Adaptive Stopping Rule for Performance Measurements

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

Pedro Bruel
bruel@hpe.com
Hewlett Packard Labs
USA

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

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

ABSTRACT

Performance variability in complex computer systems is a major challenge for accurate benchmarking and performance characterization, especially for tightly-coupled large-scale high-performance computing systems. Point summaries of performance may be both uninformative, if they do not capture the full richness of its behavior, and inaccurate, if they are derived from an inadequate sample set of measurements. Determining the correct sample set—and in particular, its size—requires balancing tradeoffs of computation, methodology, and statistical power.

In this paper, we treat the performance distribution as the primary target of the performance evaluation, from which all other metrics can be derived. We propose a meta-heuristic that characterizes the performance distribution as it is being measured, dynamically determining when enough samples have been collected to approximate the true distribution. Compared to predetermined fixed stopping criteria, this dynamic and adaptive method can be more efficient in resource use, since it can stop as early as the desired certainty level is obtained, and more accurate, since it does not stop prematurely. Importantly, it requires no advance knowledge or assumptions about the system under test or its performance characteristics.

We evaluate a prototype of our proposal using a mix of synthetic and real benchmarks. For synthetic distributions, this approach closely matches the true distribution. For actual benchmarks, the heuristic is overly conservative for some applications and overly lax for others, especially those using GPUs. But it still matches the overall shape of the distribution for benchmarks with very diverse distributions, which suggests that it is a viable approach for an adaptive stopping rule.

[*]Corresponding author

CCS CONCEPTS

• **General and reference** → **Performance**; **Experimentation**; • **Computing methodologies** → *Massively parallel and high-performance simulations*.

KEYWORDS

high-performance computing, performance evaluation

ACM Reference Format:
Viyom Mittal, Pedro Bruel, Dejan Milojicic, and Eitan Frachtenberg. 2023. Adaptive Stopping Rule for Performance Measurements. In *Proceedings of PMBS'23: Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS '23).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3624062.3624202

1 MOTIVATION AND BACKGROUND

Computer systems performance evaluation is a rich field of research, with thousands of studies dedicated to the theoretical, methodological and practical aspects of obtaining accurate performance measures [12]. One of the complicating factors of such evaluations, especially with the emergence of parallel, distributed, heterogeneous, and serverless computing, is that performance, however one chooses to define it, is rarely deterministic [17]. Nowhere is this more visible and problematic than perhaps in High-Performance Computing (HPC), a field that, as evident from its name, is focused on performance-critical applications of software, middleware, and architectures. As HPC systems grow increasingly larger and more heterogeneous, their performance in turn grows harder to capture with just a single number or two, and the causes of their performance variability grow harder to identify and quantify [22, 30].

Consequently, we should think of performance in computer systems, and especially in HPC, as a random variable. Any description of an empirical random variable—either as a point estimate or as a distribution—requires repeated measurements [14]. A critical challenge for the experimenter is choosing a good number of repetitions [12]. Choose too few, and the measurements would be unreliable; choose too many, and precious compute resources would be wasted; choose a wrong methodology, and the computations would be statistically unstable or invalid [32].

The decision of when to stop repeating a measurement (or *stopping rule*) can be classified into two approaches: *fixed-sample* rules and *sequential* rules [20]. Fixed-sample stopping rules, as their name suggests, stop an experiment after a predetermined fixed number of samples has been obtained, based on certain *a priori* knowledge. These rules are easy to implement but are inefficient in the sense that they may require more samples to reach the same conclusion that a sequential stopping rule would reach [9]. In contrast, a sequential stopping rule determines dynamically the required sample size using information collected from preceding samples.

For example, a common confidence-based rule is to measure samples until the empirical relative precision (standard deviation divided by mean) reaches a prescribed value [6]. A related (but flawed) sequential stopping rule might be to choose in advance a confidence level such as $p_{\text{stop}} = 0.05$ and stop the experiment once we can reject the null hypothesis, that is, that the performance deviates from a specified value [26]. This clearly wrong method is an example of "p-hacking": by definition, 5% of the sample set we collect randomly would "reject the null hypothesis," so by increasing the sample size indefinitely, we would always eventually hit that $p_{\text{stop}} < 0.05$ threshold. We could alleviate this problem, for example by adding a stopping criterion when p_{stop} exceeds a higher threshold [9] or by calculating coverage contours [29].

Similar approaches based on confidence intervals or hypothesis testing exist, but they suffer from statistical weaknesses, such as assuming that samples are independent, or that p-values can be computed accurately—requiring that the sample distribution is static and can be characterized in advance [9]. These assumptions are often unrealistic for computer systems performance tests.

Various works examined different distribution domains for stopping rules, such as continuous monitoring [8], autocorrelated series [11], extreme values [24], symmetric distributions [18], and Bayesian stopping rules [25, 26]. Specifically for computer system performance benchmarking, there are studies on determining the best number of repetitions in specific areas, such as HPC and parallel computing [12], networking [13], performance counters [28], and cloud computing [2, 10, 11].

Other research pointed out methodological and statistical flaws of different stopping rules and some proposed revised rules to overcome these limitations [7, 21, 23]. No single rule is perfect for every situation, so in addition to the methodological and statistical considerations, a careful choice of stopping rule must also include an understanding of the underlying distribution. This is the main limitation that we attempt to overcome here.

