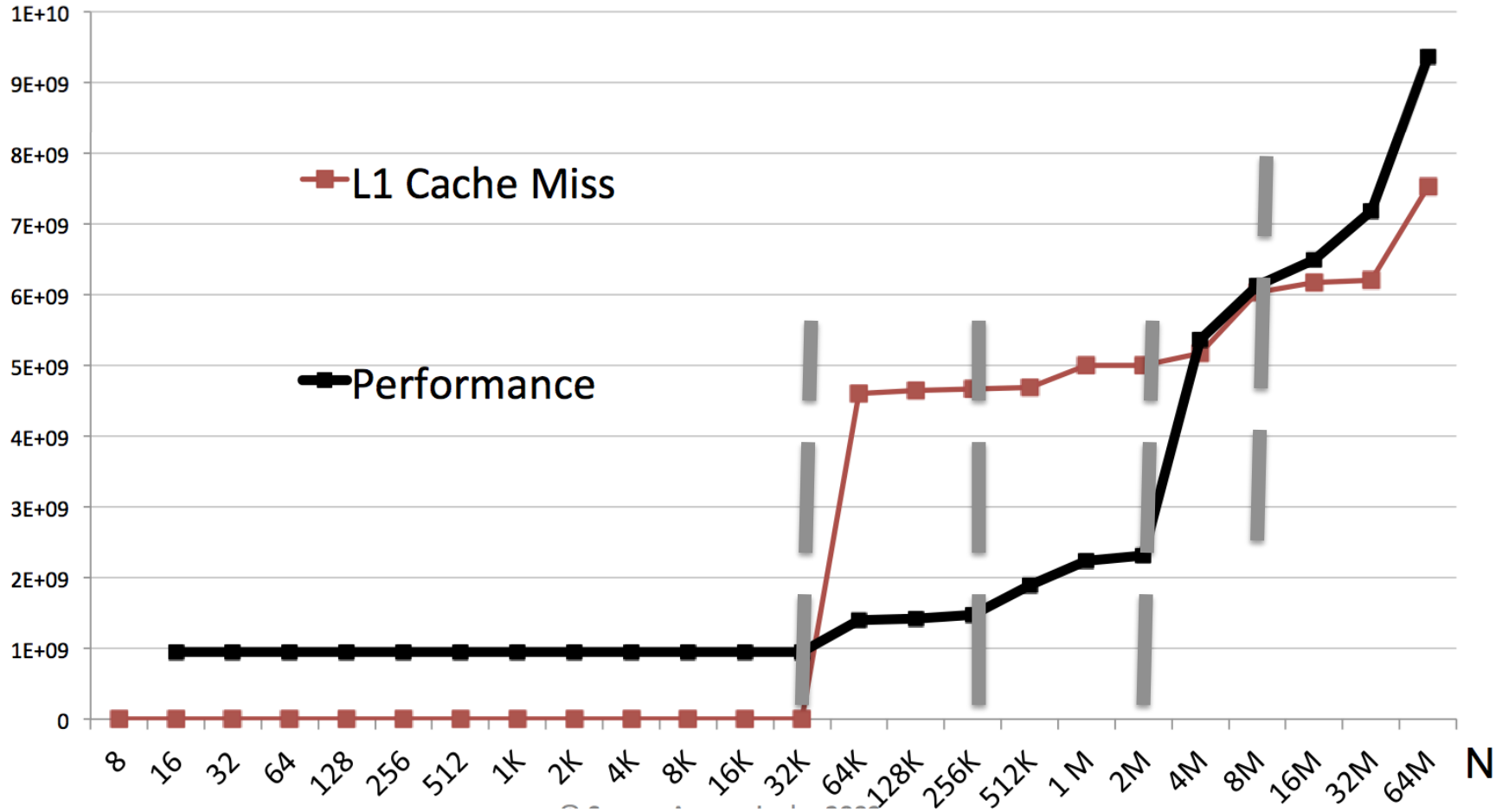
Caches in Benchmarks

Prof. Saman Amarasinghe, MIT 2009



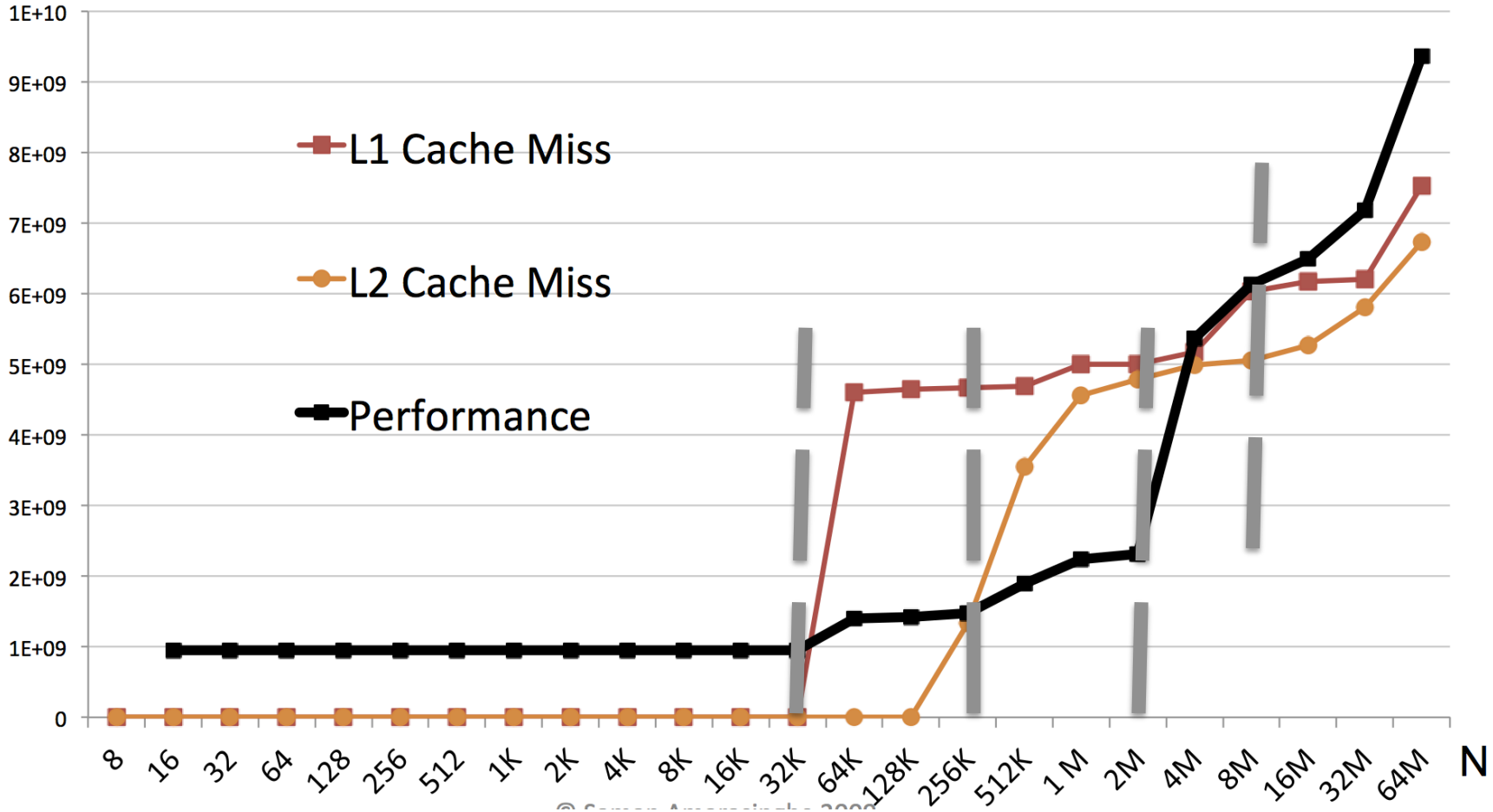
Caches in Benchmarks

Prof. Saman Amarasinghe, MIT 2009



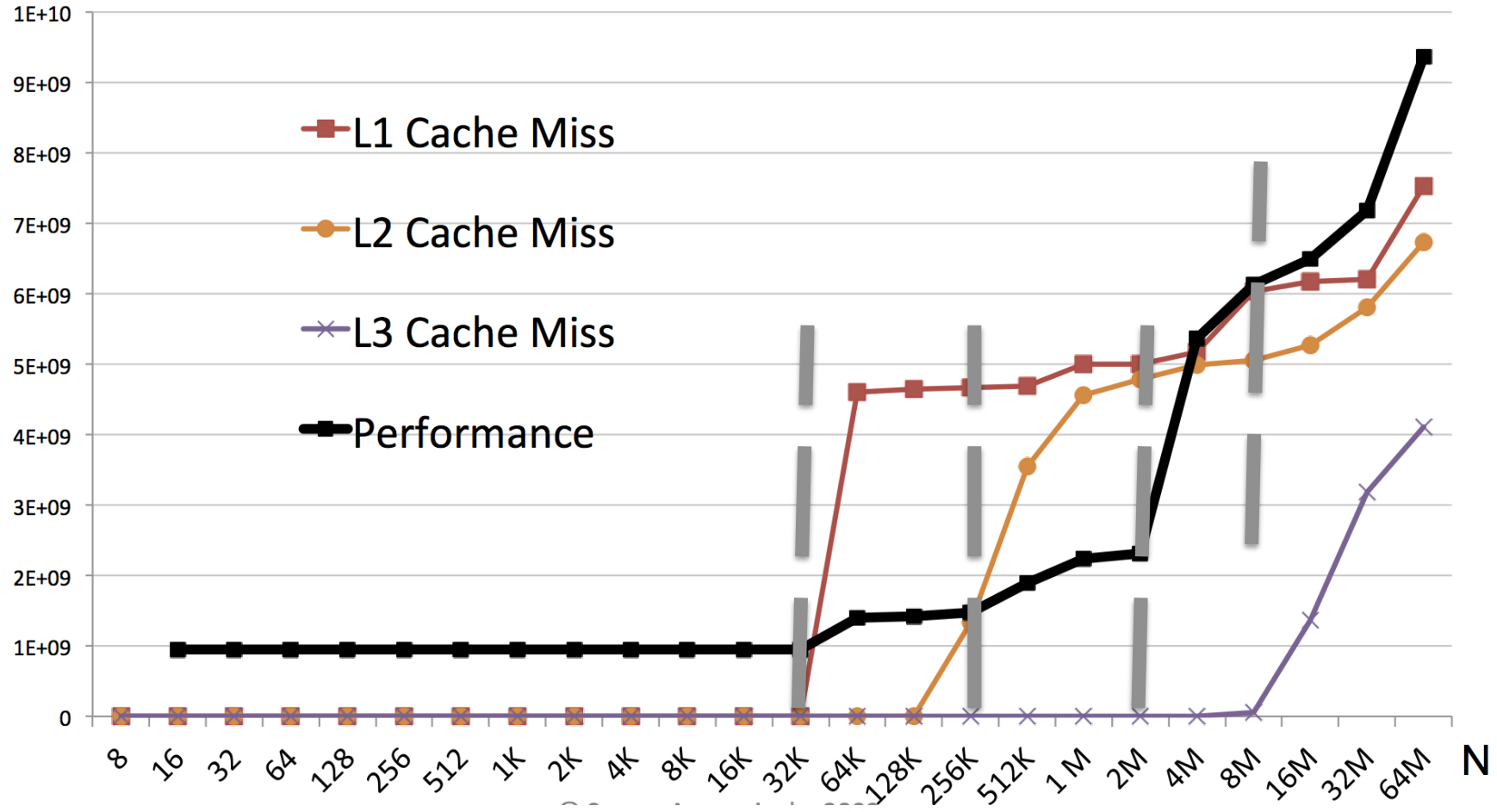
Caches in Benchmarks

Prof. Saman Amarasinghe, MIT 2009



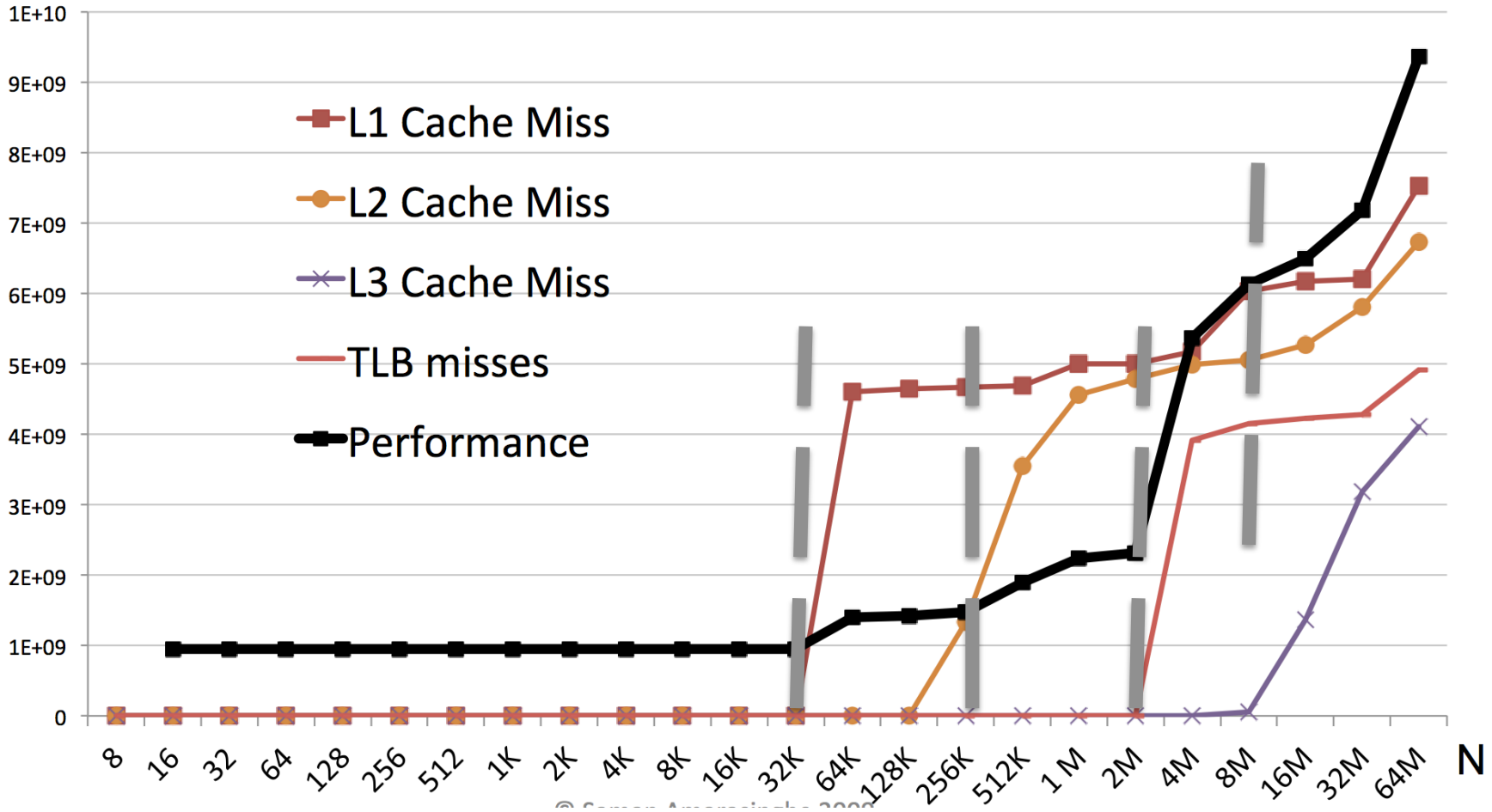
Caches in Benchmarks

Prof. Saman Amarasinghe, MIT 2009



Caches in Benchmarks

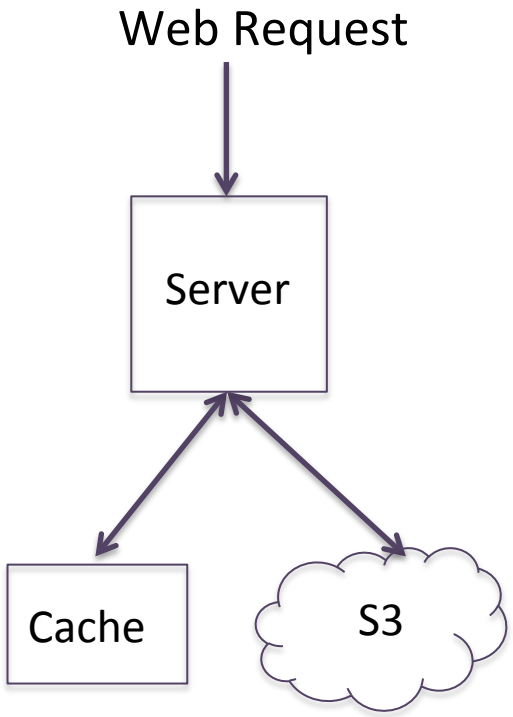
Prof. Saman Amarasinghe, MIT 2009



