Sandia National Laboratories

MEASURING SAMPLING ERROR FOR CPU PERFORMANCE COUNTERS

Daniel Puckett, Si Chen, Paul V. Gratz, Simon Garcia de Gonzalo, Avani Wildani

Motivation

- CPUs are profiled with *performance counters*
- But, performance counters have *measurement error*
- Prior work in measurement error is old (~2015) or uses toy workloads

Sources of Measurement Error

Example A: Software/Hardware Interaction

- Performance counters are started/stopped by OS
- They can be misaligned with application start/end

Examples from Linux perf

When application starts:

First, application begins

Performance Counter MSRs		
Enabled?	Event Name	Count
No	Cache Misses	0
No	Divide Instrs.	0
No	Branch Instrs.	0

Second, perf starts counting events

Performance Counter MSRs		
Enabled?	Event Name	Count
Yes	Cache Misses	**0**
Yes	Divide Instrs.	**0**
Yes	Branch Instrs.	**0**

Missed events?

When context switch begins or application ends:

First, OS handler begins

Performance Counter MSRs		
Enabled?	Event Name	Count
Yes	Cache Misses	32,543
Yes	Divide Instrs.	781
Yes	Branch Instrs.	45,812

Second, perf stops counting events

Performance Counter MSRs		
Enabled?	Event Name	Count
No	Cache Misses	**32,579**
No	Divide Instrs.	**782**
No	Branch Instrs.	**45870**

Extra events?

When context switch ends:

First, perf starts counting events

Performance Counter MSRs		
Enabled?	Event Name	Count
Yes	Cache Misses	0
Yes	Divide Instrs.	0
Yes	Branch Instrs.	0

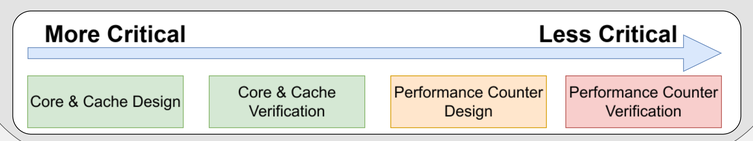
Second, OS passes control back to application

Performance Counter MSRs		
Enabled?	Event Name	Count
No	Cache Misses	**24**
No	Divide Instrs.	**3**
No	Branch Instrs.	**17**

Extra events?

Example B: Design Bugs

- Performance counters are not essential
- On tight schedules, their verification may be incomplete



Quantifying Measurement Error

- Compare performance counter's count to true event count
- Use *dynamic binary instrumentation* to find true event count

Dynamic Binary Instrumentation

Dynamic Instruction Count

Original Binary	
0	add X0, X1, X2
1	sub X0, X3, X0
2	mul X0, X4, X0
...	

Insert instructions to increment dynamic instruction count

Dynamic Multiply Count

Original Binary	
0	add X0, X1, X2
1	sub X0, X3, X0
2	mul X0, X4, X0
...	

Insert instructions to increment dynamic multiply count

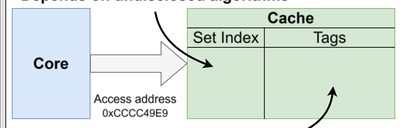
Obstacles to Finding True Event Counts

- Hardware and software unknowns obscure true event count

Hardware Unknowns

Example: Counting number of cache misses

Which set does the access map to?
Depends on undisclosed algorithms



Is the address in the set?
Depends on interference from OS or other applications and undisclosed replacement policies/prefetchers

Software Unknowns

Example: Counting number of dynamic instructions

How many times will the loop iterate?
Depends on network latency

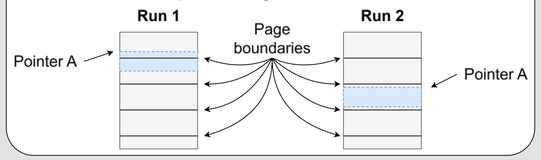
```
0   while (!received data) {
1       check network for data()
2   }
...
8   int* A, B;
9   memcmp(A, B)
```

How many instructions in this highly-optimized standard library call?
Depends on location of pointers in memory!

Finding True Event Counts

- Handle hardware unknowns by measuring events that do not depend on hardware implementation
- Handle software unknowns using consistent, predictable benchmark (SPEC 2017)
- Surprisingly, behavior of SPEC 2017 varies!
- Source of variation is highly-optimized standard libraries that behave differently if arrays cross page boundaries
- Disabling *address space layout randomization* removes variation

Address Space Layout Randomization



Turn off address space layout randomization while benchmarking!

Experiments

- Benchmarks: 7 int-speed SPEC 2017
- CPU: AMD EPYC 7F72
- Events: dynamic instruction count, dynamic conditional branch count
- Perf runs: 20
- PIN runs: 20 (to verify there's no variation)
- Measurement error between perf's count (P) and PIN's true count (T):

$$Error = \frac{P - T}{T}$$

Observed Measurement Error

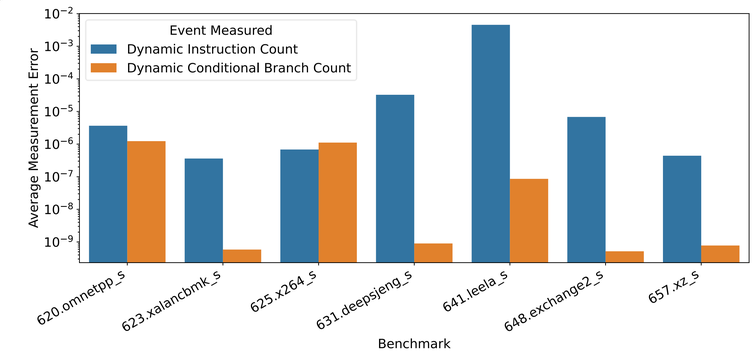


Fig. 3. Average measurement error for each workload and event type.

- Maximum observed measurement error: 0.45%
- Average measurement error typically < 0.01%

Measurement error < 1%.

Variation in Performance Counters

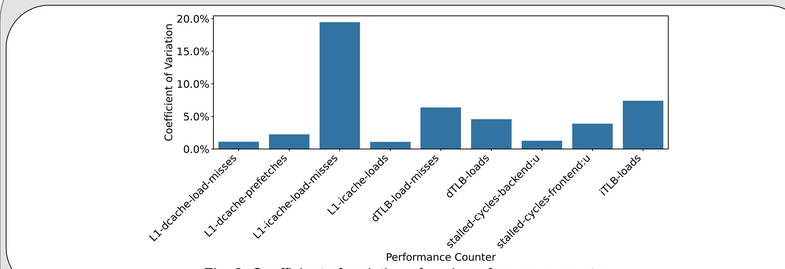


Fig. 2. Coefficient of variation of each performance counter.

- Some performance counters have very large variation
- Since measurement error is small, variation comes from differing workload behaviors
- Characterize these variations!

Conclusion

- Hardware and software unknowns obscure measurement error
- We remove these unknowns and quantify measurement error
- We typically observe < 0.01% measurement error
- Most studies using performance counters can ignore measurement error
- Future plans: more CPUs, argue for variation-aware workload characterization

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.