2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W)

Utilizing Clustering to Optimize Resource Demand Estimation Approaches

Johannes Grohmann, Simon Eismann, André Bauer, Marwin Züfle, Nikolas Herbst, Samuel Kounev
University of Würzburg, Germany
Email: {firstname}.{lastname}@uni-wuerzburg.de

***Abstract*—Resource demands are crucial parameters for modeling and predicting the performance of software systems. Direct measurement of these resource demands is usually infeasible due to instrumentation overheads causing measurement interferences and perturbation in production environments. Thus, a number of statistical estimation approaches (e.g., based on optimization, regression or Kalman filters) have been proposed in the literature. Most of these approaches are parameterized. These parameters influence the estimation quality and the required computation time. Existing work uses historical data as training sets to optimize those parameters and to minimize the estimation error of those approaches. However, if the data traces are fundamentally different, the optimal parameter settings are different as well.**

In this paper, we propose to use automated clustering in order to group training sets into groups of similar optimization behavior. This way, optimization can be specifically tailored to certain groups of traces in a self-aware manner. During run-time, every trace is first sorted into a cluster, where the respective cluster-wide parameter optimum can be applied. A preliminary case study shows that clustering can provide promising improvements.

***Index Terms*—self-adaptive systems; resource demand estimation; optimization; clustering; machine learning**

I. Introduction

A resource demand (or service demand) is the average time a unit of work (e.g., request or transaction) spends obtaining service from a resource (e.g., CPU or hard disk) in a system over all visits excluding any waiting times [1]. Timely and precise resource demand estimates are a crucial input to auto-scaling mechanisms [2] or performance modeling techniques [3], [4] used for elastic resource provisioning and therefore different self-aware computing systems [5], [6]. Hence, it has been shown that statistical estimation of resource demands is a valid and useful tool enabling elastic cloud environments [2], [7]. Over the years, a number of approaches to resource demand estimation have been proposed using different statistical estimation techniques (e.g., linear regression [8], [9] or Kalman filters [10], [11]) and based on different laws from queueing theory.

Resource demand estimation approaches need to be parameterized in order to adapt their behavior to the specific problem instance and to improve estimation accuracy. Examples of such parameters include the step size, i.e., the aggregation interval of measurements, the window size, i.e., the amount of historical information to consider or hyper-parameters of, e.g., Kalman filters or optimization engines. However, it is challenging to manually determine sensible parameter settings as the relationship between the parameters and the accuracy is not always trivial. A poor choice of parameter values can drastically decrease the estimation accuracy of some approaches, while others may remain relatively unaffected [12]. This complexity hinders the adoption of resource demand estimation techniques in practice [13].

Our previous work introduces a self-tuning approach based on discrete optimization that can be used to automatically tune the parameters of resource demand estimation methods, provided sample measurement traces from the given application domain are available [14]. This approach finds a single parameter setting minimizing the estimation error over all given traces. However, as the given traces have different characteristics, they might be impacted by different parameter settings in different ways. This implies that an accuracy increase for one measurement trace can mean a decrease for another one. As the algorithm can only optimize the average estimation error, the optimization must settle for a compromise.

In this paper, we propose to utilize clustering in order to separate the training traces into different groups. These clusters are formed based on the optimization behavior of those traces. Each of these clusters is then optimized individually, improving the estimation accuracy. Any new trace is grouped into one of the existing clusters and then estimated using the recommended optimized parameterization. By clustering the traces, we can improve the estimation error of the resulting groups by optimizing each of them separately.

Based on the number of resulting clusters or the distinguishing features between them, we can deduce the important parameters having the biggest influence on the optimization These can be used to extract some meta-knowledge about the optimization behavior of different traces. As the approach works fully autonomous, it can be seamlessly integrated with existing resource demand estimation techniques. Therefore, our approach is step forward towards our vision of self-aware performance models [15] as well as the general vision self-aware computing [5].

II. Related work

There are several works introducing resource demand estimation approaches. Rolia and Vetland [8], [16] first did some experiments for linear regression techniques. Pacifini et al. [17], Casale et al. [18], [19] and Stewart et al. [20] extend this by investigating limitations of linear regression in resource

978-1-7281-2406-3/19/$31.00 ©2019 IEEE
DOI 10.1109/FAS-W.2019.00043

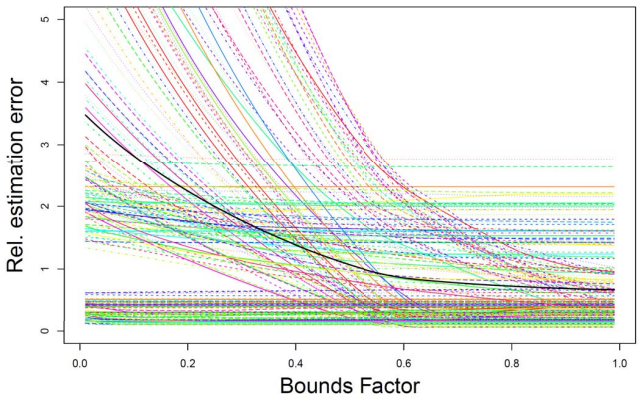

Fig. 1. Estimation error for different traces in dependence of a configurable parameter. Each color represents a different trace, the black line represents the total mean.