Website Serving Images

Access 1 image 1000 times

- Latency measured for each access
- Start measuring immediately
- 3 runs
- Find mean
- Dev environment

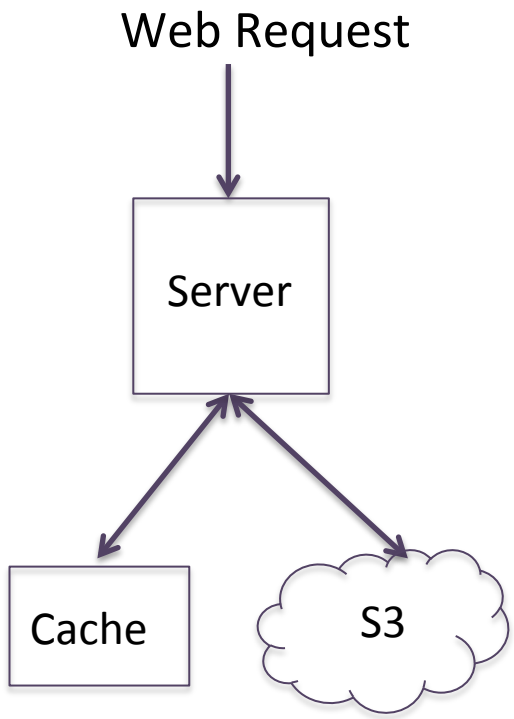


Wrong About the Machine

- Cache, cache, cache, cache!
- Warmup & timing

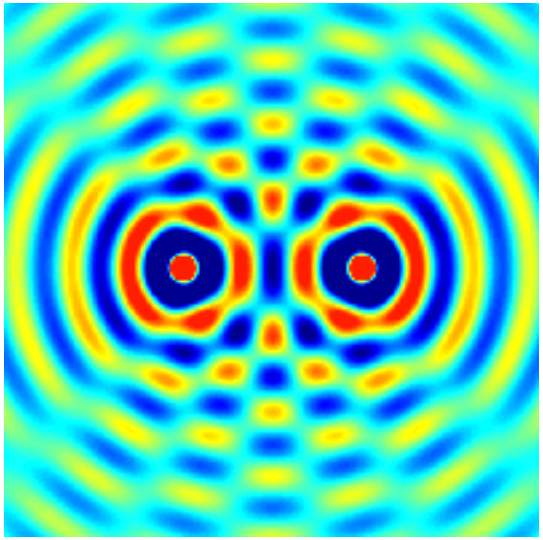
Website Serving Images

- ✗ Access 1 image 1000 times
  - Latency measured for each access
- ✗ Start measuring immediately
  - 3 runs
  - Find mean
  - Dev environment

