A Case Study on the Value of Simulated and Synthetic Microservice Applications for Performance Model Training

Yannik Lubas[1], Martin Straesser[1], Ivo Rohwer[1], André Bauer[2], Samuel Kounev[1]
[1]University of Würzburg, Germany
[2]Illinois Institute of Technology, Chicago, IL, USA

Abstract

Microservices have become a central architectural style for cloud-based systems, driving increased interest in their performance characteristics. However, the limited availability of real-world microservice applications for academic research constrains empirical studies in this area, particularly those using application-agnostic, machine learning-based approaches. Prior work has explored synthetic generation and simulation as alternative means of obtaining performance data; however, their benefits when used alongside real-world measurements are less understood. Using the saturation predictor Monitorless in a case study, we evaluate two scenarios. Our results indicate that generated and simulated applications improve predictive performance on previously unseen, real-world systems. In our case, generated applications reduced false positives, simulated applications enhanced robustness, while their combination yielded the most balanced predictor. The findings suggest that synthetic and simulated applications can effectively complement real-world data, enabling more diverse and robust evaluation of performance models.

1 Introduction

Microservices are an integral part of modern cloud systems, offering scalability and flexibility during development and operation. Yet, their distributed nature also introduces substantial challenges for performance management. Due to the heterogeneous landscape of cloud applications, combined with the fast-paced development of applications enabled by modern DevOps practices, most approaches rely on machine learning (ML) models. Instead of learning specialized models for a single service or application, these models extract application-agnostic performance models that encompass the performance characteristics of multiple, diverse services or applications, limited only by the diversity of the training data.

However, obtaining sufficiently diverse training data remains difficult. Open-source microservice applications suitable for performance studies are scarce, and setting up test environments is a labor-intensive process. Consequently, many research works rely on a limited set of applications, a recurring threat to validity that has been highlighted in the literature [4]. Microservice simulators and synthetic application generators offer a promising way to overcome this limitation by enabling rapid creation of diverse training data.

Building on our prior work, which showed the benefits of augmenting training data with generated applications for performance degradation prediction, this paper presents a case study on enriching training data with both simulated and synthetic applications. We demonstrate that such enrichment improves predictive performance on a dataset of previously *unseen* and *real* applications.

2 Foundations and Previous Work

MiSim [3] is a simulation tool designed to model and analyze microservice applications, with a focus on resilience and performance. Currently, the tool simulates the processing of requests using their CPU consumption. Other resources, such as memory or disk usage, are not simulated yet.

Creo [4] is a multi-language microservice application generator focusing on performance benchmarking. The generated applications are fully executable and have configurable performance characteristics. The built-in support for monitoring, load generation, and deployment automates most of the setup steps for performance experiments.

Monitorless [2] is an ML-based model that predicts instance saturation using platform-level metrics (i.e., CPU usage). The application-agnostic approach relies on the diversity and representativeness of the training data.

In our previous work [4], we demonstrated that Monitorless can benefit from training data enriched with measurements from generated applications. The extended training dataset improved the results on the test dataset (comprising a *real* microservice application).

3 Experiment Design

A major challenge in evaluating application-agnostic approaches, such as Monitorless, lies in constructing representative datasets. Since open-source applications are both rare and complex to deploy and benchmark, experiments with real applications are typically

limited to only a few candidates. This narrow scope restricts test diversity, which may lead to an overly optimistic or pessimistic assessment of prediction accuracy, while failing to capture the variability encountered in realistic deployment environments.

This limitation creates a fundamental trade-off. Reserving more of the limited number of applications for the test set increases test diversity but decreases training diversity, potentially reducing predictive performance. Conversely, including most available applications in the training data maximizes training diversity but forces evaluation on a homogeneous test dataset, thereby constraining the generality of conclusions. As a result, the cost of generating sufficient high-quality performance data with real applications makes it difficult to evaluate application-agnostic performance engineering approaches.

To overcome this limitation, researchers could enrich training data with both synthetic and simulated applications. This allows us to construct a diverse training dataset while simultaneously broadening the scope of test evaluation. We define the following two scenarios to investigate the impact of synthetic and simulated applications systematically.

In Scenario A, we investigate the impact of enriching training data with both simulated and generated applications. Since MiSim currently only simulates the CPU, we focus on applications with a CPU bottleneck. As a baseline, we train our predictor on data from the reference application TeaStore [1] and evaluate it on data obtained from Apache Solr. We then incrementally augment the training set; first by adding a generated application with high CPU usage, then by incorporating simulation data modeled from the microservice application RobotShop,[1] and finally by including the data from both the generated and simulated applications. This setup enables us to assess the isolated impact of simulation data and generated applications, as well as their combined effect.

In Scenario B, we investigate the predictor's generalization across heterogeneous resource bottlenecks. Unlike Scenario A, we leverage all available platform metrics as features. Here, the evaluation dataset includes not only Solr, as a CPU-bound application, but also Memcached (a network-bound application) and Cassandra (a memory-bound application). Since MiSim currently exclusively supports CPU, we focus on enriching the training set with generated applications. Specifically, we include two additional generated applications (with high network and memory usage) along with the generated, CPU-bound application from Scenario A. This setup ensures that the training data covers heterogeneous bottleneck scenarios.

With this design, we investigate whether synthetic and simulated applications can effectively substitute for real-world applications in the training data, thereby enabling a diverse and representative test dataset that includes real-world applications. Our goal is to demonstrate that incorporating generated and simulated data leads to more robust training data, while also facilitating the collection of performance data compared to real applications.

