# BayesPerf: Minimizing Performance Monitoring Errors Using Bayesian Statistics
## Extended Abstract

Subho S. Banerjee, Saurabh Jha, Zbigniew T. Kalbarczyk, and Ravishankar K. Iyer
University of Illionis at Urbana-Champaign

## 1. Motivation

Hardware performance counters (HPCs) are widely used in profiling applications to characterize and find bottlenecks in application performance. Even though HPCs can count hundreds of different types of architectural and microarchitectural events, they are limited because those events are collected (i.e., multiplexed) on a fixed number of hardware registers (usually 4–10 per core). As a result, they are error-prone because of application, sampling, and asynchronous collection behaviors that are a result of multiplexing. Such behavior in HPC measurements is not a new problem, and has been known for the better part of a decade [1, 8, 16, 17, 22, 23, 24].

This paper presents BayesPerf, a system for quantifying uncertainty and correcting errors in HPC measurements using a Bayesian model that captures microarchitectural relationships between HPCs. BayesPerf corrects HPC measurement errors at the system (i.e., CPU and OS) level, thereby allowing the down stream control and decision models that use HPCs to be simpler and faster and use less training data when used with machine learning (ML). The proposed model is based on the insight that even though individual HPC measurements might be in error, groups of different HPC measurements that are related to one another can be jointly considered) to reduce the measurement errors (using the underlying statistical relationships between the HPC measurements. We derive such relationships by using design and implementation knowledge of the microarchitectural resources provided by CPU vendors [4, 13]. For example, the number of LLC misses, the number and size of DMA transactions, and the DRAM bandwidth utilization are related quantities,[1] and can be used to reduce measurement errors in each other.

## 2. Limitations of the State of the Art

Traditional approaches of tackling HPC errors have relied on collecting measurements across several application runs, and then performing offline computations to (i) impute missing or erroroneous measurements with new values (e.g., [21, 23]); or (ii) drop outlier values to reduce overall error (e.g., [16]). The first group of methods artificially imputes data in the collected samples by interpolating between two sampled events using linear or piece wise linear interpolation, e.g., the default in Linux [21]. The advantage of such interpolation methods is that they can be run in real time: however, they might not provide good imputations [24]. The second group of methods correct measurements by dropping outlier values, instead of adding new interpolated values. Such methods are at the other extreme: they cannot be run in real time, as they need the entire trace of an application before they provide corrections. They are untenable in emergent applications that use HPCs as inputs to complete a feedback loop (using control-based or ML-based techniques) and make dynamic real-time decisions that affect system resources. Examples include online performance hotspot identification (e.g., [10]), userspace or runtime-level scheduling (e.g., [2, 3, 7, 11, 24]), and power and energy management (e.g., [9, 18, 19, 20]), as well as attack detectors and system integrity monitors [5]. In such cases, the HPC measurement errors propagate, get exaggerated, and can lead to *longer training time* and *poor decision quality*. That is unsurprising because ML systems are known to be sensitive to small changes in their inputs (e.g., in adversarial ML) [6, 12, 14]. As we show in the paper, HPC measurement errors can be large (as much as 58%); hence, they must be explicitly handled.

[1] DRAM Bandwidth = (LLC missed × Cache line size+ # DMA Transactions × Transaction size)/Clocks.

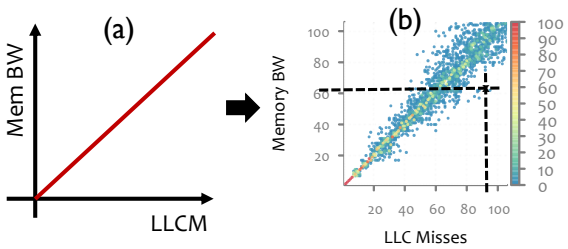


**Figure 1: Quantifying Uncertainty in HPC Measurements**

## 3. Key Insights

- *Probabilistic Error Detection:* The critical insight that drives this work is that microarchitectural invariants can be applied to measured HPC data to estimate whether the data are, in fact, erroroneous. Consider a toy example in which two performance counters, `DRAM Bandwidth (DB)` (from a memory controller) and `LLC Cache Misses (LLCM)`, are counted. Assuming there is no DMA traffic, the HPCs should always be related through the relation $\texttt{DB} = \texttt{LLCM} \times {}^{\texttt{CacheLineSize}}/_{\texttt{Interval}}$, where `Interval` is the time over which the bandwidth is computed (shown in Fig. 1(a)). When presented with data that do not fit that model (i.e., do not satisfy the equation), we can be certain that there are measurement errors. To quantify the magnitude of the error, we assume that the two HPCs are random variables (real values with added noise/error), and we calculate the probability of deviation from the above equation. The uncertainty of the measured data is illustrated in Fig. 1(b) as the color bar, which represents the likelihood

of occurrence.