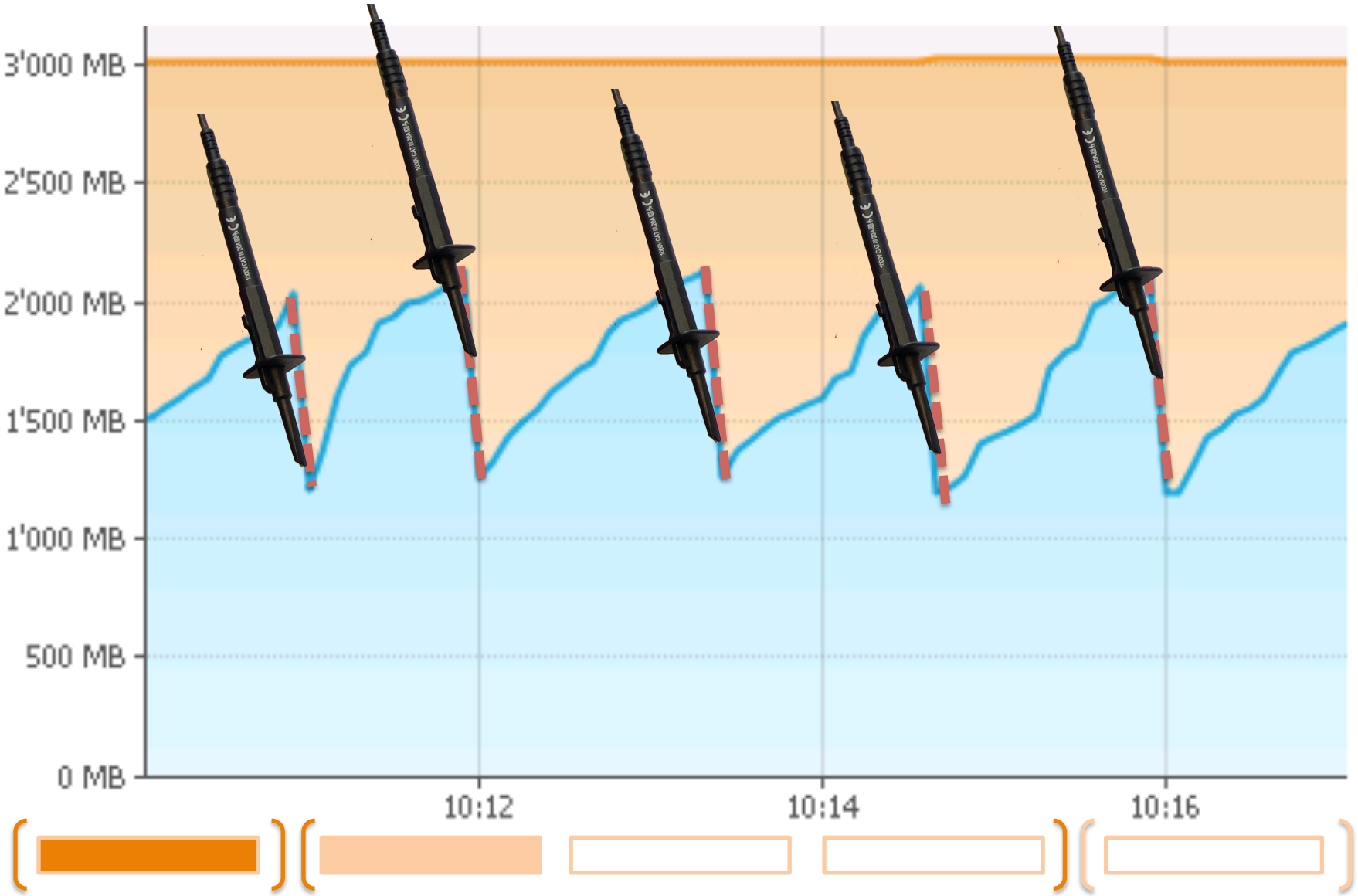


Wrong About the Machine

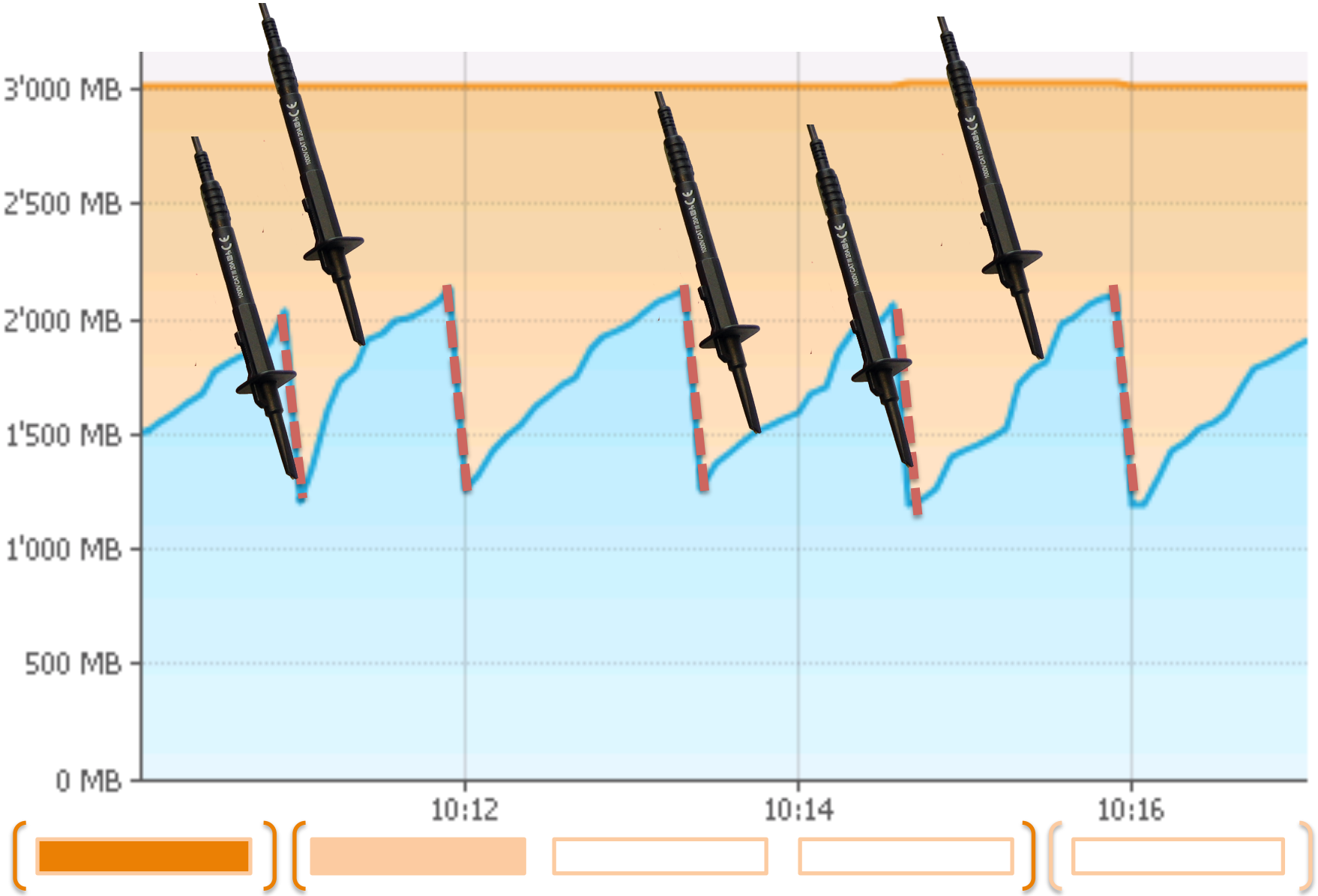
- Cache, cache, cache, cache!
- Warmup & timing
- Periodic interference



Periodic Interference

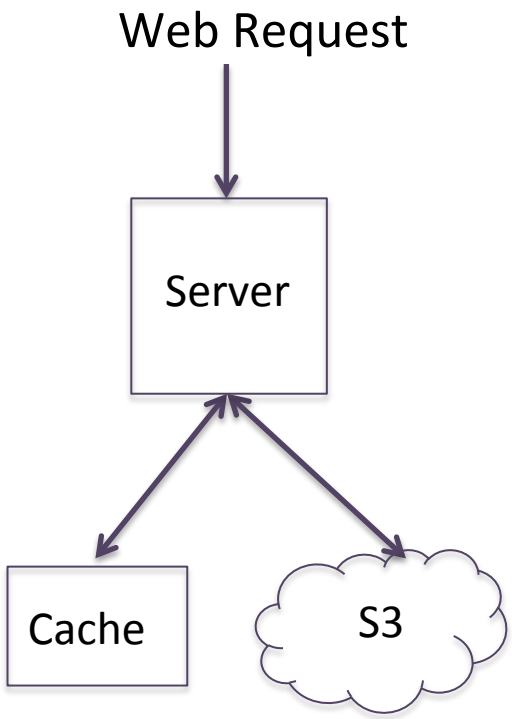


Periodic Interference



Website Serving Images

- ✗ Access 1 image 1000 times
- ✗ Latency measured for each access
- ✗ Start measuring immediately
- 3 runs
- Find mean
- Dev environment

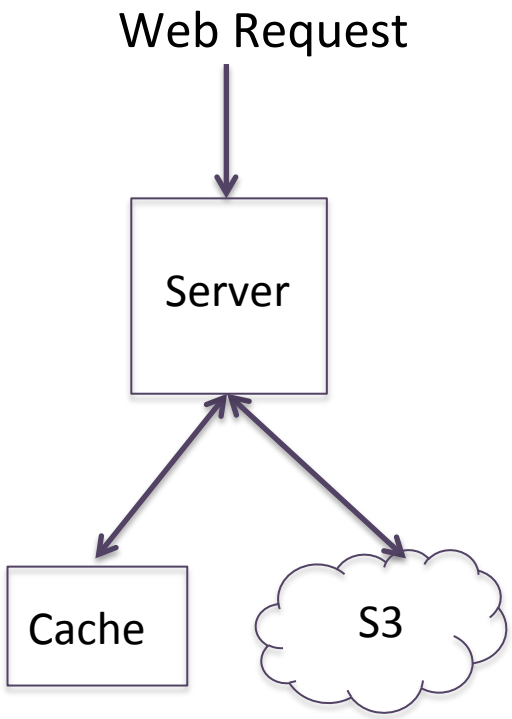


Wrong About the Machine

- Cache, cache, cache, cache!
- Warmup & timing
- Periodic interference
- Test != Prod

Website Serving Images

- ✗ Access 1 image 1000 times
- ✗ Latency measured for each access
- ✗ Start measuring immediately
  - 3 runs
  - Find mean
- ✗ Dev environment



Wrong About the Machine

- Cache, cache, cache, cache!
- Warmup & timing
- Periodic interference
- Test != Prod
- Power mode changes

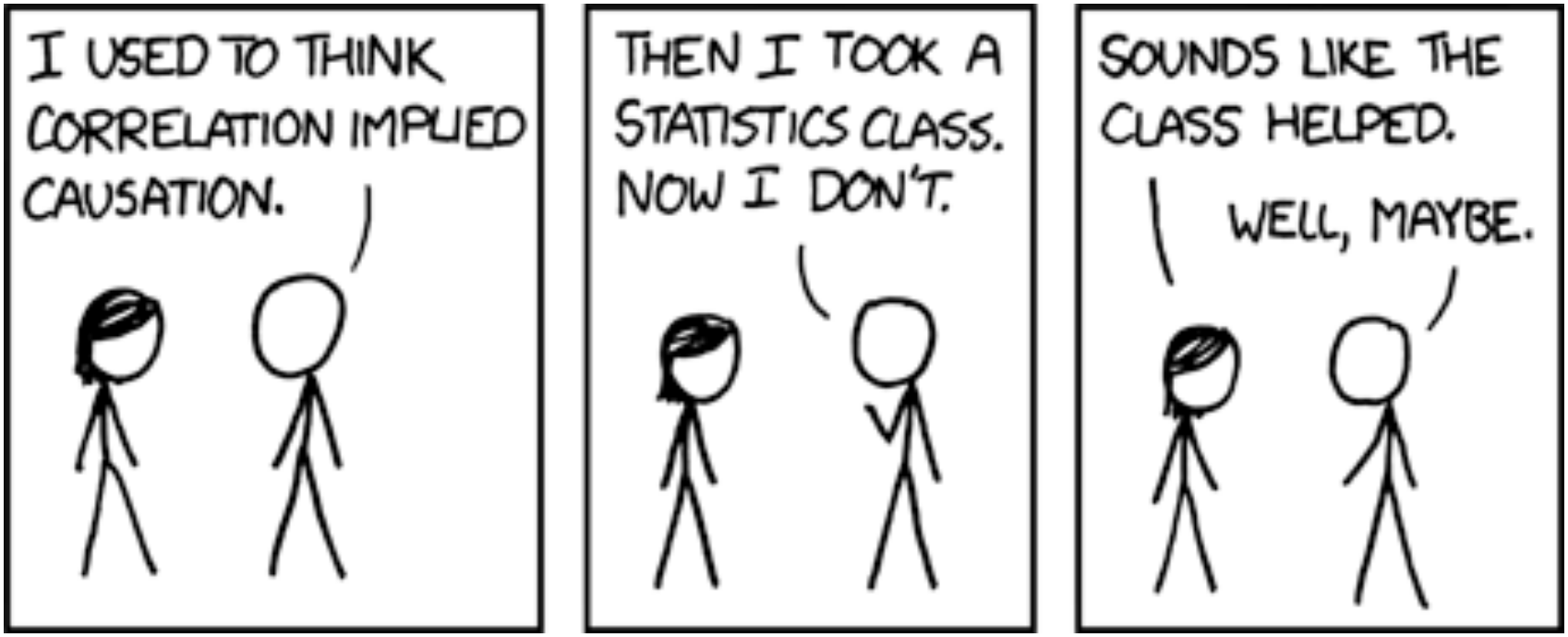
Power Modes

$ cat /sys/devices/system/cpu/*/cpufreq/scaling_governor

“ondemand” OR “performance”

Current CPU frequencies:

$ grep "MHz" /proc/cpuinfo

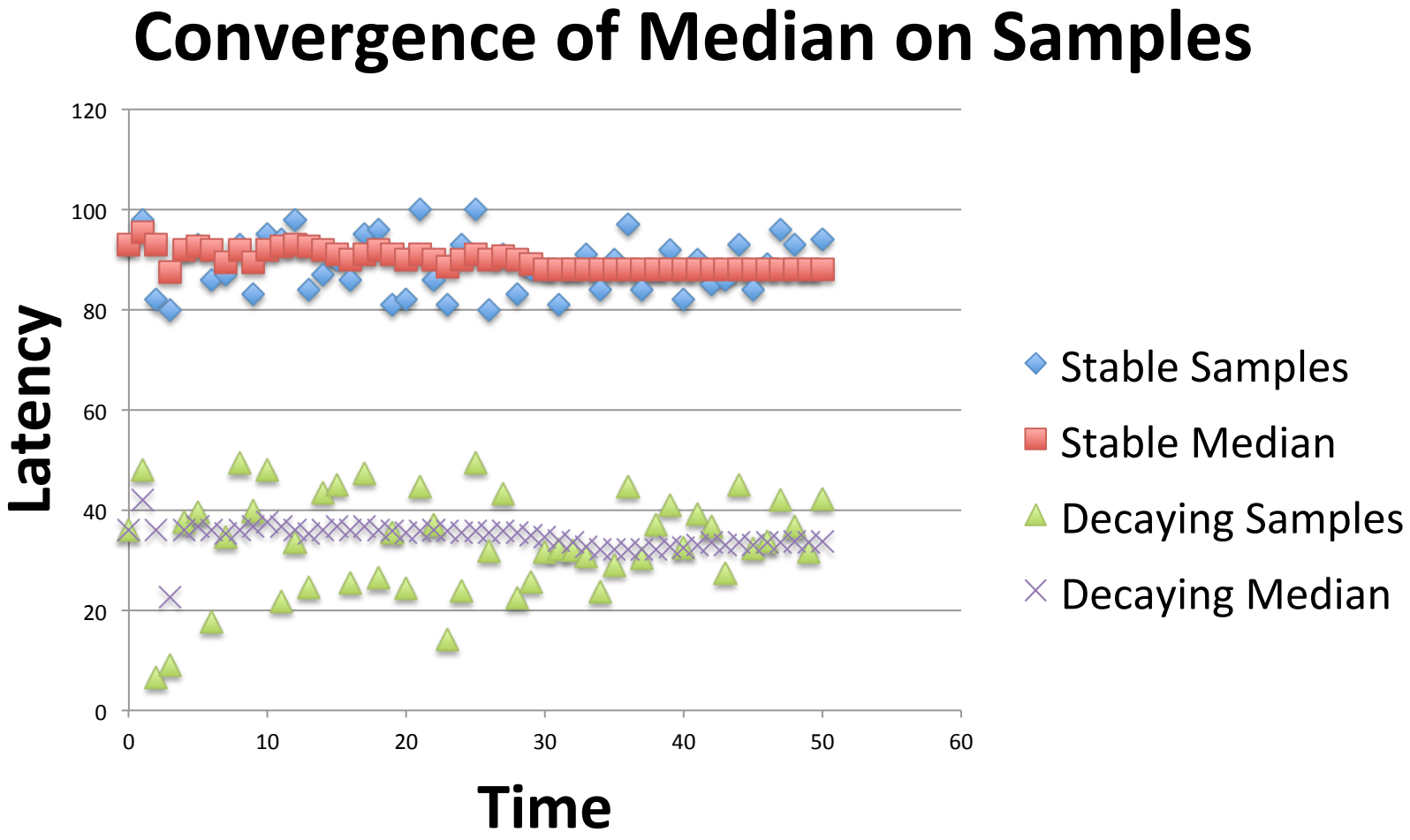


YOU'RE WRONG ABOUT THE STATS

Wrong About Stats

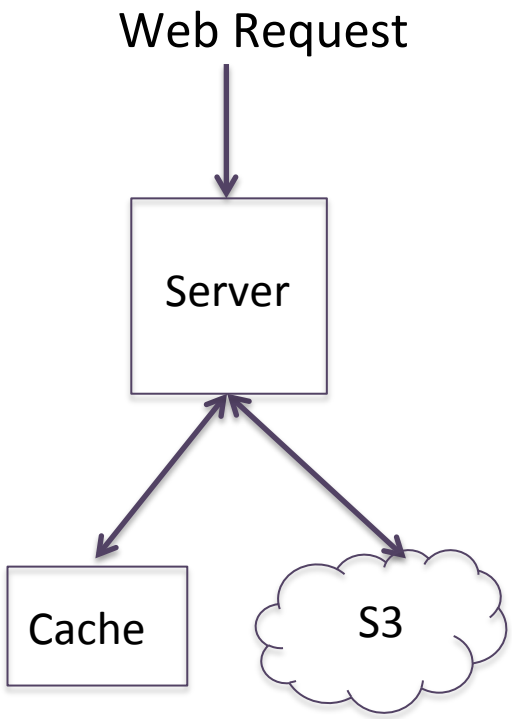
- Too few samples

Wrong About Stats



Website Serving Images

- ~~Access 1 image 1000 times~~
- ~~Latency measured for each access~~
- ~~Start measuring immediately~~
- ~~3 runs~~
- Find mean
- ~~Dev machine~~

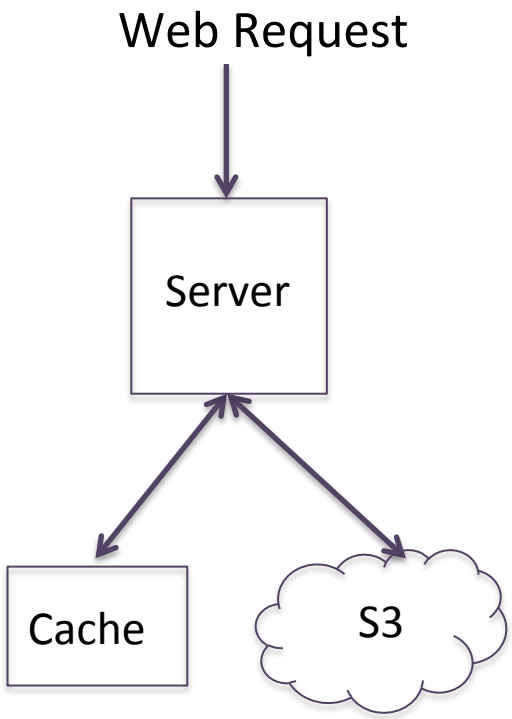


Wrong About Stats

- Too few samples
- Gaussian (not)

Website Serving Images

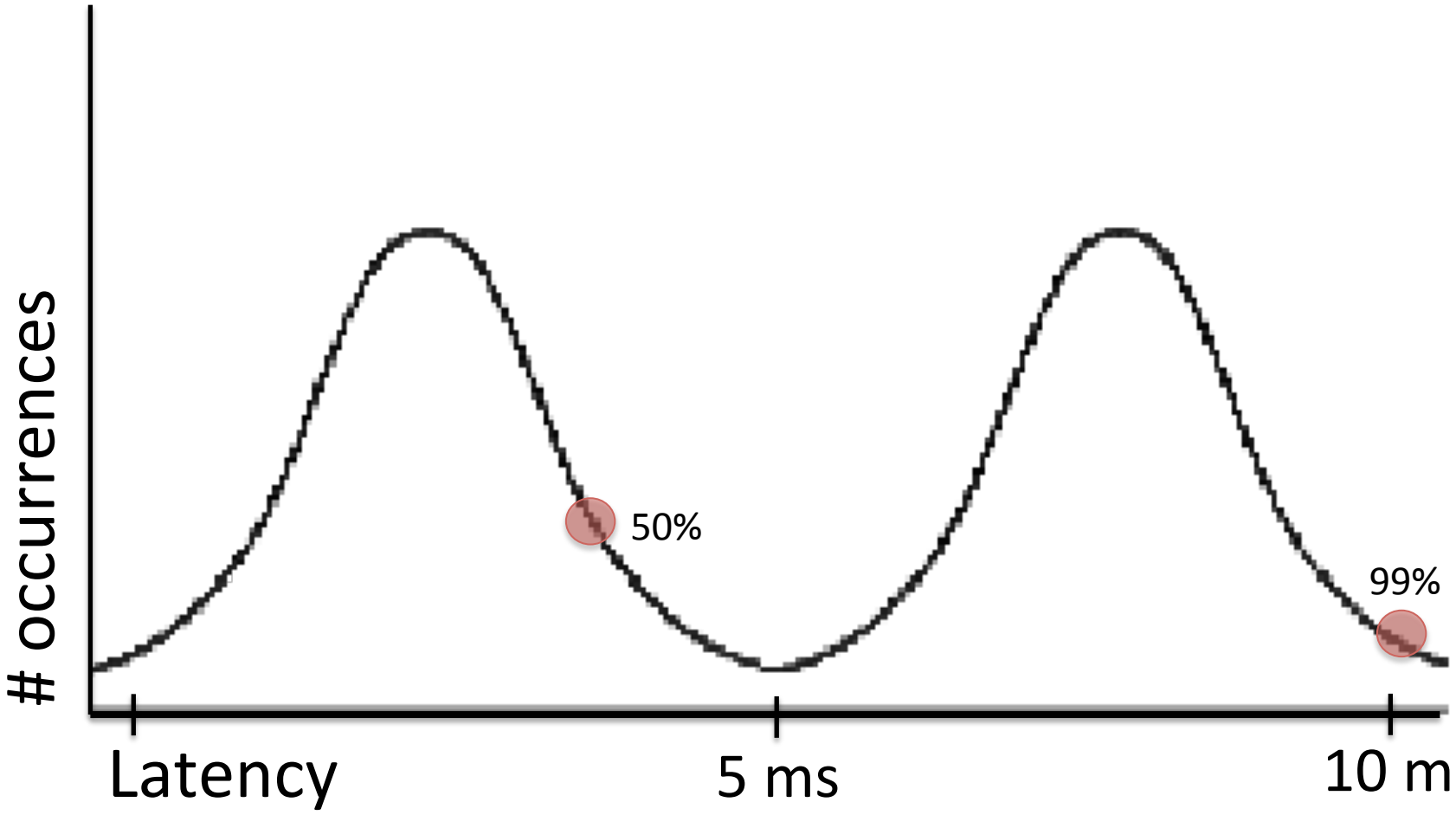
- Access 1 image 1000 times
- Latency measured for each access
- Start measuring immediately
- 3 runs
- Find mean
- Dev machine



Wrong About Stats

- Too few samples
- Gaussian (not)
- Multimodal distribution

Multimodal Distribution



Multimodal Distribution

Using Kernel Density Estimates to investigate Multimodality

By B. W. SILVERMAN

University of Bath, U.K.

[Received August 1980]

SUMMARY

A technique for using kernel density estimates to investigate the number of modes in a population is described and discussed. The amount of smoothing is chosen automatically in a natural way.

Keywords: DENSITY ESTIMATE; MODE; BOOTSTRAP; TOTAL POSITIVITY; CHONDRITES; BUMP HUNTING

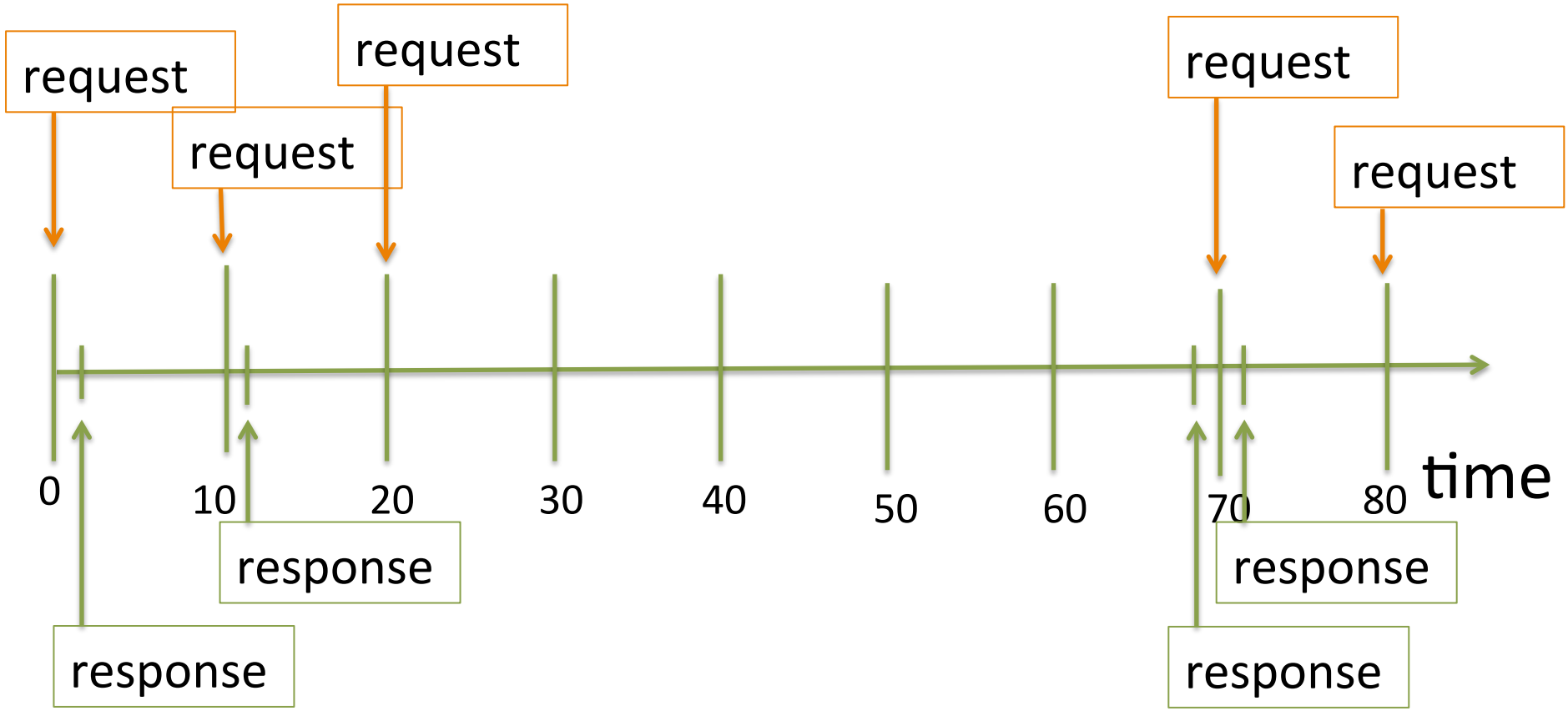
1. INTRODUCTION

INVESTIGATION of the number of modes or maxima in a density or its derivative has been considered by several authors, for example Cox (1966) and Good and Gaskins (1980). Most

Wrong About Stats

- Too few samples
- Gaussian (not)
- Multimodal distribution
- Outliers

Coordinated Omission



Wrong About Stats

- Too few samples
- Gaussian (not)
- Multimodal distribution
- Outliers

YOU'RE WRONG ABOUT WHAT MATTERS

Wrong About What Matters

- Premature optimization

"Programmers waste enormous amounts of time thinking about … the speed of noncritical parts of their programs … Forget about small efficiencies …97% of the time: **premature optimization is the root of all evil**. Yet we should not pass up our opportunities in that critical 3%."