This paper does not propose yet another sequential stopping rule, although it does reuse several existing ones. Instead, we ask a different question: if we can assume almost nothing about the performance distribution in advance, can we still select and apply an efficient sequential stopping rule? We answer this question affirmatively by prototyping such a "meta-rule". Like every other sequential stopping rule, ours collects samples, observes them, and dynamically decides how many more samples are required. But rather than focusing only on the performance samples, our system also observes the properties of the performance distribution itself (and how it varies over time), in order to select the most appropriate stopping rule for each distribution as it is discovered in real time.

The next section details the design and implementation of this heuristic. We continue with an empirical evaluation of its efficacy on various synthesized distributions and real performance benchmarks. Finally, we discuss the broader implications of our proposed approach in the Discussion section, and present a summary and potential future work in the Conclusion section.

2 DESIGN AND IMPLEMENTATION

Recall that the purpose of a benchmark, as we define it, is not to summarize any particular performance metric but rather to estimate the true distribution of the metric accurately and expediently, from which all summaries can later be derived. To this end, we measure the success of a stopping heuristic in producing a minimal set of sequential samples that approximates the true distribution to a desired accuracy, based on some distance metric. Assessing this success requires three steps:

1. Estimate the "true" distribution for each benchmark by running it 1,000 times. This sample size is adequate to establish "ground truth" because when we run each benchmark 1,000 times again, we get two very similar distributions.
2. Compute the "optimal" (minimal) number of samples that produces a "close-enough" distribution to the true distribution. Each subset of $1 \leq k \leq 1,000$ samples represents a distribution of its own, whose distance to the true distribution we can measure. Once this distance falls and stays below a predefined threshold, we can say that the subset distribution approximates the true distribution, and its size is the smallest number of samples required to achieve this approximation.
3. Finally, we can measure the success of any stopping rule, including our meta-rule, by comparing the distance of its sample set's distribution to that of the "optimal" and "true" distributions.

Both step 1 and step 2 require a distance statistic between two distributions. We evaluated four such statistics: Jensen-Shannon, Kullback-Leibler, Kolmogorov-Smirnov, and t-test, and opted to use the Kolmogorov-Sminov (KS) test because of its attractive mathematical and empirical properties [1]. As an illustrative example, Figure 1 shows the effect of the sample size k on the distance metric of the subset $1..k$ samples compared to the full $1,000$-sample set, using ten benchmarks from the Rodinia suite [5].

Naturally, all benchmarks start with maximum distance (1.0) when comparing a single sample to the full set, then decreasing exponentially down to zero when comparing the full set to itself. But the path they take there varies from benchmark to benchmark, having different distance measures from the true distribution at the same sample size. The practical implication here is that any fixed sample size would likely cause the experiment to run too many measurements for some benchmarks and too few for other benchmarks to approximate the true distribution. It emphasizes the importance of adapting the number of samples and the stopping rule to the empirical distribution at hand. Our adaptive heuristic attempts to detect the properties of the empirical distribution and stop measurements when these properties stabilize, based on distribution-specific criteria.

To estimate an "optimal" sample set that captures the distribution well, we can compare different subset sizes to the full set we computed offline (an assumption we cannot make in online experiments). Figure 2 shows the effect of choosing different metric threshold values, and consequently, the number of samples required to meet the specified distance threshold. Observe how the histograms appear more similar to the full sample as we decrease the threshold value.

Almost all histograms, except for the first column, appear fairly similar to the "ground truth" distribution, so we have decided to pick the highest threshold value of these (0.1) to represent the "minimal" requirement to approximate the true distribution. From this threshold, we can count the "optimal" number of samples required, which is different for each benchmark, as shown in Figure 2. The optimum is defined to be the minimal number of samples such that the KS distance to the ground truth drops and remains below 0.1 for the remainder of the data. Of course, this choice is somewhat arbitrary, and experimenters are free to choose their own ideal threshold to trade accuracy for speed. But we can find additional validation for this value from the last column that depicts an independent execution of another 1000 runs of the same benchmarks. Even these two large sets (rightmost) diverge somewhat because of factors beyond our control, such as CPU throttling, context switches, etc. The magnitude of this divergence rarely exceeds $KS = 0.1$, which makes it a reasonable baseline threshold for background variability.

It is worth repeating that there is no "magical" reason to pick 0.1 as our threshold. We chose it because in our graphs, it makes the optimal and ground-truth distributions appear close enough, and because it covers most of the natural variance within the ground-truth distributions themselves. But one may choose higher values if the need to complete benchmarks early takes priority over accuracy, and vice-versa.

Also note that using a distance metric like KS is not the same as using a distance metric like RMSE. The latter attempts to minimize the residuals (distance) of all points and bring the curves as closely together as possible, pointwise. Minimizing KS distance attempts only to capture the salient aspects of the distribution, like modes in the same locations, if not the same magnitude, so the distribution plots may show gaps between "optimal" peaks and "ground truth" peaks.

