# Revisiting Performance Evaluation in the Age of Uncertainty

Pedro Bruel
*Hewlett Packard Labs, USA*
bruel@hpe.com

Viyom Mittal
*Hewlett Packard Labs, USA*
viyom.mittal@hpe.com

Dejan Milojicic
*Hewlett Packard Labs, USA*
dejan.milojicic@hpe.com

Michalis Faloutsos
*University of California, Riverside*
michalis@cs.ucr.edu

Eitan Frachtenberg*
*Hewlett Packard Labs, USA*
eitan.frachtenberg@hpe.com

***Abstract*—Given a cloud-native application, how do we accurately estimate its performance, such as run time or memory consumption? Accurate estimation is necessary to ensure that the application meets performance goals without resorting to overprovisioning of resources. Additionally, in practice, performance estimation needs to be meaningful and reproducible. Unfortunately, modern HPC systems come with numerous factors affecting performance estimation, such as heterogeneous accelerators, multilevel networks, millions of cores, layered software abstractions, and specialized middleware. Each of these factors adds a degree of variability to empirical performance results.**

**The approaches currently being taught and practiced limit performance evaluation in three ways: (1) usage of incomplete performance descriptions/metrics such as point summaries (e.g., mean, 99th-percentile or median) which hide the rich behavioral patterns in different scenarios; (2) measuring insufficient performance samples, leading to inaccurate performance description; and (3) measuring excessive performance samples, leading to waste of precious computing resources.**

**To overcome these limitations, we propose a new approach to evaluate and reason about application performance in modern HPC in a meaningful way. Our contribution is threefold: (a) we show the difficulty of estimating performance in realistic scenarios: one performance measurement is not enough; (b) we propose to use distributions as the true measure of performance; and (c) we propose several practices and concepts to be taught to HPC students and practitioners, so that they may produce rich and accurate performance evaluations. We see our work having an impact both on educators and on practitioners.**

***Index Terms*—high-performance computing (HPC), performance evaluation, computer-science education, benchmarking, curriculum design**

## I. Introduction

Estimating the performance of a complex computing system is a key capability in the field of high-performance computing (HPC). The HPC literature is filled with countless papers that measure, model, or optimize performance, and its practitioners develop, use, and report on countless performance benchmarks [1]. One of the primary aims of these works is to perform accurate and reproducible reporting. Unfortunately, many of these studies report performance in ways that are incomplete, inaccurate, incorrect, or irreproducible [2], [3], [4].

Estimating the performance of modern HPC systems has become very challenging. First, the performance of modern computing systems is determined by many interacting (and sometimes interfering) factors that may appear arbitrary at times [5], [6]. Sources of nondeterministic performance range from architecture (e.g., proprietary speculative execution algorithms), to operating systems and middleware (e.g., context switches), workloads (e.g., background jobs), benchmarks (e.g., randomized algorithms and Monte-Carlo simulations), and even the environment (e.g., datacenter temperature affecting CPU throttling). Consequently, benchmarks that produce performance summaries subject to these combined random variables can create an incomplete or inaccurate characterization of the system's performance.

Second, capturing and describing the performance accurately can be complicated. HPC systems are particularly challenging to summarize in a few performance numbers because they often include more "moving parts" that result in more variable performance [7]. Examples of additional sources of nondeterminism in HPC include extensive use of independent hardware accelerators, communication and synchronization across processes and nodes, tighter resource contention, and increasingly, cloud computing. [8], [9], [10], [11].

To illustrate this point, let us consider the example of the most famous of HPC performance reports, the biannual Top-500 list [12]. For each of the largest 500 HPC systems in the world, the list reports and ranks performance on a single benchmark using just a handful of peak metrics. Aside from the fact that the list ignores many aspects relevant to usability such as availability, reliability, and cooling requirements, it even represents the performance picture itself in a woefully incomplete view. The list only presents the performance characteristics of

*Corresponding author

Linpack and ignores all the other applications. The report also fails to share details on expected performance, performance scalability, the number of runs it took to measure the optimal performance, or the variability of performance across hardware, middleware, and software parameters. Trying to describe a supercomputer's performance by concentrating on Linpack's Rpeak and Rmax alone is akin to the parable of the six blind men and the elephant: while some point summaries are certainly of interest, they all run the risk of missing the big picture or the typical behavior. These problems in performance evaluation description are not new and there exists a rich literature on the topic [9], [4], [1].