-- Donald Knuth

Wrong About What Matters

- Premature optimization
- Unrepresentative workloads

Wrong About What Matters

- Premature optimization
- Unrepresentative workloads
- Memory pressure

Wrong About What Matters

- Premature optimization
- Unrepresentative workloads
- Memory pressure
- Hidden components

Wrong About What Matters

- Premature optimization
- Unrepresentative workloads
- Memory pressure
- Hidden components
- Reproducibility of measurements

BECOMING LESS WRONG

User Actions Matter

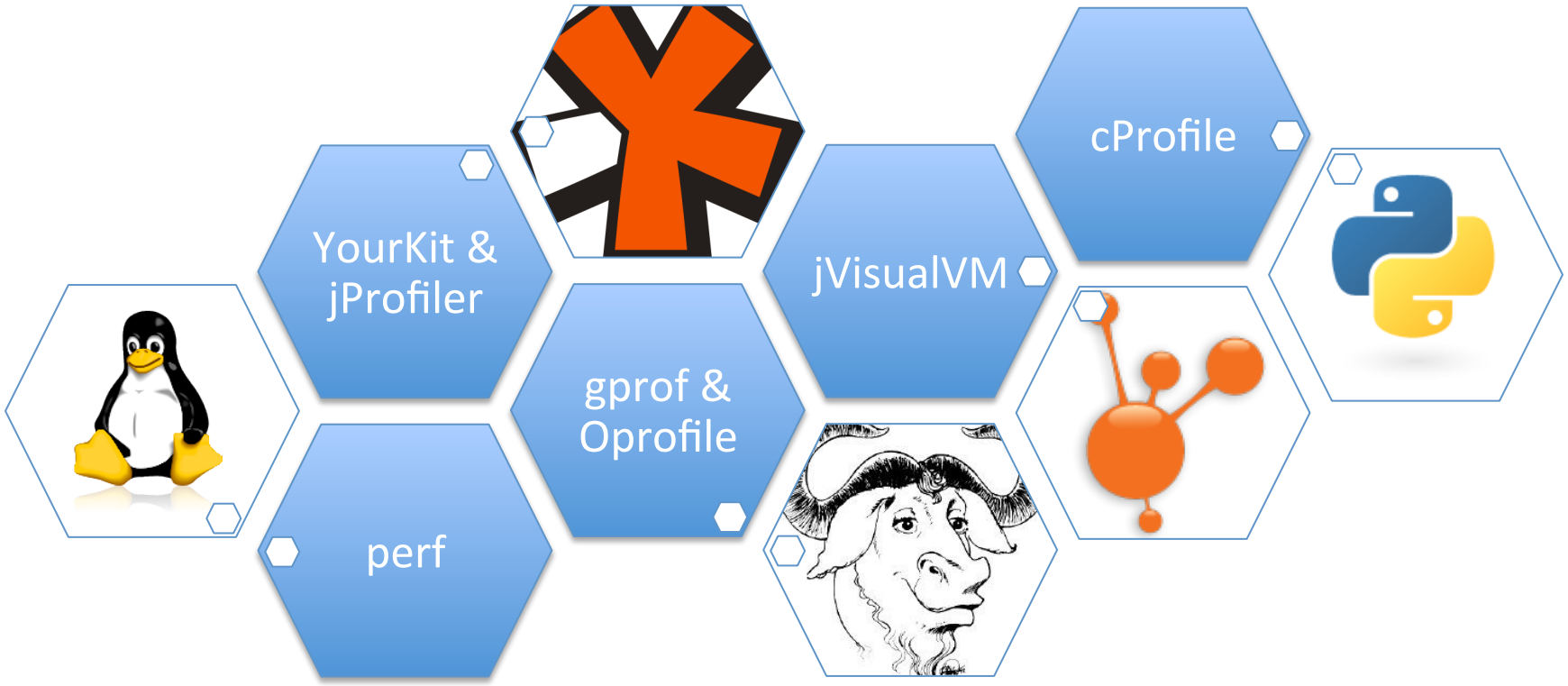
X > Y for workload Z

with trade offs A, B, and C

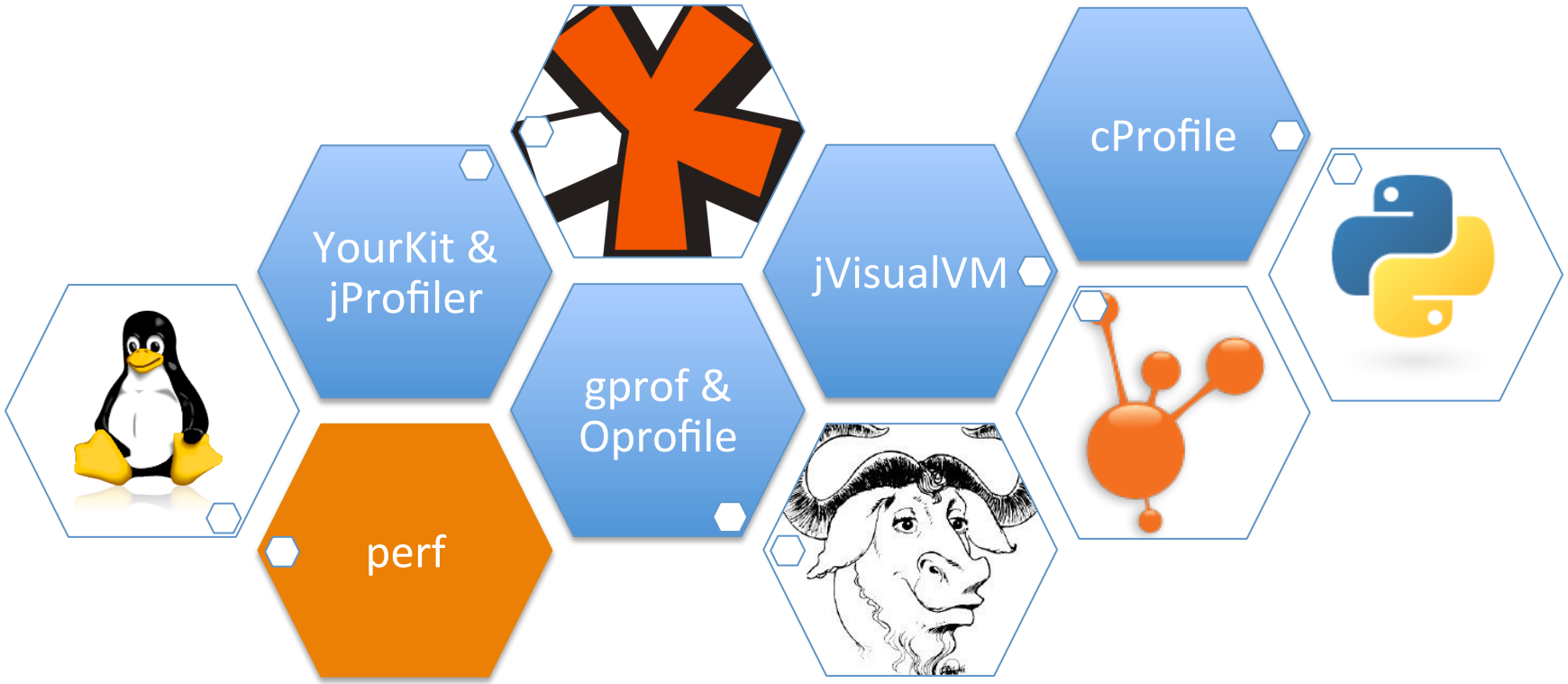
- http://www.toomuchcode.org/

Profiling

Profiling


YourKit &
jProfiler
jVisualVM
cProfile
gprof &
Oprofile
perf

Profiling


YourKit &
jProfiler
jVisualVM
cProfile
gprof &
Oprofile
perf

perf

Various basic CPU statistics, system wide, for 10 seconds

perf stat -e cycles,instructions,cache-misses -a sleep 10

Count system calls for the entire system, for 5 seconds

perf stat -e 'syscalls:sys_enter_*' -a sleep 5

Sample CPU stack traces, once every 10,000 Level 1 data cache misses, for 5 seconds

perf record -e L1-dcache-load-misses -c 10000 -ag -- sleep 5

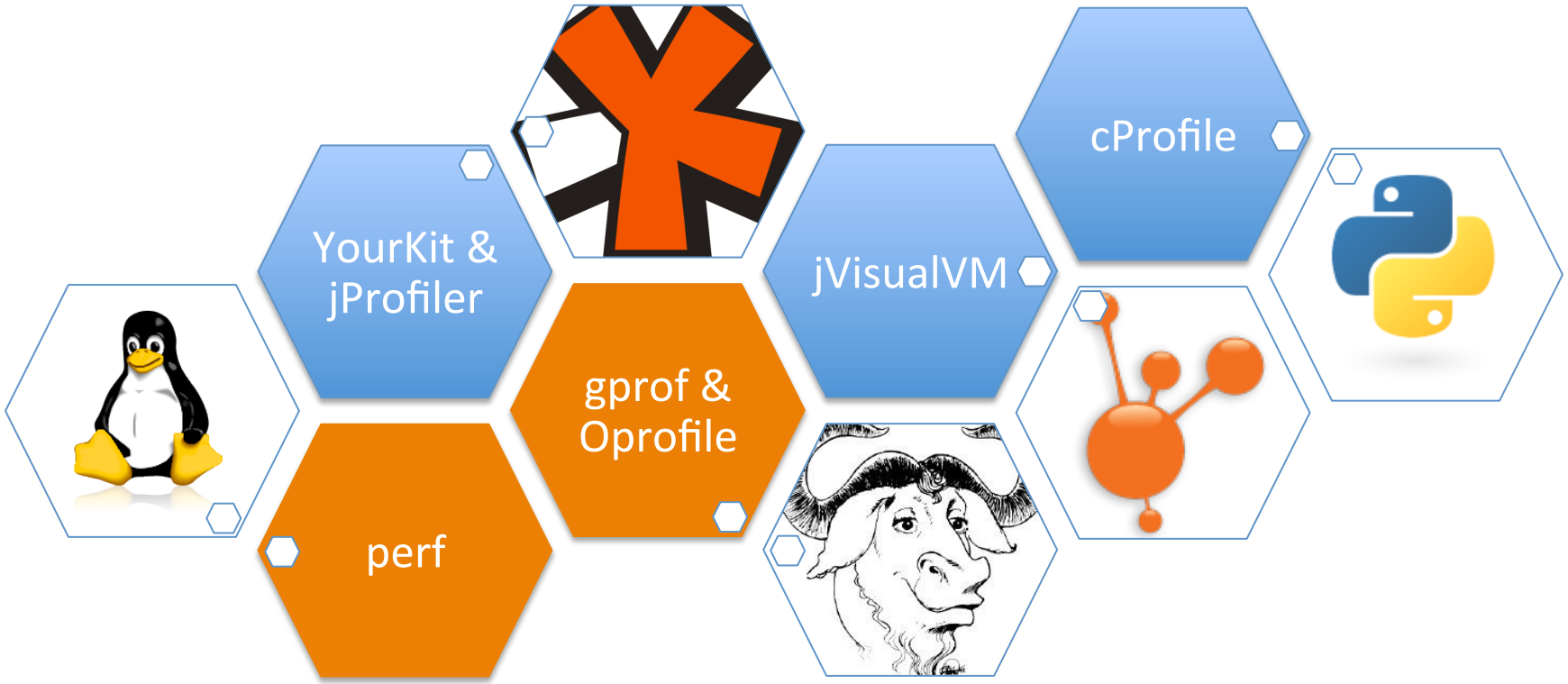
http://www.brendangregg.com/perf.html

perf

```
# perf stat gzip file1

 Performance counter stats for 'gzip file1':

       1920.159821 task-clock                #    0.991 CPUs utilized
                13 context-switches          #    0.007 K/sec
                 0 CPU-migrations            #    0.000 K/sec
               258 page-faults               #    0.134 K/sec
     5,649,595,479 cycles                    #    2.942 GHz                      [83.43%]
     1,808,339,931 stalled-cycles-frontend   #   32.01% frontend cycles idle     [83.54%]
     1,171,884,577 stalled-cycles-backend    #   20.74% backend  cycles idle     [66.77%]
     8,625,207,199 instructions              #    1.53  insns per cycle
                                             #    0.21  stalled cycles per insn [83.51%]
     1,488,797,176 branches                  #  775.351 M/sec                    [82.58%]
        53,395,139 branch-misses             #    3.59% of all branches          [83.78%]

       1.936842598 seconds time elapsed
```

http://www.brendangregg.com/perf.html

Profiling


YourKit &
jProfiler
jVisualVM
cProfile
gprof &
Oprofile
perf

gprof: Where Does It Spend Its Time?