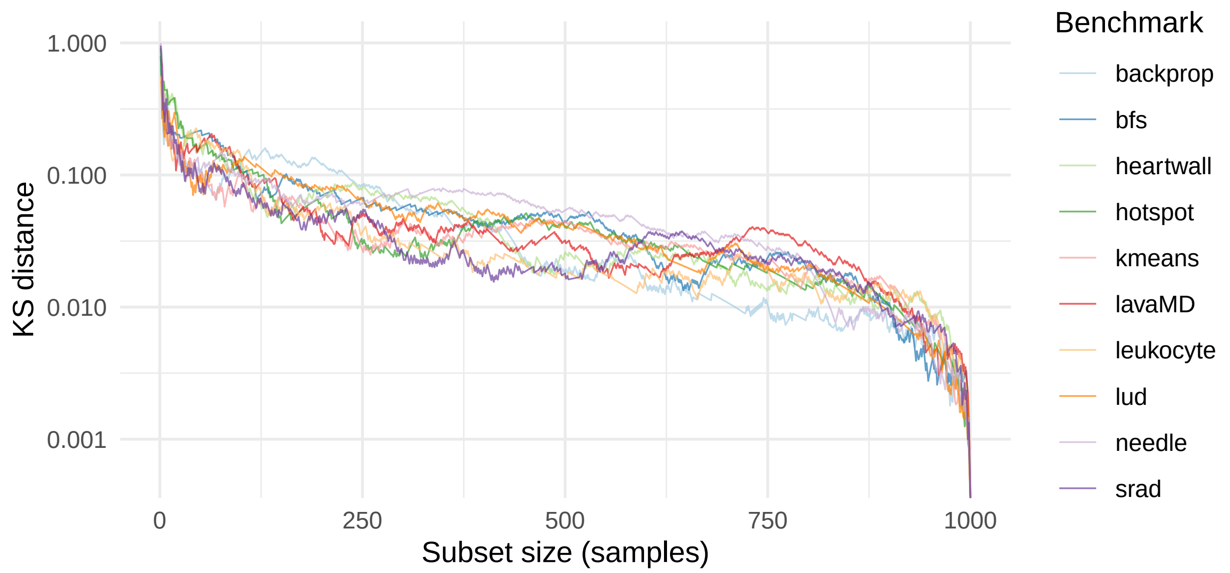

Figure 1: KS distance from full sample for sequential subset run times for ten Rodinia CPU benchmark (log-scale).

Having picked a threshold and determined an "optimal" sample size for each benchmark, we can evaluate how close to this size different stopping rules—including our proposed meta-rule—actually stop. To do so, we next describe the heuristics behind our proposed meta-rule, followed by the actual empirical evaluation.

2.1 Meta-heuristic

The adaptive stopping rule consists of two heuristics, one to identify the underlying sample distribution and one to apply a bespoke stopping criterion for the classified distribution. After collecting an initial small sample set, the rule performs a sequence of if-else queries on the accumulated samples, asking whether the data fits a known distribution, and if so, whether the appropriate stopping criterion decides to stop (Figure 3).

If at any point both queries respond affirmatively, we stop the data collection without attempting other rules. The order of queries, therefore, represents increasing generality of the stopping rules from specific to catch-all, and proceeds as follows:

1. *Constant*: if the difference between minimum and maximum samples is smaller than 10% of the mean, which is a tunable value, we can stop immediately.
2. *Monotonic*: If all samples are monotonic, that is, they keep increasing (or decreasing) over time, we can stop the experiment and warn to check whether the experiment is misconfigured or unstable.
3. *Autocorrelated*: Some performance metrics exhibit temporal patterns and periodicity (e.g., the diurnal patterns in Facebook's Memcache performance metrics [34]). If the measured sample autocorrelation at any positive gap is greater than 0.8—a tunable value—we apply a block-bootstrapping stopping rule [11].
4. *Gaussian*: For unimodal performance distributions, we test whether it fits a Gaussian (normal) distribution, using a tolerant and configurable p-value of 0.2. If so, we apply a simple test of convergence for the width of the samples' standard error.
5. *Log-normal*: Otherwise, the unimodal distribution may be asymmetric, which is common for performance with long-tail events. In this case, we test whether it fits a log-normal distribution—again with a p-value of 0.2—and if so, apply a Bayesian test of convergence for the width of the samples High-Density Interval [16].
6. *Multimodal*: If instead we identify two or more distinctive modes, we use a Gaussian Mixed-Model grid search to find the best fit to a model with up to 6 Gaussian components and 4 types of covariances. If found, we apply a stopping criterion based on a log-likelihood threshold [4].
7. *Uniform*: If all previous tests failed, we check if the sample fits a uniform distribution with a p-value of 0.2. If it does, there is likely something wrong going on with experimental conditions, and we stop and warn the user.