In this paper, we propose a fundamental shift in the way we teach and think of performance evaluation in general, and for HPC systems in particular, going from "performance as a number" to "performance as a distribution". Namely, we propose a way to describe the expected performance of an HPC system by centering the reporting on the performance distribution and moving away from a small set of numbers. We posit that the target of performance evaluation should be the accurate and expedient measurement of the performance distribution, rather than its summary. When properly measured and reported, a distribution can then be more profoundly understood, compared, reproduced, and even summarized as appropriate in each context. In a nutshell, what we propose is a shift of perspective in the way we teach performance evaluation in general, and for HPC in particular, going from "performance as a number" to "performance as a distribution".

To motivate this position, we start with a brief summary of the challenges of performance summaries, followed by motivating examples from actual empirical HPC benchmarks from the Rodinia suite. In Section IV, we present our proposed approach for educators and practitioners. Section V then discusses some pedagogical considerations for our proposed approach, followed by practical recommendations in Section VI. Finally, we conclude with a call to action in Section VII.

## II. Pitfalls of performance summaries

Of the myriad methodological concerns and challenges in performance evaluation, this paper is focused on one area, namely, the characterization of performance. The following list summarizes some of the main reasons why not capturing the full performance distribution can be problematic at times. This list is by no means exhaustive, and a more detailed discussion of pitfalls and workarounds can be found in the literature [3], [13], [14].

### A. Wrong summary

Just like the cliche about a person drowning in a one-inch-deep (on average) swimming pool, the mean, median, and other summaries can be of limited usefulness in certain scenarios [14]. For example, a sample mean isn't that descriptive for distributions that are multimodal, autocorrelated, or long-tailed (which are common for many real-world performance metrics such as latency). Even adding standard deviation and correlations may not suffice to describe the distribution in adequate detail.[1] Complex distributions require capturing more summary statistics, such as median, modes, standard deviation, confidence interval for the mean, kurtosis, and others.

### B. Wrong model

Even when capturing additional variables, performance summaries could be misleading if they are based on unrepresentative models of the data. For example, one may report the confidence interval (CI) around the mean, which would be reasonably descriptive for a Gaussian distribution. But if the underlying distribution is lognormal for example, the CI computed under assumptions of normality and symmetry would be wrong; and if the distribution is bimodal, the CI may be better replaced by a noncontiguous high-density (HDI) Bayesian interval.

### C. Ignoring important outliers

Outliers present a thorny challenge for experimenters. What do you do with values that fall so far away from the expected values that they skew the distribution? Do you include them in the summary? Do you ignore them? Do you winsorize them? Do you investigate them? Unfortunately, there is no simple answer. Outliers are highly dependent on context, and there is a whole science and art to their proper treatment [15]. However, capturing and describing entire distributions instead of summaries skirts this problem, since outliers are part of the distribution. The informed decision of how to treat them can then be made in the specific context and requirements of each interpretation.

### D. Not enough samples

Another way that plotting and describing full distributions can help the experimenter is in deciding how many samples (repetitions) are required to get an accurate performance metric. Many experimenters pick an arbitrary number of samples with the hope that it is "large enough" (sometimes as low as 1!). Others rely on rules of thumb such as "30 samples are enough for the central limit theorem," which are both unproven for Gaussian data, and simply wrong for other data [16].

Understanding the distribution of the underlying data can help in choosing an appropriate stopping rule in two ways. First, it can justify a static (or better yet, dynamic) stopping rule that is tailored for the type of distribution, and there are many to choose from. Second, if we know or can afford to measure an accurate description of the "ground truth" distribution of the data, then we can also

[1] for an illustrative example, see https://blog.revolutionanalytics.com/2017/05/the-datasaurus-dozen.html

TABLE I
All benchmarks and their parameters (under submission in a related workshop).

| Benchmark | Parameters |
|---|---|
| backprop | 6553600 |
| bfs | 4, graph1MW_6.txt |
| heartwall | test.avi, 20, 4 |
| hotspot | 1024, 1024, 2, temp_1024, power_1024 |
| kmeans | 4, kdd_cup |
| lavaMD | 4, 10 |
| leukocyte | 5, 4, testfile.avi |
| lud | 8000 |
| needle | 20480, 10, 2 |
| sc | 10, 20, 256, 65536, 65536, 1000, none, 4 |
| srad | 1000, 0.5, 502, 458, 4 |
| backprop-CUDA | 955360 |
| bfs-CUDA | graph1MW_6.txt |
| heartwall-CUDA | test.avi, 100 |
| hotspot-CUDA | 512, 2, 2, temp_512, power_512 |
| needle-CUDA | 10240, 10 |
| sc-CUDA | 10, 20, 256, 65536, 65536, 1000, none, 1 |
| srad-CUDA | 100000, 0.5, 502, 458 |

compute the minimum sample size required to approximate this distribution to a desired distance threshold, as described in Section III.