- Compile with profiling

```
$ gcc -Wall -pg test_gprof.c test_gprof_new.c -o test_gprof
```

- Execute the code

```
$ ./test_gprof
```

- Run the gprof

```
$  gprof test_gprof gmon.out > analysis.txt
```

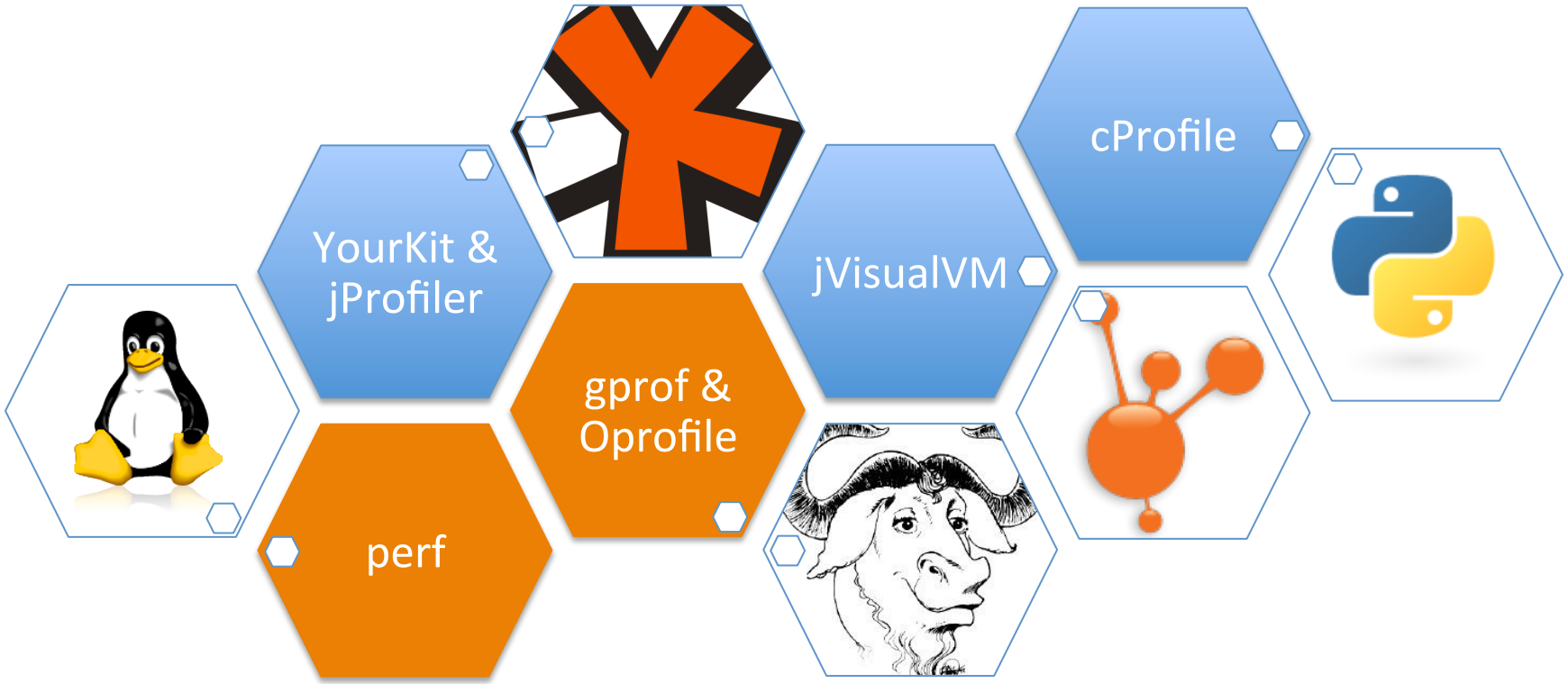
http://www.thegeekstuff.com/2012/08/gprof-tutorial/

gprof: Where Does It Spend Its Time?

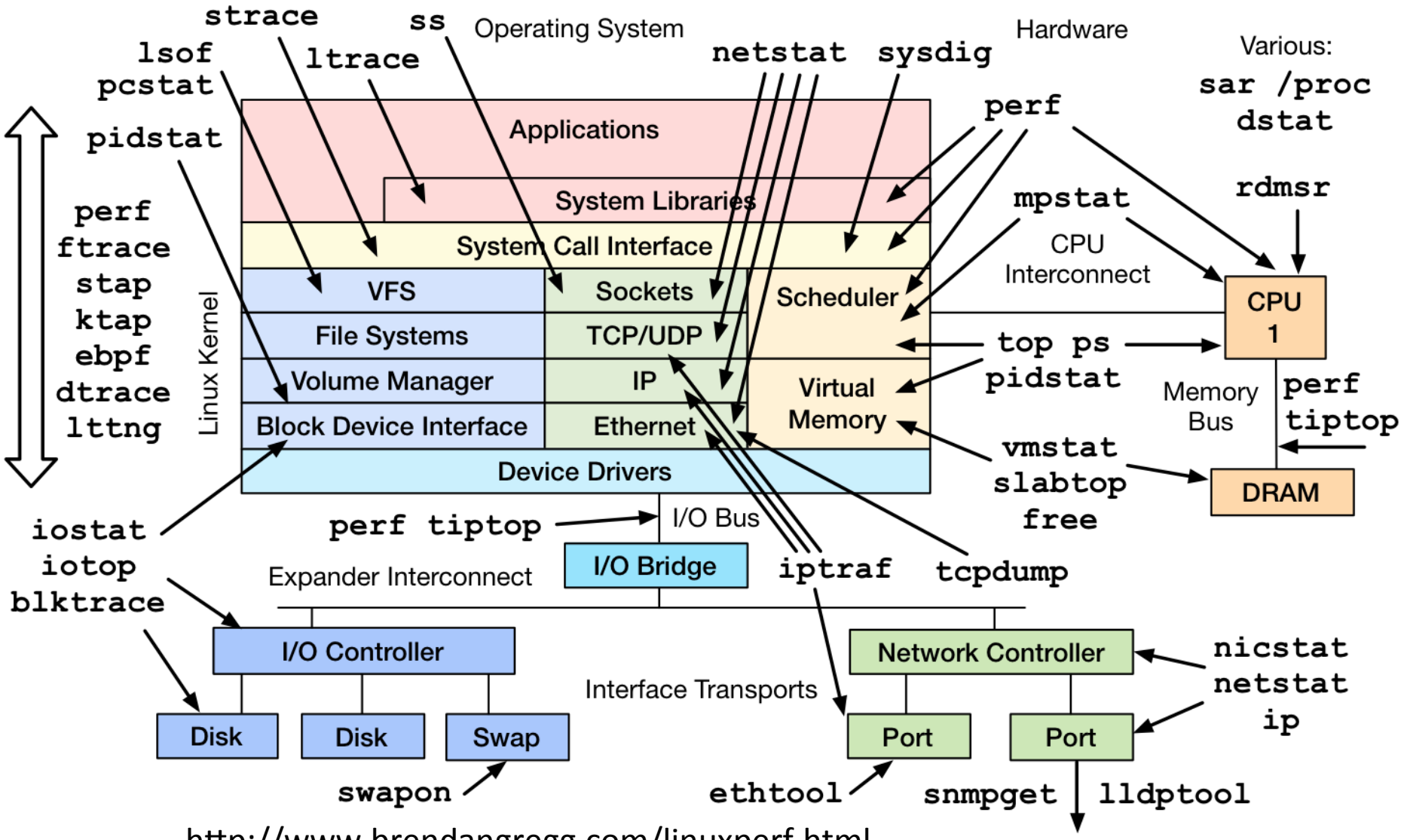
```
%    cumulative self              self   total
time seconds    seconds calls s/call s/call name
33.86 15.52     15.52    1    15.52  15.52  func2
33.82 31.02     15.50    1    15.50  15.50  new_func1
33.29 46.27     15.26    1    15.26  30.75  func1
0.07  46.30     0.03                       main
```