- *Defining Invariants:* Further, the invariants described above can have transitive relationships, depending on the microarchitectural component of the system they are profiling. In the example, a count of L1/L2 misses can provide statistical support for the values measured for `LLCM`, i.e., we know that correct values of the counters would satisfy `L1M` $\geq$ `L2M` $\geq$ `LLCM`. Those microarchitecture-dependent invariants are automatically extracted from vendor-provided manuals [4, 13] or from machine-parsable JSON files in the Linux source tree [21]. BayesPerf expresses the composition of all such invariants as a joint probability distribution over all HPC measurements of interest and uses the popular *probabilistic graphical model* formalism to infer the measurement uncertainty.
- *Inferring Correct Value (Error Correction):* We finally use Bayesian inference and maximum likelihood estimation approach to integrate the data and prior knowledge (i.e., statistical relationships) of the system to attenuate the highly erroneous measurements (that cause high uncertainty) effectively and significantly amplify correct measurements (that results in low uncertainty) in real-time. That is the reason we are able to significantly outperform traditional, purely data-driven statistical approaches for outlier detection.

## 4. Main Artifacts

- A Bayesian-ML model, training, and inference strategies for detecting and correcting HPC errors.
- A prototype implementation of a system that integrates the ML model into the Linux `perf_event` subsystem. It includes a design for an FPGA-based accelerator to enable real-time inference of the proposed ML model.
- A benchmark application based on Apache Spark that uses BayesPerf-corrected HPC measurements in a control loop to actuate PCIe transfers.
- The artifacts were evaluated on two real systems, including x86 (Intel BroadwellX) and ppc64 (IBM Power 9) CPUs, using a Xilinx Ultrascale+ FPGA.

## 5. Key Results and Contributions

- *The BayesPerf ML Model.* We present a probabilistic ML model that incorporates microarchitectural domain knowledge to combine measurements from several noisy HPCs to infer their true values, as well as quantify the uncertainty in the inferred values due to measurement noise. The model enables:
  1. Decision-making with explicit quantification of HPC measurement uncertainty.
  2. Reduced need for aggressive (high-frequency) HPC sampling (which negatively impacts application performance) to capture high-fidelity measurements, thereby increasing the system's observability.

  Although we demonstrate the model for a CPU-based HPC, the algorithm is generic and not limited to CPUs. It can be extended for any hardware-counter-based measurements (e.g., from the Berkeley Packet Filter), which are becoming an integral part of measurements in Linux systems.
- *The BayesPerf Accelerator.* To enable use of the BayesPerf ML model in latency-critical, real-time decision-making tasks, this paper presents the design and implementation of an accelerator for Monte Carlo-based training and inference of the BayesPerf model. The accelerator exploits:
  1. High-throughput random-number generators.
  2. Maximal parallelism to rapidly sample conditionally independent parts of the model.
- *A Prototype Implementation.* We describe an FPGA-based prototype implementation of the BayesPerf system (on a Xilinx Virtex 7 FPGA) on a Linux-based Intel x86 Sky Lake and IBM Power9 processor. The BayesPerf system is designed to provide API compatibility with Linux's `perf` subsystem [15], allowing it to be used by *any* userspace performance-monitoring tool for both x86 and ppc64 systems. Our experiments demonstrate that BayesPerf reduces the average error in HPC measurements by as much as 5.39× (i.e., from 42.11% to 7.8%) when events are being multiplexed. The BayesPerf accelerator provides an 11.8× reduction in power consumption, while adding less than 2% read latency overhead over native HPC sampling.
- *Increasing Training and Model Efficiency of Decision-making Tasks.* We demonstrated the generality of the BayesPerf system by integrating it with a high-level ML-based IO scheduler that controls transfers over a PCIe interconnect. We observed that the training time for the scheduler was reduced by 34% and the average makespan of scheduled workloads decreased by 19%.

## 6. Why ASPLOS?

This paper describes multi disciplinary research incorporating innovations in:

- *ML* (i.e., the BayesPerf model and inference strategy);
- *Operating Systems* (i.e., the transparent integration of the BayesPerf model into Linux, allowing it to be used by any process capable of using profiling data); and
- *Architecture* (i.e., the design and implementation of the BayesPerf accelerator to enable real-time inference).

ASPLOS is hence an ideal venue for showcasing this research.

## 7. Citation for Most Influential Paper Award

The BayesPerf system represents innovative work for detecting and correcting errors in hardware performance counters. The system is workhorse for a wide range of measurement-driven feedback techniques (like those that use control theory or ML to control system resources) for which low-variance, low-noise measurement data are critical for making accurate predictions, thereby making those models robust, efficient, and easier (faster) to train.

## References


[1] Glenn Ammons, Thomas Ball, and James R. Larus. Exploiting hardware performance counters with flow and context sensitive profiling. *SIGPLAN Not.*, 32(5):85–96, May 1997.