### *E. Too many samples*

A related problem is that our stopping criterion may be too restrictive, leading to the measurement of unnecessary samples with little effect on the empirical distribution or its summary. This problem is more benign, in the sense that it does not lead to inaccurate results, only to wasted resources. But there are situations in which these resources are scarce (e.g., time on the latest supercomputer). Again, understanding the underlying distribution can lead to better stopping rules.

These problems are not just theoretical, nor do they affect only novices. They are commonly present in classrooms and the scientific literature, even when following standard practices of measurement on standard benchmarks, as demonstrated in the next section.

## III. Empirical Examples

As a motivating example, we ran benchmarks from the Rodinia HPC benchmark suite 1,000 times to explore their run time distributions [17]. CPU-only benchmarks used the OpenMP library and GPU benchmarks used the CUDA library, using the command-line parameters listed in Table I.

The first thing to note about these distributions is how diverse they are, which makes accurate summaries more difficult. Some benchmarks appear multimodal (hotspot, kmeans) and some are clearly unimodal (lud, needle); Some show pronounced concentrations on a single value (bfs and lavaMD), while others are more spread out, despite being modal (leukocyte, sc); and some have tails within a narrow range of ±20% around the center (hotspot), while others spread to around ±70% (needle). Moreover, all of the pitfalls described in the previous section can be exemplified in this set, as follows.

**Wrong summary**: Many of these histograms are long-tailed and asymmetric, so picking the mean run time to stand for an entire distribution is clearly not representative of either the most common cases, or the extreme ones. All of the Rodinia benchmarks exhibit a strong mode (which is not necessarily true of other benchmarks), so one may surmise that summarizing the distribution with this value would capture the most common performance. Perhaps so, but it would miss the fact that several benchmarks (especially running on the GPU) have more than one prominent mode, suggesting more than one configuration performance, which would be important to capture and understand.

**Wrong model**: Since some of these distributions are long-tailed, trying to describe their variance with a Gaussian confidence interval would be inappropriate. Even taking the log-transform for the long-tailed distributions would not always be appropriate, since a few of them are left-tailed (e.g., hotspot).

**Ignoring outliers**: The results for needle are fairly clustered, with the mean, median, and mode coinciding within a narrow range of around 2.75s. But over 13% of the measurements exceed 3s, some reaching almost 12s. Important information about this not-uncommon case would be lost if we only summarized the distribution with these three statistics.

**Not enough samples**: Suppose one still insists on summarizing the distributions with mean performance (perhaps because they only care about expected run time, not variability). Suppose further that the average is taken after a fixed number of samples, as is common in many systems studies. A sample size of 10, for example, would approximate the mean to within 2% or so for most benchmarks, but some, like heartwall, would still be 8% off. Even a sample size that is adequate for a particular benchmark on the CPU can fail for the same benchmark on the GPU. For example, the first 10 samples of bfs compute a mean that is only 2.1% off from the full 1,000, but for bfs_gpu, the error is much larger (6.6%).

**Too many samples**: A counter-reaction to this finding might be to employ an overly conservative approach. A fixed sample size of 100 would approximate all the means to within 3% of the full sample size. But for most benchmarks tested, this represents a waste of 90% or more of the time and compute resources required to reach this precision level.

To illustrate further the last two points, let us see why finding a "good" single sample size for the entire suite is infeasible. We could choose some distance criterion to decide when one empirical distribution approximates a "true" distribution closely enough, and try to derive an adequate number of samples from that. In this example, we'll use the Kolmogorov-Smirnov distance metric between distributions with some arbitrary closeness thresh-