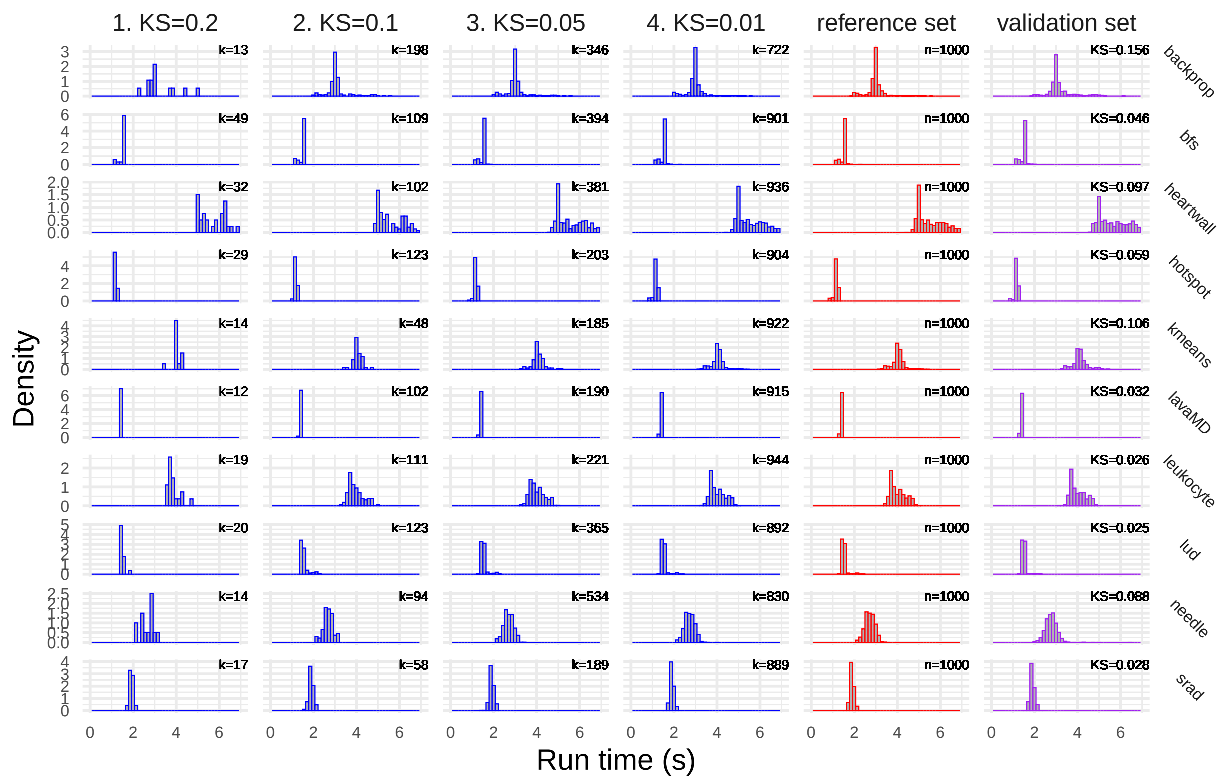

Figure 2: Density histograms for ten Rodinia benchmarks: a reference set with 1,000 runs (red); four subsets for different threshold values of KS distance (blue) from reference; and a separate 1,000-run validation set (purple).

8. Finally, if no applicable stopping rule was found or none chose to stop the experiment, we apply an upper-bound test on the sample size (in essence, a fixed-sample limit of 400) to ensure that the process ends at some point.

Note that all of these heuristics have tunable parameters. Parameter tuning is tedious and error-prone, which is a major disadvantage of our decision-tree predictor. Nevertheless, we decided to start with this simple model because it is easy to interpret and debug. As discussed in Sec. 5, there are more sophisticated models to investigate that could be less brittle and produce better predictions, and would likely supersede this model. Even within the confines of this model, parameter choices could be improved further with an optimization or search heuristic, but this sophistication was not necessary to demonstrate the efficacy of an adaptive meta-rule.

We therefore manually adjusted all of the tunable parameters for our experiments concurrently to approximate the $KS = 0.1$ threshold on the synthetic benchmark suite, and applied those same values to the application benchmark suite, as described next.

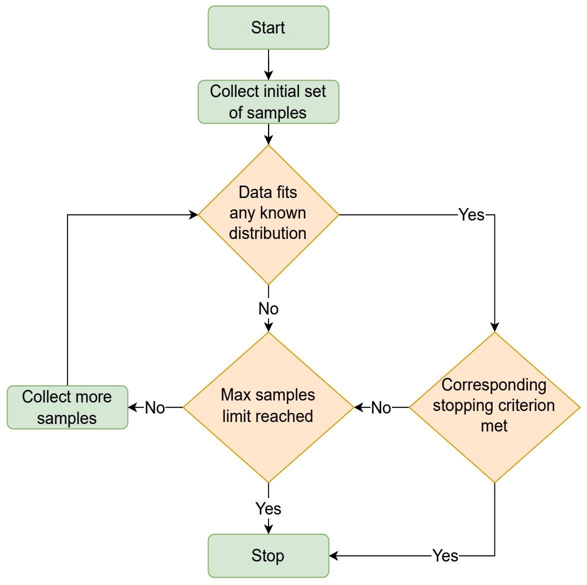
Start
Collect initial set of samples
Data fits any known distribution
Yes
No
Corresponding stopping criterion met
No
Max samples limit reached
No
Collect more samples
Yes
Yes
Stop

Figure 3: Flow chart for meta-heuristic

3 EXPERIMENTAL EVALUATION

Our primary criterion of success for the adaptive stopping rule is how closely it matches the true distribution, as defined in the previous section. To this end, we ran and logged 1,000 measurements each of benchmark from two sets: one synthetic, where we can generate and control the distribution of run times, and one from real benchmarks with diverse distribution characteristics. We then replayed the logs to our adaptive decision rule heuristic and recorded the number of samples it deemed necessary to reach distribution stability.

All of the experiments were executed on a two-node Kubernetes cluster using the Fission framework to represent emerging deployment models, which also add variability [3][1]. The worker node running the benchmarks used an AMD EPYC 7443 24-Core Processor, 256GiB of RAM and an NVIDIA A100X GPU, compile with gcc 8.3.0 and nvcc 11.7.64 on Linux 5.15-0-78 (Debian). The Fission environment used to run the experiments used 40-48000 millicores and 64-168000 MB memory.

3.1 Synthetic Distributions

As a first step toward evaluating and validating our approach, we generated a suite of synthetic "benchmarks" that produce randomized performance numbers based on a desired distribution. We selected ten distributions that represent a spectrum of interesting behaviors, such as nonconverging (Cauchy), autocorrelated (sine), and naturally occurring performance distributions (normal, lognormal, multimodal). This is a limited set of distributions, but we argue that it is capable of representing most performance distributions observed in computer experiments, supported by some evidence from our experiments described in the next section. A different calibration benchmark suite that could improve upon this initial selection could be built from a collection of varied, real, and representative computer experiments such as the Rodinia benchmark. Such a suite could be community-built and reflect currently used applications.