[1] `https://github.com/instana/robot-shop`

4 Results

This section presents our evaluation results. All results can be replicated with our replication package.[2] Across all experiments, we construct the training data from the following datasets:

- **Base:** performance data from the reference application TeaStore
- **Gen-A:** performance data from the generated, CPU-bound application
- **Sim:** performance data from the simulated application RobotShop
- **Gen-B:** performance data from generated applications with targeted bottlenecks (CPU, network, memory)

All predictors are evaluated on *real-world* applications outside the training dataset. As described in the previous section, Scenario A focuses solely on CPU features and utilizes Apache Solr as the target application. In Scenario B, we use all available platform metrics as potential features. Here, the evaluation covers heterogeneous resource bottlenecks with Solr (CPU), Memcached (network), and Cassandra (memory) as target applications. In the following, we adopt the *lagged* scoring method introduced in Monitorless [2]. When an application becomes saturated, platform-level metrics exhibit an immediate increase. However, the corresponding degradation in key performance indicators (KPIs) manifests only after a delay resulting from the inherent latency of response-time measurements. To account for this temporal offset, the lagged scoring method is employed as an adjustment. We refer to the original paper for a detailed explanation.

The results for Scenario A are shown in Table 1. The baseline model trained solely on TeaStore achieves an F1 score of 87.0%, with a particularly high recall (99.4%) but relatively low precision (77.3%). This indicates that the model is highly sensitive in detecting saturation events, but at the cost of more false positives. During operation, this would mean that the predictor raises saturation warnings frequently, ensuring degradations are rarely missed, but at the expense of frequent scaling decisions.

Augmenting the training set with the generated CPU-bound application (Base+Gen-A) improves the balance between precision and recall; precision increases to 88.0% while recall remains high at 92.7%.

[2] `https://doi.org/10.24433/CO.4603902.v1`

Training Dataset	F1	Precision	Recall
Base	87.0%	77.3%	99.4%
Base+Gen-A	90.3%	88.0%	92.7%
Base+Sim	89.3%	85.1%	94.0%
Base+Gen-A+Sim	90.5%	89.9%	91.2%

Table 1: Scores of Scenario A

Training Dataset	F1	Precision	Recall
Base	60.2%	59.2%	61.1%
Base+Gen-B	66.3%	62.1%	71.3%

Table 2: Scores of Scenario B

The resulting F1 score (90.3%) demonstrates that generated data contributes effectively to refining the decision boundaries and reducing false alarms.

Similarly, incorporating simulated data from RobotShop (Base+Sim) raises precision to 85.1% and recall to 94.0%, yielding an F1 of 89.3%. While this configuration improves over the baseline, its precision gain is slightly lower than with generated data. However, the recall drop-off is less. The combination of generated and simulated applications (Base+Gen-A+Sim) achieves the highest F1 score (90.5%), with balanced precision (89.9%) and recall (91.2%).

In summary, the results from Scenario A demonstrate that both generated and simulated applications may effectively complement real-world data. In our case, generated applications are particularly beneficial for reducing false positives, while simulation data provides additional robustness. Their combination produces the most balanced predictor.

Table 2 reports the results for Scenario B. Here, the baseline achieves an F1 score of 60.2%, with balanced precision (59.3%) and recall (61.1%). Compared to Scenario A, the overall performance is notably lower, reflecting the challenge of generalizing from a single CPU-bound training source (TeaStore) to heterogeneous bottlenecks (CPU, network, and memory).

When extending the training dataset with generated applications targeting each bottleneck, performance improves across all three metrics; F1 rises to 66.3%, driven by increases in both recall (71.3% and precision (62.1%). These results demonstrate that generated applications enable the predictor to generalize more effectively across heterogeneous bottleneck resources. Importantly, this is achieved without requiring additional real-world applications, leveraging the built-in benchmark automation of Creo.

Across both scenarios, the results indicate that enriching training data with generated or simulated applications improves prediction performance compared to a baseline classifier trained on a single real-world application. In our case study, the impact is most pronounced in precision: the enriched training sets reduce the degree to which the predictor overestimates saturation events, thereby lowering the number of false positives. This may be particularly valuable in practical deployments, where false alarms can lead to unnecessary adaptation actions.

At the same time, recall remains consistently high across training sets. The results suggest that true saturation events are reliably detected. In the context of this case study, recall is critical for avoiding missed degradations. The desired balance of precision and recall in real-world scenarios may depend on the deployment environment and the actions taken due to saturation events. In general, we observed a disproportionate increase in precision compared to the decrease in recall, resulting in a net improvement in the F1 score.

5 Conclusion

In this paper, we investigated the impact of enriching training datasets for ML-based performance degradation prediction with synthetic and simulated microservice applications. Across two experimental scenarios, we demonstrated that both data sources can improve predictive performance for previously unseen, real-world applications. In our case, generated applications helped reduce false positives, while simulated applications added robustness. Their combination produced the most balanced results for CPU-bound saturation. Moreover, generated applications facilitated generalization across heterogeneous bottlenecks without requiring additional real-world systems.

Overall, our findings highlight that synthetic and simulated applications can mitigate the scarcity of suitable benchmarks in performance engineering research. By reducing the labor-intensive benchmark process, generated and simulated applications provide a practical means for constructing diverse and more robust training data.

Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 536362030.

References

[1] J. von Kistowski et al. “TeaStore: A Micro-Service Reference Application for Benchmarking, Modeling and Resource Management Research”. In: *MASCOTS*. 2018, pp. 223–236.

[2] J. Grohmann et al. “Monitorless: Predicting Performance Degradation in Cloud Applications with Machine Learning”. In: *Middleware*. 2019, pp. 149–162.

[3] S. Frank et al. “MiSim: A Simulator for Resilience Assessment of Microservice-Based Architectures”. In: *QRS*. 2022, pp. 1014–1025.

[4] Y. Lubas et al. “Generating Executable Microservice Applications for Performance Benchmarking”. In: *ICPE*. 2025, pp. 31–44.