http://www.thegeekstuff.com/2012/08/gprof-tutorial/

Profiling

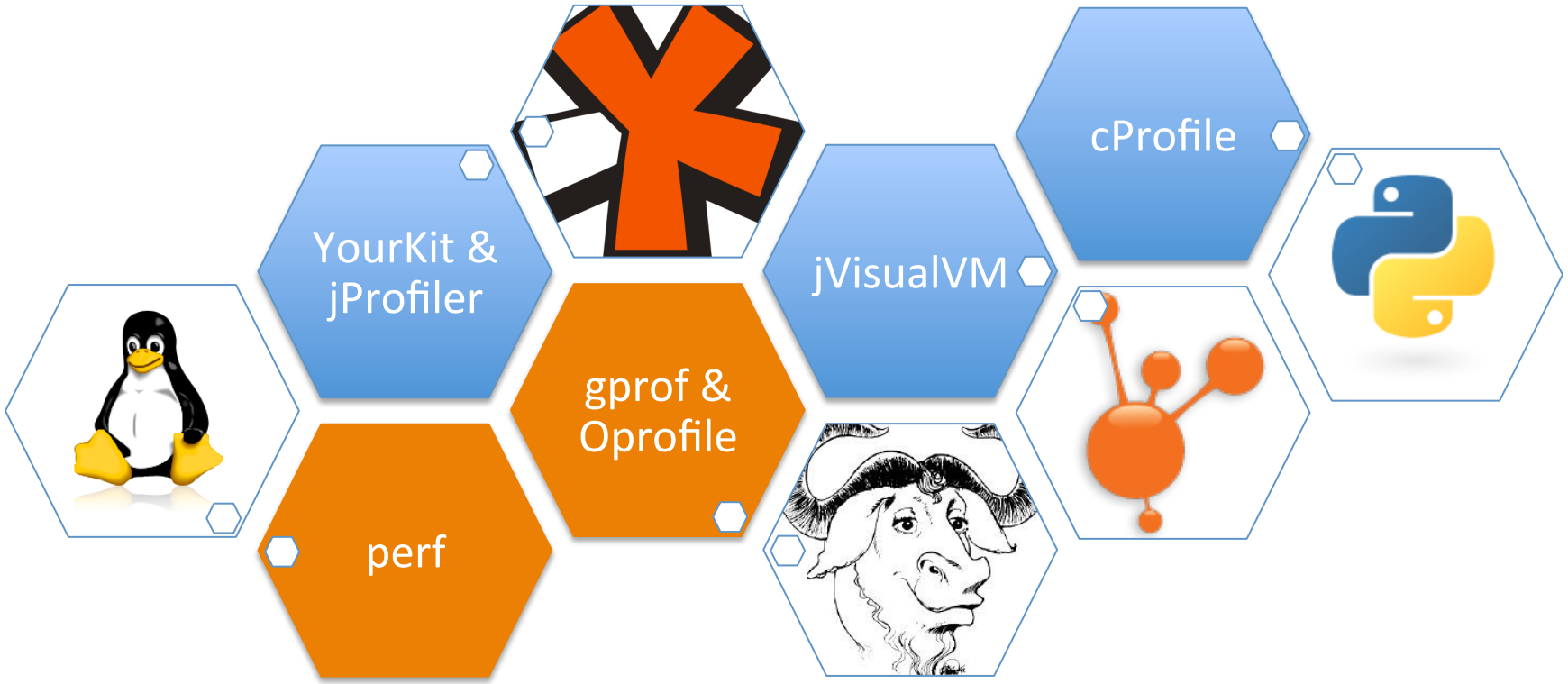

YourKit &
jProfiler
jVisualVM
cProfile
gprof &
Oprofile
perf

Linux Performance Observability Tools

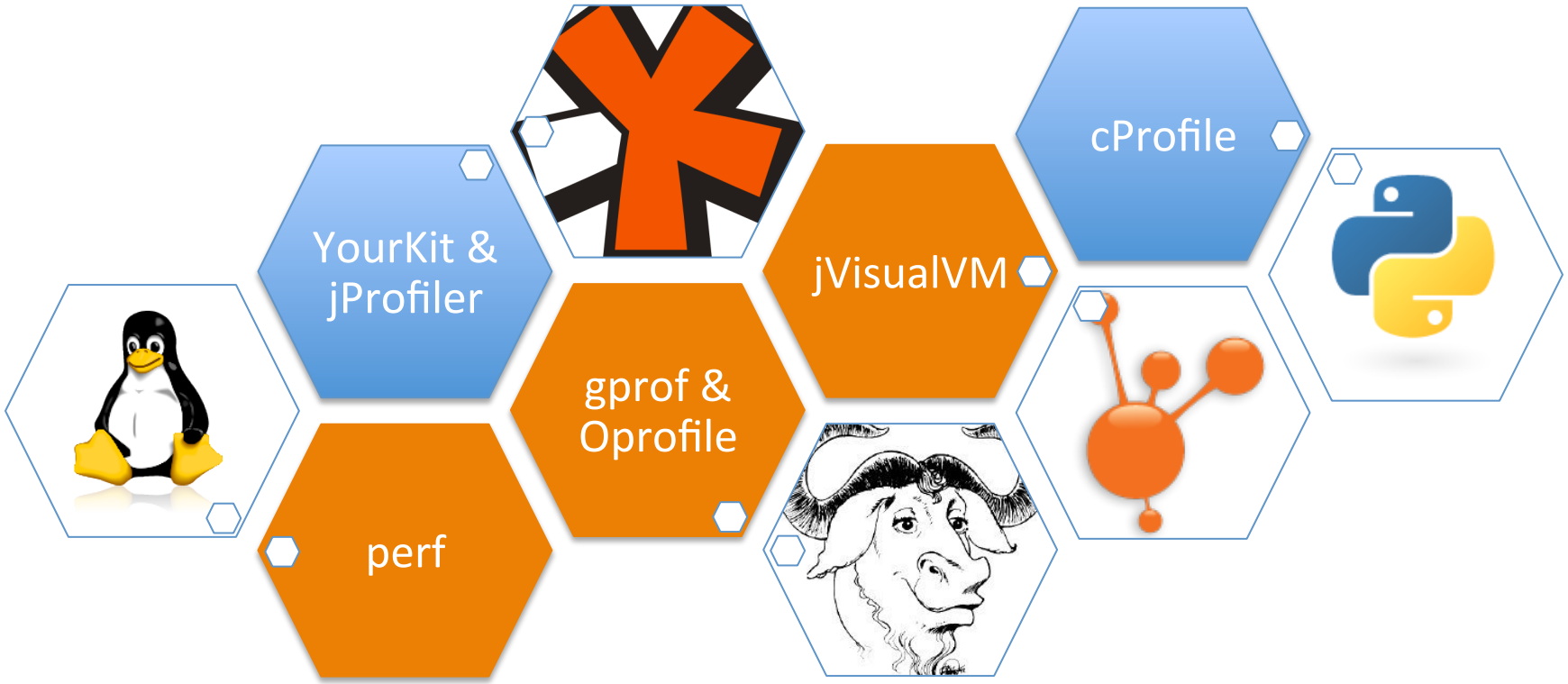


http://www.brendangregg.com/linuxperf.html

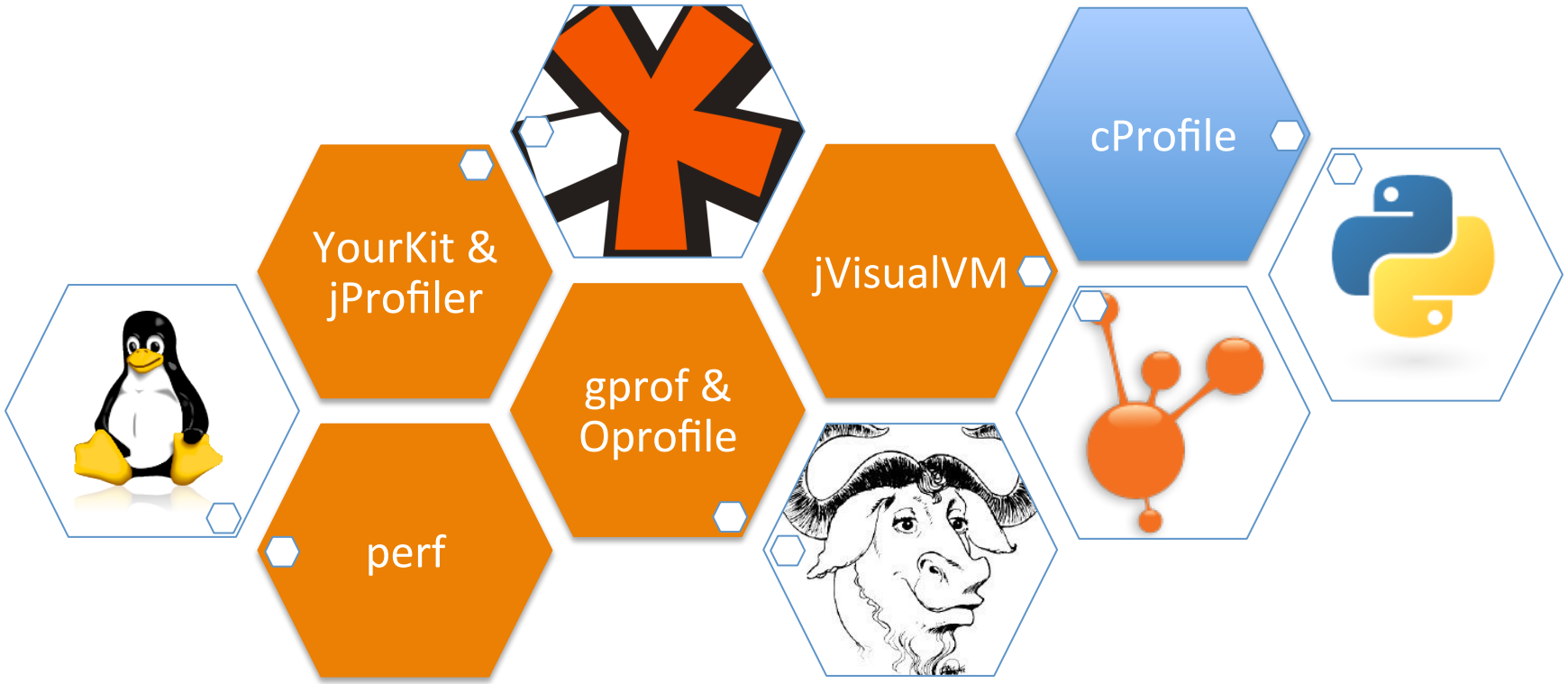
Profiling


YourKit &
jProfiler
jVisualVM
cProfile
gprof &
Oprofile
perf

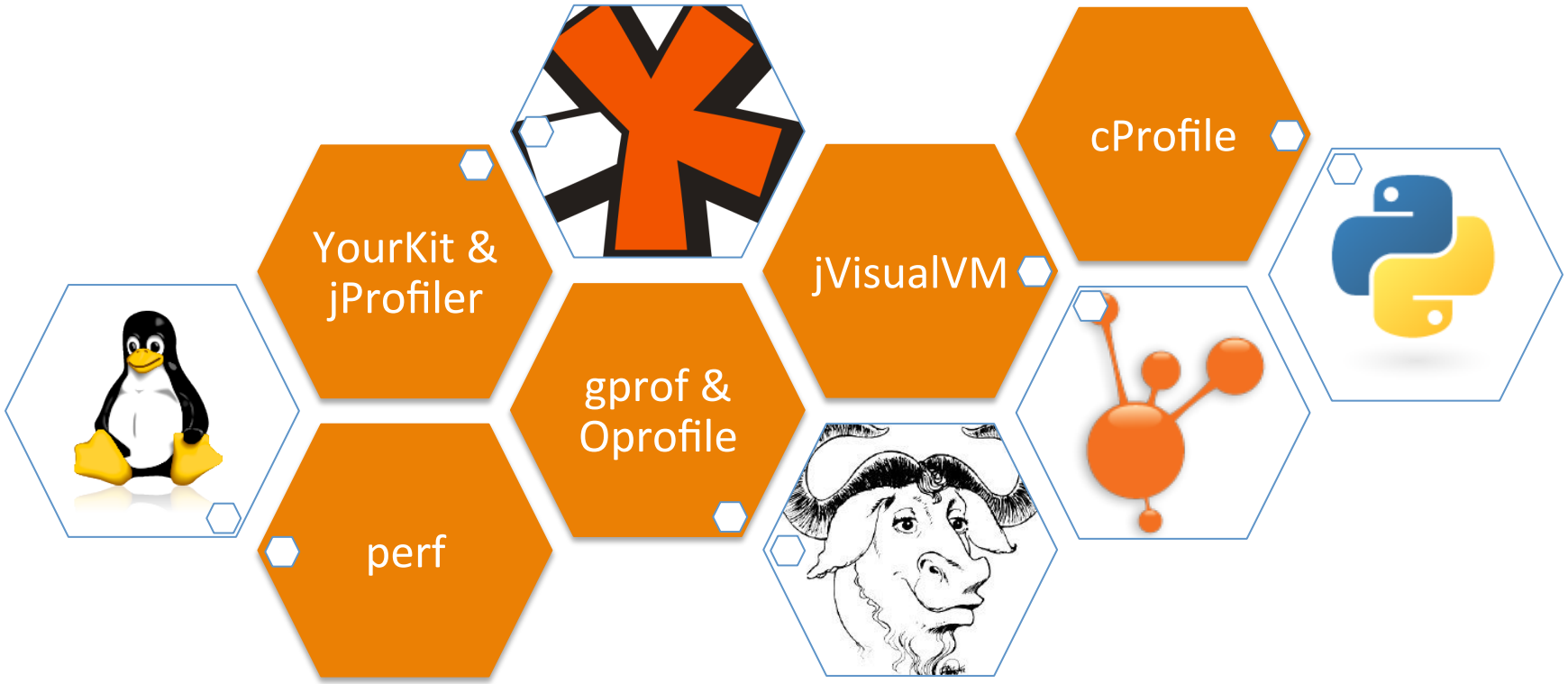
Profiling


YourKit &
jProfiler
gprof &
Oprofile
jVisualVM
cProfile
perf

Profiling


YourKit &
jProfiler
gprof &
Oprofile
jVisualVM
cProfile
perf

Profiling


YourKit &
jProfiler
gprof &
Oprofile
jVisualVM
cProfile
perf

Profiling

Code instrumentation

Aggregate over logs

Traces

Microbenchmarking: Blessing & Curse

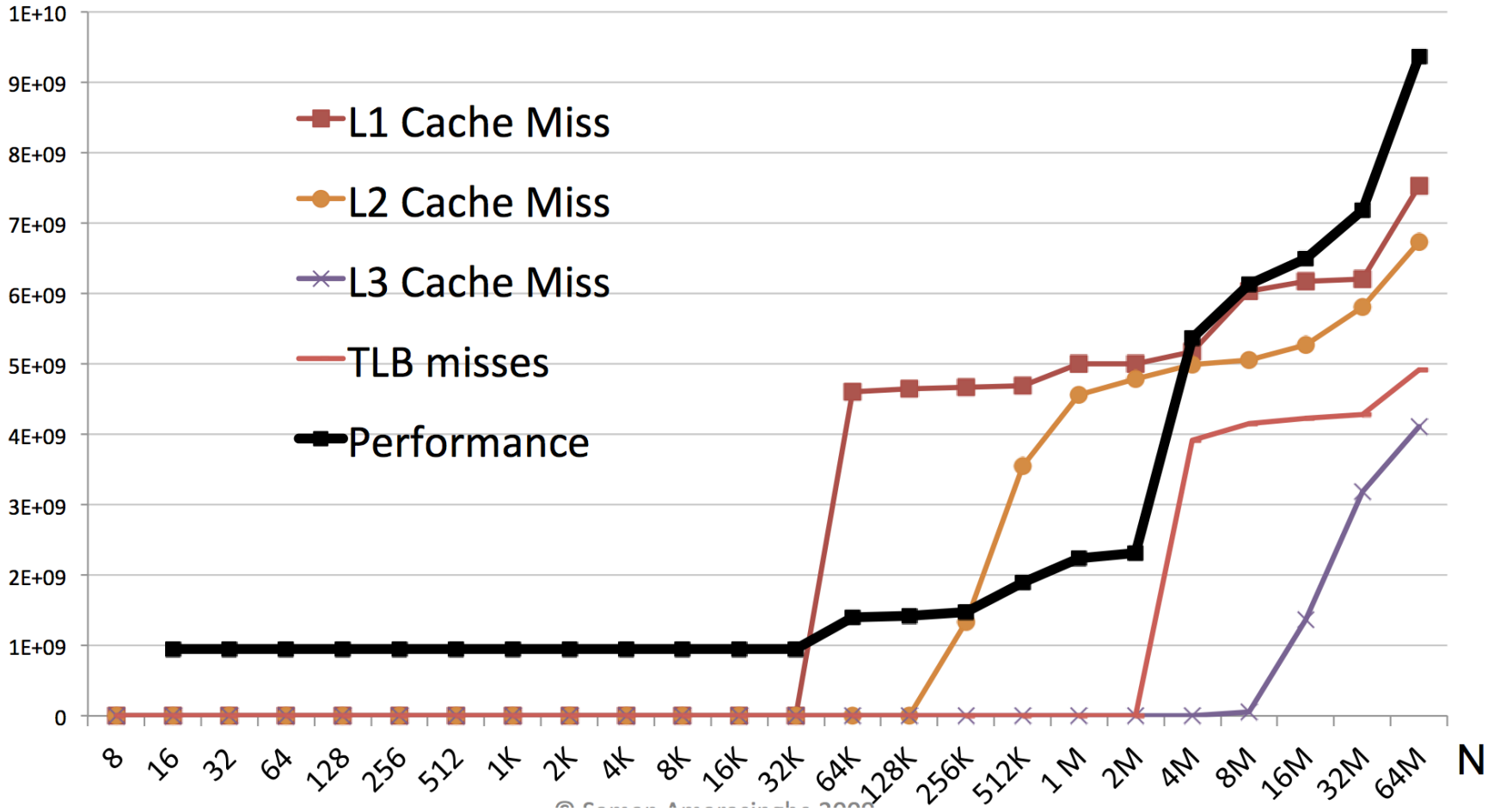
+ Quick & cheap
+ Answers narrow ?s well
– Often misleading results
– Not representative of the program

Microbenchmarking: Blessing & Curse

- Choose your N wisely

Choose Your N Wisely

Prof. Saman Amarasinghe, MIT 2009



Microbenchmarking: Blessing & Curse

- Choose your N wisely
- Measure side effects

Microbenchmarking: Blessing & Curse

- Choose your N wisely
- Measure side effects
- Beware of clock resolution

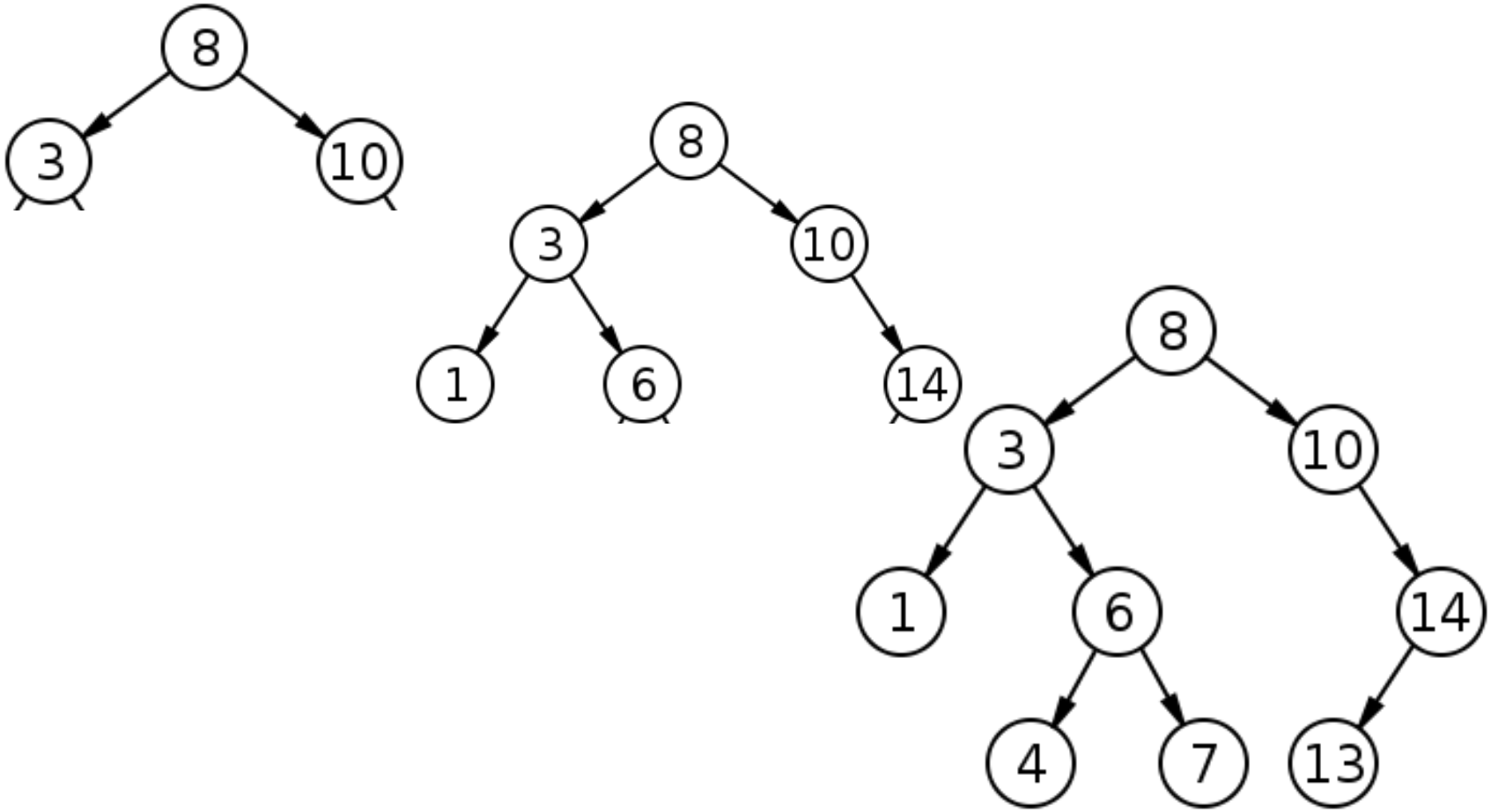
Microbenchmarking: Blessing & Curse

- Choose your N wisely
- Measure side effects
- Beware of clock resolution
- Dead Code Elimination