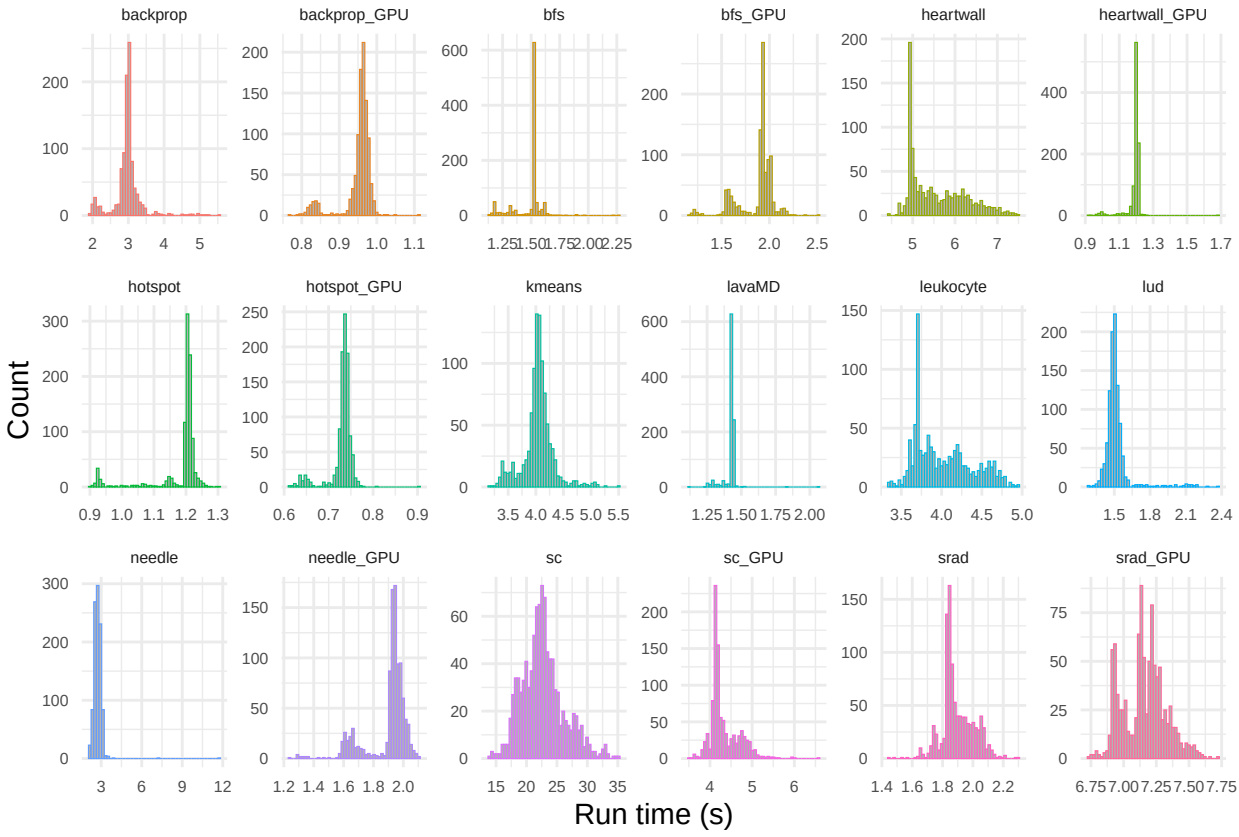


Fig. 1. Performance histograms for repeated runs of different Rodinia benchmarks on CPU and GPU

old (say, 0.1), and compute how many sequential samples each benchmark requires to approximate the full 1,000-sample set. The results of the computation vary by an order of magnitude, requiring as few as 41 samples for heartwall_GPU and as many as 690 for srad_GPU (mean: 150.2, SD: 163.5). Any fixed sample size chosen for all these benchmarks would be inadequate in one way or another for most benchmarks.

The upshot of all these experiments is that a description of HPC performance with only summary statistics or using fixed sample sizes may in some cases be inaccurate, hard to contextualize, impossible to generalize, or even plain wrong. Let us next turn our attention to an educational approach to remedying this common practice.

## IV. Proposed approach

As the previous example illustrated, different performance benchmarks and configurations exhibit very different distributions. The practical implication here is that metrics that summarize the distribution, like mean and median, cannot work equally well in all situations. Furthermore, sampling methods that estimate these metrics, like a fixed sample size or a stopping rule based on normality assumptions share the same weakness of non-generality under different distributions. The solution we propose is to treat performance as a distribution and treat distributions as first-class citizens in the evaluation. The focus of the performance evaluation then shifts from succinctly describing a distribution to measuring, understanding, and reporting it correctly. Let us examine all three aspects.

On the measuring front, there are numerous methodological issues in performance evaluation that have been studied extensively. Examples include obtaining a sufficient (but not excessive) number of samples, possibly using dynamic stopping rules; hypothesis testing to validate convergence of the results, and sensitivity analysis for all salient factors. The difference in our approach is that we are advocating the use of careful methodology not to estimate any particular statistic accurately, but rather to estimate the underlying distribution accurately. We have also recently proposed an adaptive stopping rule that can dynamically adjust the observed distribution at measurement time and heuristically pick the most appropriate stopping criteria for the data [18].