We iterated dozens of runs over the distributions on our synthetic benchmark to fine-tune the various heuristic parameters and thresholds. This is a one-time tuning step that should not need to be repeated for different applications and domains since it covers all the distributions recognized by the meta-heuristic.

Figure 4 shows density plots for ten different synthetic distributions, comparing four methods of data generation: (1) the analytical method generates a continuous PDF from a theoretical distribution function; (2) the ground-truth method generates 1,000 random samples from the distribution, and is the empirical reference point for the next two methods: (3) optimal, which is the smallest subset that meets the KS distance criterion (0.1) from the ground truth, and (4) experimental, which is the subset of samples determined adequate by our meta-heuristic.

Overall, the experimental and optimal distributions look fairly similar. The tunable parameters we picked tend to be conservative for all but three of the distributions, running a little too long and reaching a KS value below the 0.1 value. For the constant and loguniform distributions, the parameters are a little too lax, stopping before reaching the threshold. Only the autocorrelated experiment (sine) fails to approximate the distribution well. The interactions between heuristics for detecting different distributions prevented proper tuning of the autocorrelated detection heuristic, which ended up being triggered and stopping before other more relevant heuristics if it was too strict. We expect that using search and optimization methods to tune these parameters could provide better solutions.

Having tuned our parameters on the synthetic dataset, analogous to a training phase, we continue to a testing phase by evaluating them on real applications.

3.2 HPC Benchmarks

We ran benchmarks from the Rodinia [5] suite for heterogeneous computing (Table 1), of which seven applications had both CPU and GPU versions. CPU-only benchmarks used the OpenMP API while GPU benchmarks used the CUDA API. All of these benchmarks were invoked using a python wrapper which checks the system time before

Table 1: All benchmarks and their parameters

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

[1]When running the experiments locally instead of with Fission, most experiments show similar distributions, except with a shift to the left due to their slightly faster run times.

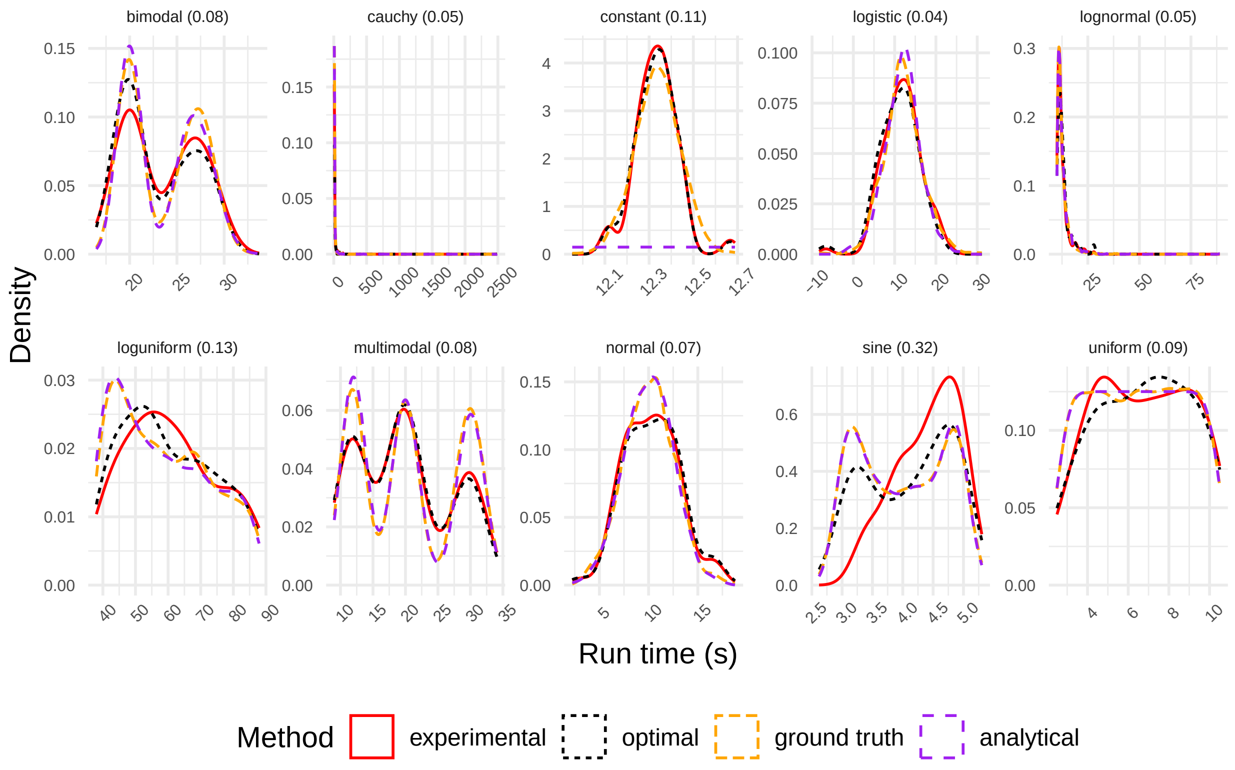

Figure 4: Density plots of synthetic distributions using four methods of generation. Each distribution name is followed by KS distance of the experimental sample relative to the ground truth.

and after the execution of the benchmark's binary. The difference is reported as the execution time in a resolution of $\approx$ 10 microseconds. Since this measurement is done within the application, it captures system performance without including other networking and queuing delays. We consider this execution time the metric of performance. We use default parameters for most of the benchmarks and updated the rest to make the execution time similar to those of Azure functions [27] used in the real world. As in the previous case, each benchmark was repeated 1,000 times to obtain the "ground truth" distribution. Then, we computed the "optimal" sample set at which the KS threshold drops below 0.1. Finally, we replayed each log to our adaptive heuristic and recorded the point where it decided to stop. All three distributions are plotted in Figure 5.