Microbenchmarking: Blessing & Curse

- Choose your N wisely
- Measure side effects
- Beware of clock resolution
- Dead Code Elimination
- Constant work per iteration

Non-Constant Work Per Iteration



What Should a Benchmark Do?

Measures behavior of system

Represents realistic workload

Runs for sufficiently long time

Compares in the same context

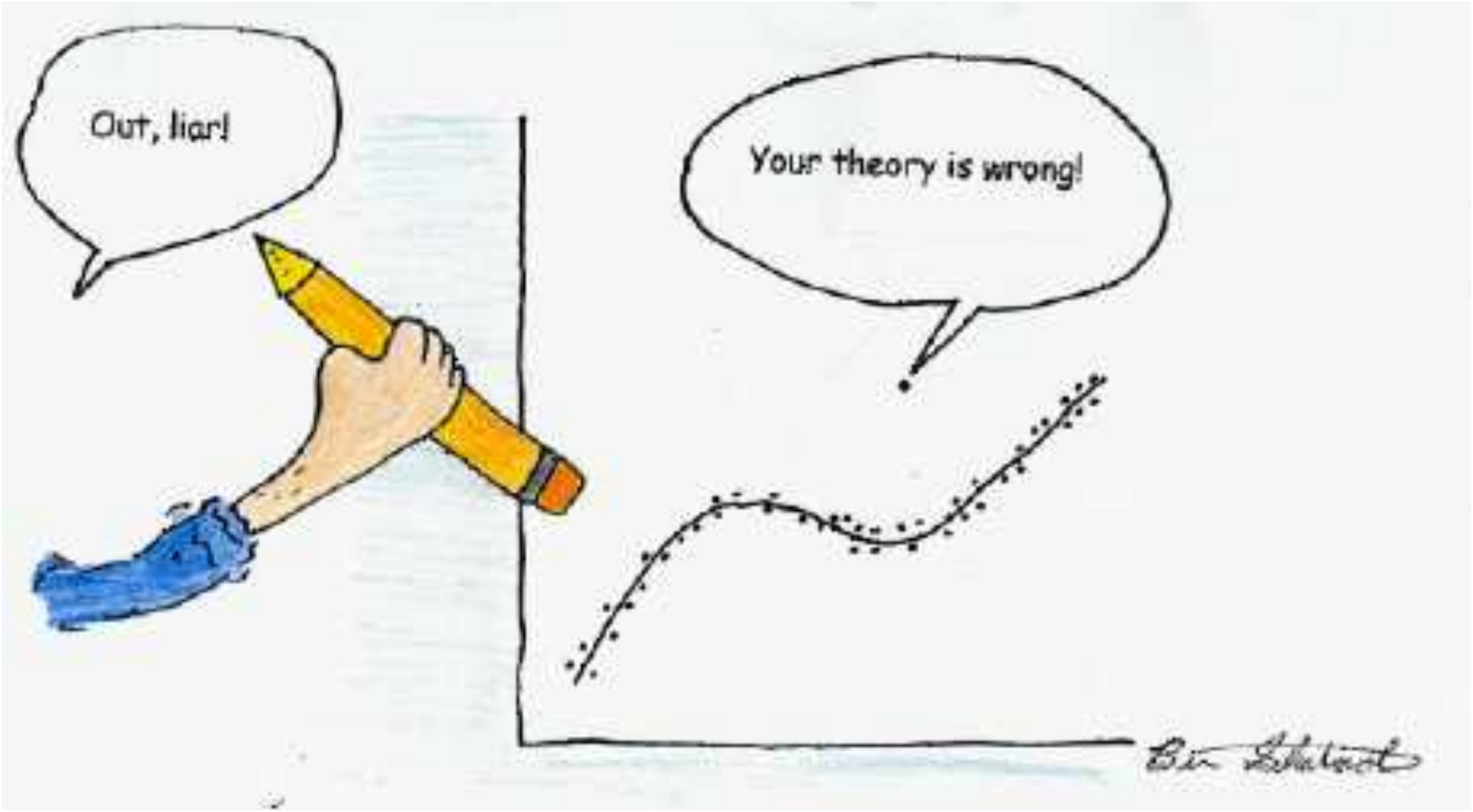
Predictable and reproducible results

Follow-up Material

- *How NOT to Measure Latency* by Gil Tene
  - http://www.infoq.com/presentations/latency-pitfalls
- *Taming the Long Latency Tail* on highscalability.com
  - http://highscalability.com/blog/2012/3/12/google-taming-the-long-latency-tail-when-more-machines-equal.html
- *Performance Analysis Methodology* by Brendan Gregg
  - http://www.brendangregg.com/methodology.html
- *Silverman's Mode Detection Method* by Matt Adereth
  - http://adereth.github.io/blog/2014/10/12/silvermans-mode-detection-method-explained/
- *How Not To Measure System Performance* by James Bornholt
  - https://homes.cs.washington.edu/~bornholt/post/performance-evaluation.html
- *Trust No One, Not Even Performance Counters* by Paul Khuong
  - *http://www.pvk.ca/Blog/2014/10/19/performance-optimisation-~-writing-an-essay/#trust-no-one*

Takeaway #1: Cache

Takeaway #2: Outliers



Takeaway #3: Workload

DEVNEXUS™

Benchmarking:
You're Doing It Wrong

Aysylu Greenberg
@aysylu22

Google