On the understanding front, the focal shift to distributions means that we need a broader statistical toolbox. For example, when evaluating two sample sets, we might be comparing two means with a t-test (if we can justify normality), or two medians with a Wilcoxon signed-rank test. But when comparing whole distributions,

perhaps to estimate how far an empirical distribution is from the ground-truth distribution, or how a change of a parameter affects the performance distribution, we need other tools. There are various methods to compare distributions, including t-test, Jensen-Shannon, Kullback-Leibler, Kolmogorov-Smirnov, Population Stability Index, and others. A thorough understanding of their differences and subtleties is beyond the scope of this paper but is advisable for a more informed perspective on the experimental artifacts. In our experience, the Kolmogorv-Smirnov test (KS) works well in practice for performance distributions because it is a metric (fulfills the triangle inequality), symmetric, nonparametric, relatively stable, and not prone to division by zero [19]. Obviously, manipulating and understanding distributions requires more than a rudimentary grasp of statistics, so this toolbox must also include familiarity with a spectrum of statistical distributions, their properties, and their manifestations in actual performance scenarios. For example, it is highly unlikely in computer experiments that any distribution of empirical data mirrors exactly any analytical distribution. Expect surprises when comparing distributions graphically and using statistical methods. For example, an outlier does not necessarily imply a long tail or non-normality, and divergences from analytical forms can be the product of hidden variables or intrinsic measurement errors. This does not mean we should always expect large divergences and complicated distributions. It is reasonable to expect empirical data to often be of similar shape to a handful of distributions, such as the normal distribution, when we control for most variables; the lognormal distribution, when sporadic delays such as cache misses can impact performance; or a multimodal distribution when the program has conditional branching that impacts performance.

Finally, even a well-measured and well-understood distribution is less useful if it isn't reproducible, and has no impact if it isn't communicated effectively. Reporting performance as a single number robs the data's consumer of the opportunity to understand and explore its full behavior. At the very least, one should strive to report performance with multiple descriptors and summaries, such as mean, standard deviation, kurtosis, certain percentile values, and outliers. Even better would be to depict the complete distribution of every performance measure, for example as a histogram, box plot, density plot, violin plot, raincloud plot, or any other plot that makes sense in context. Fitting a parametric distribution to the empirical one can also be beneficial, not only for a better understanding of the underlying process' behavior but also for simulating it in studies that do not have access to this system [20]. Above all, nothing beats raw data for reproducibility. Students should be encouraged to version, archive, and share all of the raw performance results, from which distributions and summaries can be easily recreated. Naturally, these habits should extend to professional and scientific performance evaluations, where artifact sharing is still not as common a practice as it should be [21].

To recap, following these guidelines to treat performance as a distribution confers several advantages over performance-as-a-number:

- Distributions capture the nuances in performance behavior that a single number cannot.
- Distributions allow the student/practitioner to identify, analyze, and understand outliers before deciding whether to toss them. Outliers are worth investigating because they may indicate something real in the system; they may also exacerbate hurdles to achieving "expected performance" in HPC systems because of the tight synchronization of typical HPC applications [22].
- Distributions enable reproducibility: two consecutive benchmarks can yield diverging summaries, but when looking at the distributions, both summaries may be well within each other's confidence interval.
- Distributions add an important tool to debug performance, which is particularly useful in an educational setting. For example, an operating system class can use performance distributions to explore the effect of context switches, and an architecture class can explore the effects of nonuniform memory access.
- Similarly for practitioners, distributions add a tool for understanding and controlling performance. For example, one may focus not only on improving a performance metric but also on reducing its variability, to lower the occurrence of tail events.

If these hardly novel guidelines were simple to implement, everyone would be following them already. The next section describes some of the pedagogical opportunities for teaching this proposal, followed by practical recommendations on how to address them.

## V. Pedagogical considerations

HPC systems have grown more complex in recent years, with many moving parts such as power throttling, speculative execution, heterogeneous processors and accelerators, multilevel communication, large-scale system noise, and others that vary performance. This new reality requires that we teach and practice performance evaluation in a way that expects, includes, and benefits from this variability [14].

One helpful example is the viewpoint that outliers and secondary modes, rather than being anomalies to be ignored, represent opportunities to discover new or deeper relationships between the system's components that affect performance. It is when experiments diverge from our expectations that we make our most profound discoveries [23]. As another example, sensitivity analysis has always been a part of good experimental design, but performance variability now also adds another factor to evaluate, namely time (or repetitions).