[2] Subho S. Banerjee, Saurabh Jha, and Ravishankar K. Iyer. Inductive bias-driven reinforcement learning for efficient schedules in heterogeneous clusters. *CoRR*, abs/1909.02119, 2019.

[3] Andrew Baumann, Paul Barham, Pierre-Evariste Dagand, Tim Harris, Rebecca Isaacs, Simon Peter, Timothy Roscoe, Adrian Schüpbach, and Akhilesh Singhania. The multikernel: A new os architecture for scalable multicore systems. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, pages 29–44, New York, NY, USA, 2009. ACM.

[4] Intel Corp. Intel® 64 and IA-32 Architectures Software Developer Manuals. https://software.intel.com/en-us/articles/intel-sdm, 2016. Accessed 2019-03-05.

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

[6] Pritam Dash, Mehdi Karimibiuki, and Karthik Pattabiraman. Out of control: Stealthy attacks against robotic vehicles protected by control-based techniques. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, page 660–672, New York, NY, USA, 2019. Association for Computing Machinery.

[7] Christina Delimitrou and Christos Kozyrakis. Paragon: Qos-aware scheduling for heterogeneous datacenters. In *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '13, pages 77–88, New York, NY, USA, 2013. ACM.

[8] M. Dimakopoulou, S. Eranian, N. Koziris, and N. Bambos. Reliable and efficient performance monitoring in linux. In *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 396–408, Nov 2016.

[9] Yi Ding, Nikita Mishra, and Henry Hoffmann. Generative and multi-phase learning for computer systems optimization. In *Proceedings of the 46th International Symposium on Computer Architecture*, ISCA '19, pages 39–52, New York, NY, USA, 2019. ACM.

[10] Yu Gan, Yanqi Zhang, Kelvin Hu, Dailun Cheng, Yuan He, Meghna Pancholi, and Christina Delimitrou. Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, pages 19–33, New York, NY, USA, 2019. ACM.

[11] Jana Giceva, Gustavo Alonso, Timothy Roscoe, and Tim Harris. Deployment of query plans on multicores. *Proc. VLDB Endow.*, 8(3):233–244, November 2014.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[13] Brian Hall, Peter Bergner, Alon Shalev Housfater, Madhusudanan Kandasamy, Tulio Magno, Alex Mericas, Steve Munroe, Mauricio Oliveira, Bill Schmidt, Will Schmidt, et al. *Performance optimization and tuning techniques for IBM Power Systems processors including IBM POWER8*. IBM Redbooks, 2017.

[14] Saurabh Jha, Shengkun Cui, Subho S Banerjee, Timothy Tsai, Zbigniew T Kalbarczyk, and Ravishankar K Iyer. ML-driven Malware for Targeting AV Safety. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2020.

[15] Linux Community. perf: Linux profiling with performance counters. https://perf.wiki.kernel.org/index.php/Main_Page, 2019. [Online; accessed 19-November-2019].

[16] Y. Lv, B. Sun, Q. Luo, J. Wang, Z. Yu, and X. Qian. Counterminer: Mining big performance data from hardware counters. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 613–626, Oct 2018.

[17] T. Mytkowicz, P. F. Sweeney, M. Hauswirth, and A. Diwan. Time interpolation: So many metrics, so few registers. In *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*, pages 286–300, Dec 2007.

[18] R. P. Pothukuchi, S. Y. Pothukuchi, P. Voulgaris, and J. Torrellas. Yukta: Multilayer resource controllers to maximize efficiency. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 505–518, June 2018.

[19] Raghavendra Pradyumna Pothukuchi, Joseph L. Greathouse, Karthik Rao, Christopher Erb, Leonardo Piga, Petros G. Voulgaris, and Josep Torrellas. Tangram: Integrated control of heterogeneous computers. In *Proceedings of the 52Nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '52, pages 384–398, New York, NY, USA, 2019. ACM.

[20] Stephen J. Tarsa, Rangeen Basu Roy Chowdhury, Julien Sebot, Gautham Chinya, Jayesh Gaur, Karthik Sankaranarayanan, Chit-Kwan Lin, Robert Chappell, Ronak Singhal, and Hong Wang. Post-silicon cpu adaptation made practical using machine learning. In *Proceedings of the 46th International Symposium on Computer Architecture*, ISCA '19, pages 14–26, New York, NY, USA, 2019. ACM.

[21] Linus Torvald. Linux Perf Subsystem Userspace Tools. https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/tools/perf/pmu-events/arch, 2020. Accessed 2020-03-05.

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

[23] Vincent M Weaver and Sally A McKee. Can hardware performance counters be trusted? In *2008 IEEE International Symposium on Workload Characterization*, pages 141–150. IEEE, 2008.

[24] Gerd Zellweger, Denny Lin, and Timothy Roscoe. So many performance events, so little time. In *Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems*, APSys '16, pages 14:1–14:9, New York, NY, USA, 2016. ACM.