Measurement Error for CPU Performance Counters

Daniel Puckett
Paul V. Gratz[✉*]
dpuckett98@tamu.edu
pgratz@tamu.edu
Texas A&M University
College Station, Texas, USA

Si Chen
meditates@gmail.com
Uber
Sunnyvale, California, USA

Simon Garcia de Gonzalo
simgarc@sandia.gov
Sandia National Laboratories
Albuquerque, New Mexico, USA

Avani Wildani
agadani@gmail.com
USA

Abstract

Understanding supercomputers' performance and behavior is critical for their optimization and acquisition. Performance counters, which count the number of various types of CPU events, are the primary means of profiling CPUs. However, they may be inaccurate due to measurement errors. We can identify measurement error if we can calculate the true event count, but the true count is obscured by hardware and software unknowns. We handle hardware unknowns by measuring events that do not depend on the hardware implementation, and we handle software unknowns by carefully configuring standardized benchmarks. While we find measuring errors up to 0.45% of the true count, the measuring errors we find are typically much less than 0.01%.

Keywords

CPU, performance counters, measurement error

ACM Reference Format:
Daniel Puckett, Paul V. Gratz, Si Chen, Simon Garcia de Gonzalo, and Avani Wildani. 2026. Measurement Error for CPU Performance Counters. In *Proceedings of Measurement Error for CPU Performance Counters (SC '26).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Summary

CPUs are profiled using performance counters to count various metrics (such as the number of cache misses or divide instructions from an application). Performance counters have measurement error, which may obscure CPUs' true behavior.

Measurement error can come from several sources. For example, performance counters are configured using system calls, so they are not perfectly synced with the application being profiled. Another potential source is design errors. These may occur because performance counters are not essential to the correctness and performance of CPUs, so if a CPU's development has fallen behind schedule performance counters' verification may be left incomplete.

While several previous works have studied measurement error, they use small hand-crafted workloads or processors that are now out-of-date [1, 5–7]. In this work, we study measurement error on full workloads and newer processors.

We can observe measurement error by subtracting the true event count from the performance counters' count. We can measure the true count for many events using dynamic binary instrumentation tools.

However, it is challenging to find the true count for each event due to hardware and software unknowns, which may hinder counting or introduce variation. Hardware unknowns prevent us from fully modeling the internal behavior of CPUs, which is why we cannot use simulators like ChampSim [2] or gem5 [3] to find true event counts. For example, to exactly model a CPU's cache, we must know the cache's set indexing scheme, replacement policy, and prefetchers. These are undisclosed by CPU designers. Software unknowns cause the true event count to change with each run. For example, a workload that sits in a while loop waiting on data for the network will execute different numbers of instructions based on the network latency.

We handle hardware unknowns by measuring two events that do not depend on the hardware implementation: dynamic instruction count and dynamic conditional branch count. We attempt to remove software unknowns by utilizing the SPEC 2017 benchmark suite [4], which is constructed to be reproducible. Surprisingly, these benchmarks do not always execute the same instructions each run because they rely on highly-optimized standard libraries. For example, 620.omnetpp_s and 641.leela_s use the memcmp function from the GNU C library, which executes different instructions based on pointers' location in memory. Pointer locations are randomized each run due to address space randomization, a Linux security feature. We find that turning address space randomization off removes all instruction variation for 7 out of the 10 int-speed SPEC 2017 benchmarks we tested. We use these 7 benchmarks for our evaluations.

We run the 7 SPEC 2017 benchmarks described above 40 times on an AMD EPYC 7F72 CPU while collecting the dynamic instruction and conditional branch counts for each benchmark with address space randomization turned off. We use perf to collect these metrics

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SC '26, Chicago, IL
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

on 20 runs, and PIN, a dynamic binary instrumentation framework, on the other 20 runs. We verify that there is no variation in PIN's counts. We calculate measurement error as the percent difference between perf's count (P) and PIN's true count (T):

$$Error = \frac{P - T}{T} \times 100\%$$

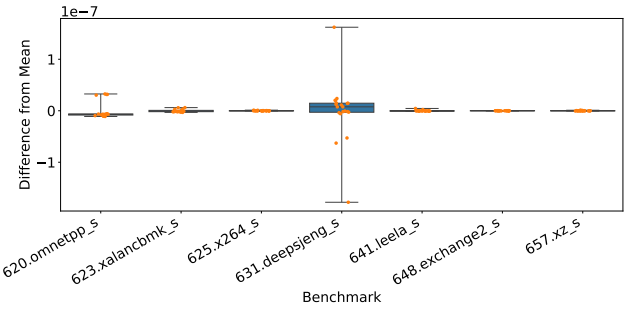


Figure 1: Percent difference from workload mean of dynamic instruction count. Individual points are marked in orange, and the box plots mark the min, quartiles, and max.