Overall, the shapes of the distributions match closely for most benchmarks, affirming our goal of a stopping rule that produces a representative distribution under diverse workloads. The results are clearly not perfect, however, as evidenced by the experimental distributions' KS distance from the ground truth. Some distributions remain unclassified until they hit the maximum sample limit at the end of the meta-heuristic, resulting in unnecessarily low KS values. Other benchmarks are stopped too early, resulting in higher KS values than our target 0.1. The fact that the model was tuned for synthetic applications, combined with the high levels of irregularity and noise exhibited by real applications, likely explains some of the distance from our target KS values.

This divergence is particularly noticeable in the GPU benchmarks. It is worth noting that none of the synthetic benchmarks on which we trained runs on a GPU, and that GPU distributions appear more irregular and right-tailed than CPU distributions. The seven GPU benchmarks required on average nearly twice as many samples to drop below the KS threshold compared to their CPU counterparts, with one requiring as many as 708 samples! This finding further strengthens the argument that the introduction of heterogeneous computing increases performance variability and requires adaptation of stopping criteria compared to CPU-only implementations.

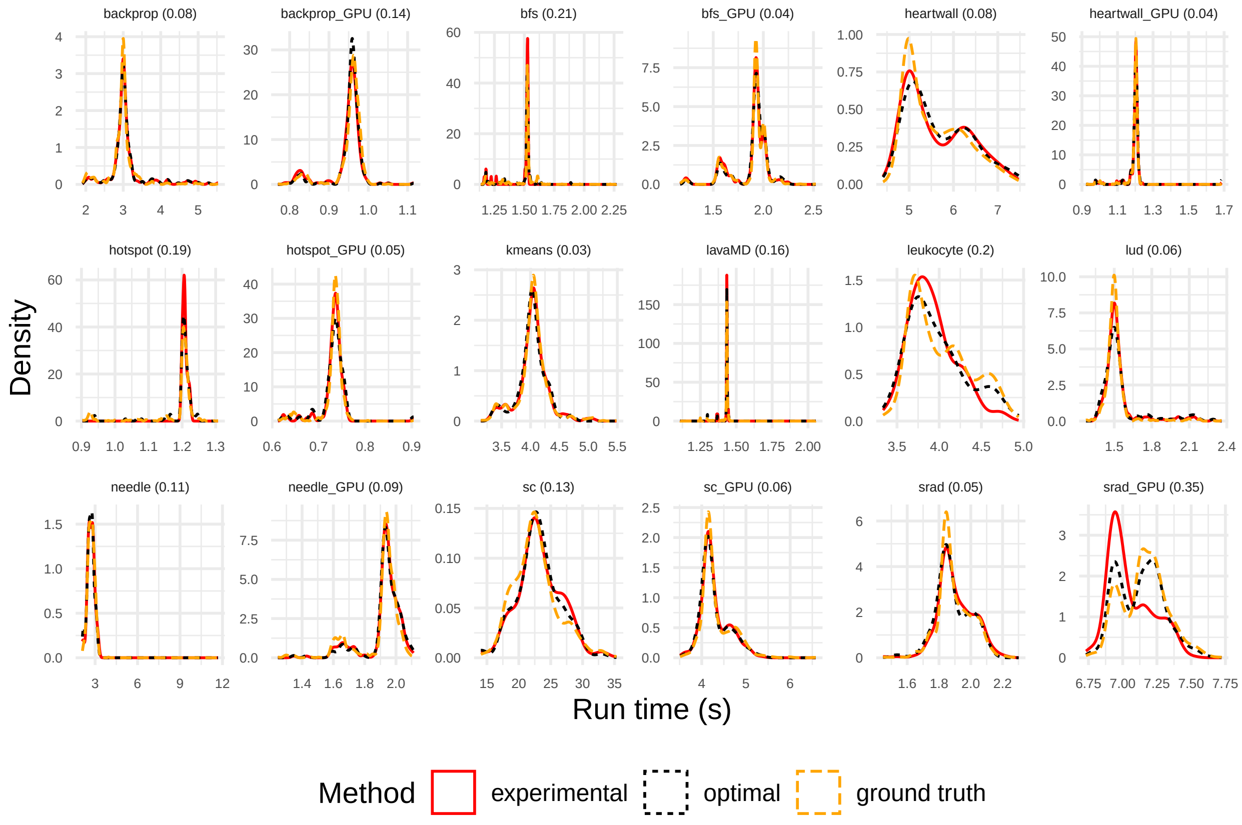

Figure 5: Density plots of Rodinia benchmark distributions using three methods of generation. Each distribution name is followed by KS distance of the experimental sample relative to the ground truth.

4 DISCUSSION

As computer systems grow larger and more heterogeneous, their performance also grows more complex and variable. Understanding systems performance is therefore understanding performance distributions, and measuring distributions accurately requires numerous measurements. Performance evaluations, and more generally, online experiments, should ideally be designed to collect only as many measurements as are needed to understand and summarize the data at a desired confidence level, no more and no less. There are various stopping rules that can be applied to figure out the best time to stop an experiment, but choosing the best rule requires an advanced understanding of the empirical data and its statistical properties. Often, we do not have this understanding before we collect the data, which may be an expensive or time-consuming process. We may also not have the expertise to determine the optimal stopping point for data we do have. Complicating further the decision of when to stop the experiment is the fact that some systems exhibit distinct changepoints, when performance metrics exhibit persistent changes [35].