Obtaining a good description of a distribution may require more repetitions of each experiment than in the

"good old days" of deterministic performance. This constraint in turn stresses the need to understand and learn about dynamic stopping rules—as opposed to fixed sample sizes [24]. It also opens the door to a productive discussion of the trade-offs between accuracy and economy, which has always existed in performance evaluation, but can become more informed when the underlying distribution is understood. Educators must emphasize that performance evaluation is often an iterative process, and design assignments that exercise this aspect. For example, an initial measurement and visualization of the performance distribution may reveal that it is skewed, leading to: a hypothesis of a lognormal distribution; a derivation of an appropriate number of samples to estimate its parameters; another measurement and visualization; a new hypothesis based on newly discovered outliers; and so forth.

Our proposed approach also adds another subject to an already burgeoning list of prerequisites before taking an HPC class, namely, statistics. Much like it is in social science research, intermediate statistics should be an indispensable part of the advanced computer systems curriculum. It should include topics such as properties of different distributions, model generation and fitting, Bayesian data analysis, and hypothesis testing methods.

## VI. Recommendations and assignment ideas

By this point, strategies to address these challenges and requirements may be self-evident to the reader. It is nevertheless worth summarizing our main recommendations for curricula, assignments, and practice inside and outside the classroom.

***Recommendation 0: Expect distributions.*** We can rarely expect any more to get meaningful single-number answers to questions like "How fast is this program on that machine?" Instead, plan and teach to measure a complete distribution every time. It goes without saying that all of the principles of good experimental design for performance-as-a-number still apply. For example, use statistical tests to verify that the sample set is stable or representative, quantify uncertainty, and perform sensitivity analysis to identify factors that affect variability.

***Recommendation 1: Report distributions.*** When reporting computer performance, describe all distribution aspects, not just point summaries, as elaborated in Section IV. In the classroom environment, this can be easily controlled at the assignment level. In a publication with strict page limits, this recommendation may not always be feasible. In that case, take advantage of digital appendices, supplementary materials, or even just a link in the paper to a data repository with the complete raw data in machine-readable form. In keeping with this recommendation, this paper includes as supplementary material the complete performance results measured for Figure 1.

***Recommendation 2: Measure enough samples.*** Make sure to account for actual variability in your experiment without relying on rules of thumb for sample size. Use techniques such as sensitivity analysis, statistical power calculation, hypothesis testing, and distribution divergence metrics like the KS test to ensure that you have a sample size that is representative of the underlying distribution and is stable for any summaries of interest.

***Recommendation 3: Measure just enough samples.*** This complementary recommendation is concerned with efficiency and economy rather than accuracy. Once the underlying performance distribution has been confidently identified, compute the minimum number of samples required to approximate the distribution to a desired level, and do not exceed this sample size unnecessarily. If there are no analytical or statistical estimates for a reasonable minimum for the underlying distribution, use a dynamic sequential stopping rule that is appropriate for the underlying distribution, such as those based on high-density interval for asymmetric data [25] or on block bootstrapping for autocorrelated data [26].

***Recommendation 4: Embrace statistics.*** Learn and teach statistics beyond the basics to build a solid foundation in and practice the requisite skills. Use data-science libraries available in languages like Python and R to compute any of the tasks described in this paper, such as modeling sample distributions, evaluating stopping rules, and estimating distribution divergence. Teaching these concepts should remain tied to real-world use cases [27], preferably using actual performance evaluation scenarios.

***Recommendation 5: Embrace exceptions.*** Outliers in data should be investigated before being dismissed. Similarly, irregular or poorly fitting distributions should also be investigated to see if they actually represent a composition of informative distributions. Approach non-Gaussian distributions with Bayesian tools and other flexible statistical models, such as Gaussian Mixture Models. Use model-selection tools to determine how likely it is for an empirical performance distribution to present different modes. Investigate outliers and modes as leads to rejecting previous hypotheses about the measured performance, or to forming new ones.

Based on the above recommendations, we suggest some assignment ideas for different courses and requirements levels. These ideas do not require a new curriculum but rather can be integrated into existing classes that teach about computer performance, such as operating systems, high-performance computing, scientific computing, etc. Even in classes that are less focused on systems, these assignments can be incorporated when discussing measuring efficiency and performance. For example, a class on databases, while discussing performance, typically talks about benchmarks like TPC, and a class on image processing or machine learning might talk about the current benchmarks in their fields. In these contexts, the instructor can introduce the tenets of good benchmarking methodology, including the emphasis on measuring and understanding performance distributions. All of these classes could benefit from motivating examples, like the ones presented

in Sections II and III, but tailored specifically for the topic and applications of the class being taught.