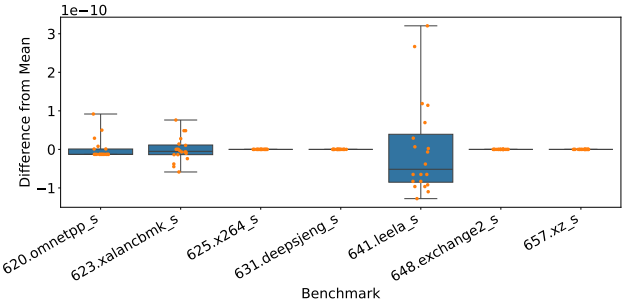


Figure 2: Percent difference from workload mean of dynamic conditional branch count. Individual points are marked in orange, and the box plots mark the min, quartiles, and max.

In Figs. 1 and 2, we plot the variation of perf's dynamic instruction count and dynamic conditional branch count, respectively. Some benchmarks have more variation than others, but the variation is always relatively small.

In Fig. 3, we plot the average measurement error we observe for each workload and event type. The maximum observed measurement error over all our data is 0.45%. However, the average measurement error for each benchmark and event is typically less than 0.01%. The average measurement error is larger than the variation for each workload and event, implying that perf's counts are clustered around the wrong count.

In conclusion, performance counters' measurement error is challenging to observe because the true event count is obscured by hardware and software unknowns. By carefully removing these unknowns, we measure performance counters' measurement error. We observe up to 0.45% measurement error, indicating at least this much error is possible on performance counter measurements.

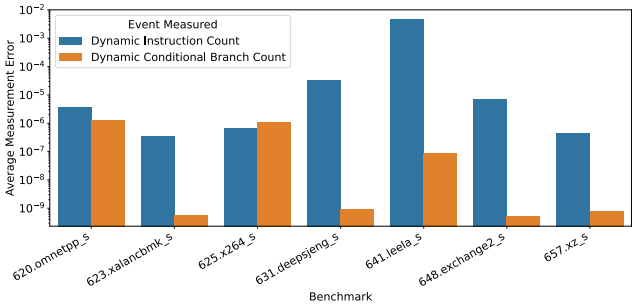


Figure 3: Average measurement error for each workload and event type.

References

[1] Das, S., Werner, J., Antonakakis, M., Polychronakis, M., and Monrose, F. Sok: The challenges, pitfalls, and perils of using hardware performance counters for security. In *2019 IEEE Symposium on Security and Privacy (SP)* (2019), pp. 20–38.

[2] Gober, N., Chacon, G., Wang, L., Gratz, P. V., Jimenez, D. A., Teran, E., Pugsley, S., and Kim, J. The championship simulator: Architectural simulation for education and competition, 2022.

[3] Lowe-Power, J., Ahmad, A. M., Akram, A., Alian, M., Amslinger, R., Andreozzi, M., Armejach, A., Asmussen, N., Beckmann, B., Bharadwaj, S., Black, G., Bloom, G., Bruce, B. R., Carvalho, D. R., Castrillon, J., Chen, L., Derumigny, N., Diestelhorst, S., Elsasser, W., Escuin, C., Fariborz, M., Farmahini-Farahani, A., Fotouhi, P., Gambord, R., Gandhi, J., Gope, D., Grass, T., Gutierrez, A., Hanindhito, B., Hansson, A., Haria, S., Harris, A., Hayes, T., Herrera, A., Horsnell, M., Jafri, S. A. R., Jagtap, R., Jang, H., Jeyapaul, R., Jones, T. M., Jung, M., Kannoth, S., Khaleghzadeh, H., Kodama, Y., Krishna, T., Marinelli, T., Menard, C., Mondelli, A., Moreto, M., Mück, T., Naji, O., Nathella, K., Nguyen, H., Nikoleris, N., Olson, L. E., Orr, M., Pham, B., Prieto, P., Reddy, T., Roelke, A., Samani, M., Sandberg, A., Setoain, J., Shingarov, B., Sinclair, M. D., Ta, T., Thakur, R., Travaglini, G., Upton, M., Vaish, N., Vougioukas, I., Wang, W., Wang, Z., Wehn, N., Weis, C., Wood, D. A., Yoon, H., and Éder F. Zulian. The gem5 simulator: Version 20.0+, 2020.

[4] Spec cpu2017, 2025. https://www.spec.org/cpu2017.

[5] Weaver, V. M., and McKee, S. A. Can hardware performance counters be trusted? In *2008 IEEE International Symposium on Workload Characterization* (2008), pp. 141–150.

[6] Weaver, V. M., Terpstra, D., and Moore, S. Non-determinism and overcount on modern hardware performance counter implementations. In *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* (2013), pp. 215–224.

[7] Zaparanuks, D., Jovic, M., and Hauswirth, M. Accuracy of performance counter measurements. In *2009 IEEE International Symposium on Performance Analysis of Systems and Software* (2009), pp. 23–32.