Our proposal automates the stopping-rule decision process and adapts at run time. It incorporates two primary online components: data analysis to dynamically learn the patterns in the data, and statistical models to determine the most appropriate stopping rule for each pattern. As data is collected and analyzed, the model evaluates and reevaluates the most appropriate stopping rule, and applies it to the measured data. Thus, this proposal both pre-encodes expertise on stopping rules and reacts dynamically at run time to changes and transient effects in the data. Effectively, our proposal trains a very simple distribution classifier using a decision tree. It may be naive and somewhat crude for the tested benchmarks, but it is easy to interpret and debug. We expect that by leveraging results from the extensive literature on machine learning, we could train a more accurate classifier that is more robust to the noise of real benchmarks, which should eventually replace our model.

Although the Rodinia benchmark suite mostly represents HPC use cases, which often scale beyond a single node, we did not need to go outside the single-node limits to observe high performance variability. The benchmarks were ran as parallel programs, using multiple CPU or GPU cores, but without using the network. Large-scale HPC systems introduce even more performance variability because of contention over access to shared resources, such as I/O and the network, but we believe that the same statistical underpinning of our approach still apply.

This work could also generalize beyond HPC performance evaluation to many classes of experiments. Some static and dynamic sequential stopping rules have been studied in domains other than performance evaluation. For example, in deciding when to stop a gradient descent search, an adaptive sequential stopping rule based on an information-content criterion has been proposed [19]. More rules have been proposed in the context of information retrieval [15], surveys [33], and even computer-adaptive testing [31]. While these proposals make sense for their domains, they do not generalize as well to performance evaluation. On the other hand, we believe that our proposed meta-heuristic adapts to most situations where a stopping criterion needs to accommodate an unknown parameter distribution.

5 CONCLUSIONS

In this paper, we address the question of how to estimate the distribution of a data stream as accurately and efficiently as possible without advance knowledge of the distribution. We propose a meta-heuristic (or stopping rule) that dynamically assesses the fit of preceding data to a set of statistical models and based on this fit, chooses the most appropriate stopping rule to compu te and apply. To the best of our knowledge, this is the first proposed meta-heuristic stopping rule that generalizes to nearly any performance evaluation (and in fact, any online experiment). This approach confers the following advantages over static stopping rules:

- Obviate the need to pre-analyze the data.
- Automatically compensate for or avoid the statistical deficiencies of static stopping rules.
- Increase confidence in the statistical strength of the measurements.
- Decrease experimentation time and cost for a desired confidence level.

We implemented a prototype for this meta-heuristic that we plan to open-source in the near future. Our experimental evaluation of this prototype shows that it can identify synthetic distributions accurately and stop very close to our desired threshold. For actual benchmarks, results are mixed, with the occasional large deviations from the target distance values, but also with an overall close approximation of the distribution's shape.

The main direction for future research is therefore to improve the accuracy of our model. This work could include training on more datasets (applications) and using more sophisticated classification models to identify the distribution and more sophisticated regression models to predict the optimal stopping point. We could also explore the use of complex time-series prediction models that are trained offline on thousands of application logs, which may improve prediction accuracy.

Another limitation of our model is that it is not currently designed for efficiency itself. Repeatedly selecting a statistical model to fit the complete sample set at every point is resource-hungry. Although we have not noticed a problem in practice when having to recompute all these heuristics for every new sample, conceivably, some real-time or fast online systems require a more optimized model. One such optimization, for example, is to preserve all the internal state of the functions that do model fitting, and reevaluate them every time with only the incremental sample data. We plan to explore these avenues in the future.

REFERENCES

[1] Taylor B. Arnold and John W. Emerson. 2011. The R Journal: Nonparametric Goodness-of-Fit Tests for Discrete Null Distributions. *The R Journal* 3 (2011), 34–39. Issue 2. https://doi.org/10.32614/RJ-2011-016.

[2] Eytan Bakshy and Eitan Frachtenberg. 2015. Statistics and Optimal Design for Benchmarking Experiments Involving User Traffic. In *24th International World Wide Web Conference (WWW'15)*. Florence, Italy, 108–118.

[3] David Balla, Markosz Maliosz, and Csaba Simon. 2020. Open source faas performance aspects. In *43rd International Conference on Telecommunications and Signal Processing (TSP)*. IEEE, 358–364.

[4] Christopher M Bishop and Nasser M Nasrabadi. 2006. *Pattern recognition and machine learning*. Vol. 4. Springer.

[5] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy W Sheaffer, Sang-Ha Lee, and Kevin Skadron. 2009. Rodinia: A benchmark suite for heterogeneous computing. In *International Symposium on Workload Characterization (IISWC)*. IEEE, 44–54.

[6] J-C Chen, Dingqing Lu, John S. Sadowsky, and Kung Yao. 1993. On importance sampling in digital communications. *IEEE Journal on Selected Areas in Communications* 11, 3 (4 1993), 289–299.

[7] Rianne De Heide and Peter D Grünwald. 2021. Why optional stopping can be a problem for Bayesians. *Psychonomic Bulletin & Review* 28 (2021), 795–812.

[8] Alex Deng, Jiannan Lu, and Shouyuan Chen. 2016. Continuous monitoring of A/B tests without pain: Optional stopping in Bayesian testing. In *2016 IEEE International conference on data science and advanced analytics (DSAA)*. IEEE, 243–252.

[9] Robert W Frick. 1998. A better stopping rule for conventional statistical tests. *Behavior Research Methods, Instruments, & Computers* 30 (12 1998), 690–697.