***Beginner: Realizing the need for distributions as a performance measure.*** There are many good benchmark suites available online like Rodinia [17], Stress-ng[2], or the TPC suite[3]. The course instructor can choose the best suite based on course requirements and topics, and prepare an assignment to ask questions like:

- What can you say about the performance of each application in the benchmark suite?
- How many repeated measurements did you make of each application?
- Why did you choose this number of repetitions?
- What can you observe and infer across different runs?
- What is the best way to represent your inferences?

***Intermediate: Understanding variability in systems with interference.*** Here, the course instructor can again choose the required benchmark suite for the assignment. The measurements should be done with different levels of interference from other applications and stressors. The students can run a benchmark with and without interference from another program, and compare the performance histograms to derive insights. Additionally, the students can try to eliminate as much interference as possible from the operating system to make the distribution substantially narrower.

***Advanced: Understanding variability across hardware.*** The course instructor can either choose a parallel processing benchmark suite or a simple matrix multiplication program for the assignment. The students should run and compare the performance distributions for applications on the suite using different programming models and libraries such as OpenMP, MPI, and CUDA, and provide detailed analyses.

## VII. Conclusion and call to action

The days of deterministic and absolute performance are long gone, and HPC education has to adapt accordingly. Performance variability in turn means that performance summaries can be hard to estimate accurately, and even then, can obscure critical insights about the system under test. In this paper, we described and demonstrated the challenges of treating performance as a number, and advocated a predominant perspective of performance-as-a-distribution. This perspective on performance comes with its own pedagogical challenges, and we list a number of concrete recommendations for educators and practitioners to ease the transition.

These recommendations are just the first step in transforming curricula (and practice) to this perspective. Much of the hard lifting remains in the area of developing detailed assignments, case studies, and syllabi for courses on performance evaluation, HPC, and systems programming. There is also much room and need for additional research on distribution-focused topics within the scientific area of performance evaluation methodology. Such topics include distribution approximation and divergence, advanced and adaptive stopping rules for online experiments, and distribution visualization.

Although this paper's focus is on HPC performance, its main thesis also extends to other aspects of computer system evaluation. Chief among these are sustainability measures like power consumption and energy efficiency, which can also fluctuate significantly in the context of power management and renewable energy resources. Similarly, system reliability and availability are also highly variable metrics that can benefit from a distribution-oriented perspective.

We hope that this general principle attracts significantly more attention and acceptance, and plan to continue pursuing research towards this end. We call upon fellow researchers, practitioners, and educators to embrace this perspective and invest in revamping curricula, textbooks, and future publications that center on performance distributions, not performance summaries.

[2]https://github.com/ColinIanKing/stress-ng

[3]https://www.tpc.org/information/benchmarks5.asp

## References


[1] N. Ihde, P. Marten, A. Eleliemy, G. Poerwawinata, P. Silva, I. Tolovski, F. M. Ciorba, and T. Rabl, "A survey of big data, high performance computing, and machine learning benchmarks," in *Performance Evaluation and Benchmarking: 13th TPC Technology Conference, TPCTC 2021, Copenhagen, Denmark, August 20, 2021, Revised Selected Papers 13*. Springer, 1 2022, pp. 98–118.

[2] P. J. Fleming and J. J. Wallace, "How not to lie with statistics: the correct way to summarize benchmark results," *Communications of the ACM*, vol. 29, no. 3, pp. 218–221, 3 1986.

[3] T. Hoefler and R. Belli, "Scientific benchmarking of parallel computing systems: twelve ways to tell the masses when reporting performance results," in *Proceedings of the international conference for high performance computing, networking, storage and analysis (SC'15)*. ACM, 11 2015, pp. 1–12.

[4] S. Hunold, "A survey on reproducibility in parallel computing," *arXiv preprint arXiv:1511.04217*, 11 2015. [Online]. Available: https://arxiv.org/pdf/1511.04217.pdf

[5] J. Garland, R. James, and E. Bradley, "Determinism, complexity, and predictability in computer performance," *arXiv preprint arXiv:1305.5408*, 5 2013. [Online]. Available: https://arxiv.org/pdf/1305.5408.pdf

[6] M. Hocko and T. Kalibera, "Reducing performance nondeterminism via cache-aware page allocation strategies," in *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*. ACM, 1 2010, pp. 223–234.