[10] Steffen Haak and Michael Menzel. 2011. Autonomic benchmarking for cloud infrastructures: an economic optimization model. In *Proceedings of the 1st ACM/IEEE workshop on Autonomic computing in economics*. 27–32.

[11] Sen He, Tianyi Liu, Palden Lama, Jaewoo Lee, In Kee Kim, and Wei Wang. 2021. Performance testing for cloud computing with dependent data bootstrapping. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 666–678.

[12] Torsten Hoefler and Roberto Belli. 2015. 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, 1–12.

[13] Di Huang, Sanfeng Zhang, and Zhou Chen. 2013. An optimal stopping decision method for routing in opportunistic networks. In *2013 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2074–2079.

[14] Sascha Hunold and Alexandra Carpen-Amarie. 2016. Reproducible MPI benchmarking is still not as easy as you think. *IEEE Transactions on Parallel and Distributed Systems* 27, 12 (12 2016), 3617–3630.

[15] Donald H Kraft and T Lee. 1979. Stopping rules and their effect on expected search length. *Information Processing & Management* 15, 1 (1979), 47–58.

[16] John Kruschke. 2014. Doing Bayesian data analysis: A tutorial with R. *JAGS, and Stan* 2 (2014).

[17] Aleksander Maricq, Dmitry Duplyakin, Ivo Jimenez, Carlos Maltzahn, Ryan Stutsman, and Robert Ricci. 2018. Taming performance variability. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 409–425. www.usenix.org/conference/osdi18/presentation/maricq

[18] Adam T Martinsek. 1988. Negative regret, optional stopping, and the elimination of outliers. *J. Amer. Statist. Assoc.* 83, 401 (3 1988), 160–163.

[19] Andreas Mayr, Benjamin Hofner, and Matthias Schmid. 2012. The importance of knowing when to stop. *Methods of Information in Medicine* 51, 02 (2012), 178–186.

[20] Luis Mendo and José M Hernando. 2006. A simple sequential stopping rule for Monte Carlo simulation. *IEEE Transactions on Communications* 54, 2 (2 2006), 231–241.

[21] Raymond S Nickerson. 2000. Null hypothesis significance testing: a review of an old and continuing controversy. *Psychological methods* 5, 2 (2000), 241.

[22] Alessandro V Papadopoulos, Laurens Versluis, André Bauer, Nikolas Herbst, Jóakim Von Kistowski, Ahmed Ali-Eldin, Cristina L Abad, José Nelson Amaral, Petr Tuma, and Alexandru Iosup. 2019. Methodological principles for reproducible performance evaluation in cloud computing. *Transactions on Software Engineering* 47, 8 (8 2019), 1528–1543.

[23] Jeffrey N Rouder. 2014. Optional stopping: No problem for Bayesians. *Psychonomic bulletin & review* 21 (4 2014), 301–308.

[24] Anne Sabourin. 2021. *Extreme Value Theory and Machine Learning*. Ph.D. Dissertation. Institut polytechnique de Paris.

[25] Adam N Sanborn and Thomas T Hills. 2014. The frequentist implications of optional stopping on Bayesian hypothesis tests. , 283–300 pages. https://doi.org/10.3758/s13423-013-0518-9

[26] Felix D Schönbrodt, Eric-Jan Wagenmakers, Michael Zehetleitner, and Marco Perugini. 2017. Sequential hypothesis testing with Bayes factors: Efficiently testing mean differences. *Psychological methods* 22, 2 (2017), 322.

[27] Mohammad Shahrad, Rodrigo Fonseca, Inigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. 2020. Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. In *2020 USENIX annual technical conference (USENIX ATC 20)*. 205–218.

[28] Weiyi Shang, Ahmed E Hassan, Mohamed Nasser, and Parminder Flora. 2015. Automated detection of performance regressions using regression models on clustered performance counters. In *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*. ACM, 15–26.

[29] Dashi I Singham and Lee W Schruben. 2012. Finite-sample performance of absolute precision stopping rules. *INFORMS Journal on Computing* 24, 4 (2012), 624–635.

[30] David Skinner and William Kramer. 2005. Understanding the causes of performance variability in HPC workloads. In *Proceedings of the IEEE Workload Characterization Symposium*. IEEE, 137–149.

[31] Rose E Stafford, Christopher R Runyon, Jodi M Casabianca, and Barbara G Dodd. 2019. Comparing computer adaptive testing stopping rules under the generalized partial-credit model. *Behavior research methods* 51 (6 2019), 1305–1320.

[32] Alexandru Uta, Alexandru Custura, Dmitry Duplyakin, Ivo Jimenez, Jan Rellermeyer, Carlos Maltzahn, Robert Ricci, and Alexandru Iosup. 2020. Is big data performance reproducible in modern cloud networks?. In *17th symposium on networked systems design and implementation (NSDI)*. USENIX, 513–527. www.usenix.org/conference/nsdi20/presentation/uta

[33] James Wagner, Xinyu Zhang, Michael R Elliott, Brady T West, and Stephanie M Coffey. 2023. An experimental evaluation of a stopping rule aimed at maximizing cost-quality trade-offs in surveys. *Journal of the Royal Statistical Society Series A: Statistics in Society* (2023), qnad059.

[34] Yuehai Xu, Eitan Frachtenberg, Song Jiang, and Mike Paleczny. 2013. Characterizing facebook's memcached workload. *IEEE Internet Computing* 18, 2 (2013), 41–49.

[35] Yuxuan Zhao, Dmitry Duplyakin, Robert Ricci, and Alexandru Uta. 2021. Cloud performance variability prediction. In *Companion of the ACM/SPEC International Conference on Performance Engineering*. ACM, 35–40.