[7] W.-F. Chiang, G. Gopalakrishnan, Z. Rakamaric, D. H. Ahn, and G. L. Lee, "Determinism and reproducibility in large-scale hpc systems," in *Workshop on Determinism and Correctness in Parallel Programming (WoDet)*, 2013. [Online]. Available: http://wodet.cs.washington.edu/wp-content/uploads/2013/03/wodet2013-final12.pdf

[8] F. Cappello, A. Guermouche, and M. Snir, "On communication determinism in parallel HPC applications," in *Proceedings of 19th International Conference on Computer Communications and Networks*. IEEE, 8 2010, pp. 1–8.

[9] L. Gonnord, L. Henrio, L. Morel, and G. Radanne, "A survey on parallelism and determinism," *ACM Computing Surveys*, vol. 55, no. 10, pp. 1–28, 2023.

[10] A. V. Papadopoulos, L. Versluis, A. Bauer, N. Herbst, J. Von Kistowski, A. Ali-Eldin, C. L. Abad, J. N. Amaral, P. Tuma, and A. Iosup, "Methodological principles for reproducible performance evaluation in cloud computing," *IEEE*

*Transactions on Software Engineering*, vol. 47, no. 8, pp. 1528–1543, 8 2019.

[11] D. Skinner and W. Kramer, "Understanding the causes of performance variability in hpc workloads," in *Proceedings of the IEEE Workload Characterization Symposium.* IEEE, 11 2005, pp. 137–149.

[12] E. Strohmaier, H. W. Meuer, J. Dongarra, and H. D. Simon, "The top500 list and progress in high-performance computing," *Computer*, vol. 48, no. 11, pp. 42–49, 11 2015.

[13] E. Frachtenberg and D. G. Feitelson, "Pitfalls in parallel job scheduling evaluation," in *Job Scheduling Strategies for Parallel Processing: 11th International Workshop, JSSPP 2005, Cambridge, MA, USA, June 19, 2005, Revised Selected Papers 11.* Springer, 6 2005, pp. 257–282.

[14] L. L. Murray and J. G. Wilson, "Generating data sets for teaching the importance of regression analysis," *Decision Sciences Journal of Innovative Education*, vol. 19, no. 2, pp. 157–166, 3 2021.

[15] D. Ghosh and A. Vogt, "Outliers: An evaluation of methodologies," in *Joint statistical meetings*, vol. 12, no. 1, 2012, pp. 3455–3460.

[16] J. K. Brewer, "Statistical rules-of-thumb," *Florida Journal of Educational Research*, vol. 30, no. 1, pp. 5–14, 11 1988.

[17] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *International Symposium on Workload Characterization (IISWC).* IEEE, 10 2009, pp. 44–54.

[18] V. Mittal, P. Bruel, D. Milojicic, and E. Frachtenberg, "Adaptive stopping rule for performance measurements," in *14th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems.* Denver, CO: IEEE, Nov. 2023.

[19] T. B. Arnold and J. W. Emerson, "The r journal: Nonparametric goodness-of-fit tests for discrete null distributions," *The R Journal*, vol. 3, pp. 34–39, 2011, https://doi.org/10.32614/RJ-2011-016.

[20] Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny, "Characterizing facebook's memcached workload," *IEEE Internet Computing*, vol. 18, no. 2, pp. 41–49, 2013.

[21] E. Frachtenberg, "Research artifacts and citations in computer systems papers," *PeerJ Computer Science*, vol. 8, p. e887, 2 2022.

[22] D. Tsafrir, Y. Etsion, D. G. Feitelson, and S. Kirkpatrick, "System noise, os clock ticks, and fine-grained parallel applications," in *Proceedings of the 19th annual international conference on Supercomputing.* ACM, 6 2005, pp. 303–312.

[23] M. K. Kneeland, M. A. Schilling, and B. S. Aharonson, "Exploring uncharted territory: Knowledge search processes in the origination of outlier innovation," *Organization Science*, vol. 31, no. 3, pp. 535–557, 2020.

[24] M. J. Gilman, "A brief survey of stopping rules in monte carlo simulations," 1968.

[25] J. Kruschke, "Doing bayesian data analysis: A tutorial with r," *JAGS, and Stan*, vol. 2, 2014.

[26] S. He, T. Liu, P. Lama, J. Lee, I. K. Kim, and W. Wang, "Performance testing for cloud computing with dependent data bootstrapping," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE).* IEEE, 11 2021, pp. 666–678.

[27] R. Y. Mustafa, "The challenge of teaching statistics to non-specialists," *Journal of statistics education*, vol. 4, no. 1, 12 1996.