demand estimation and the impact of different factors. The performance of Kalman filters for resource demand estimation is researched by Zheng et al. [11], [21], Kumar et al. [22] and Wang et al. [10], [23]. Kraft et al. [24] and Sharma et al. [25] both compare least-squares regression with their maximum likelihood estimation [24] and independent component analysis [25] approach, respectively. None of these works performs an automatic and systematic optimization of the best parameter settings for a given data set, since these approaches only do manual testing and develop rules of thumb for a chosen small set of parameters.

Our previous work [14] proposed an algorithm optimizing the hyper-parameters of each approach for that data set. Another work deals with the selection of the best approach for a given scenario [26]. However, both works assume the test set to be indivisible. Therefore, the approaches have to search a single configuration for the complete training set, resulting in sub-optimal decisions for individual traces.

III. Motivating example

This section describes an example optimization motivating the need for clustering algorithms, when optimizing different traces for resource demand estimation. Figure 1 shows the optimization behavior of different traces for a certain scenario. In this example, we show the relative estimation errors for each training trace of a Kalman filter implementation based on the utilization law [10] in dependence on the so-called bounds factor, a parameter of the Kalman filter. Every line represents one specific estimation problem, i.e., one measurement series used for estimating resource demands.

We observe that the arithmetic average over all traces (black line) decreases monotonously with increasing values of the Bounds Factor parameter. Almost any optimization algorithm would therefore select a parameter value close to 1.0. However, by analyzing the individual traces we can observe at least two clusters, exhibiting different properties. The first group reacts sensitive to the bounds factor and drastically decreases its error for increasing values of the bounds factor. Another relatively

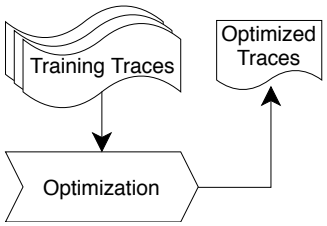

Fig. 2. The standard optimization approach [14].

compact group has a low error anyway and is therefore not sensitive to changes in the bounds factor. Hence, the second group does not profit from an increased bounds factor. We can identify more clusters based on how steep the accuracy increase is, or when the optimal value is reached.

This cluster separation can be important if the optimized parameters have an impact on the performance of the resource demand estimation and/or if two or more cluster optima are different from each other. Since the current optimization has to choose *one* optimum for all traces, a compromise has to be reached. Instead, we propose to split the total set of training traces into different clusters. This way, every cluster can be optimized separately.

IV. Approach

In this section, we describe two possibilities of clustering a set of training traces based on their optimization properties. We already outlined the envisioned benefit of the proposed clustering in Section III. The respective advantages and disadvantages are discussed in Section V-C.

We first discuss the traditional approach to parameter optimization, in order to highlight our improvements to the current process. Our previous paper [14] envisioned the optimization process as depicted in Figure 2. The approach uses a set of training traces as input. Each trace represents one problem instance, for which a resource demand estimation is required. Executing the estimation requires to parameterize a given approach with specific values.

The optimization algorithm explores different (multi-dimensional) parameter configurations and select the best one based on minimizing the estimation error calculated using k-fold cross-validation. Therefore, the optimization algorithm returns a set of tuned parameter settings that have empirically proven to be optimal for the respective training set. As this is a general multi-parameter optimization problem, a multitude of different optimization approaches can be applied. Our previous work [14] explored different algorithms, namely one hill-climber algorithm, a variant of stepwise sampling search (S3) [27] and a brute force algorithm.

The found parameter set can now be used for any other trace. One core assumption is that the new trace has somewhat similar properties to the traces used during the training/optimization phase. If so, all succeeding estimation traces will benefit from optimized parameter settings and therefore improved accuracy. It is hence beneficial to augment the

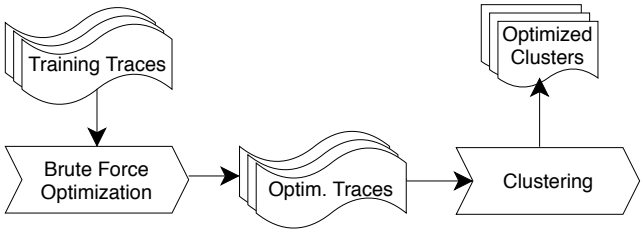

Fig. 3. Clustering traces using brute force optimization.

training set as much as possible, in order to include as many different traces as possible. However, as outlined in Section III, different groups of traces react differently to changes in parameter settings. Therefore, the optimized parameter is always the result of a compromise.

We address this issue by clustering the training set into different groups and optimize their parameters separately. For every new trace, we calculate a set of features and assign it to a cluster based on these features. Now, the parameter set of the given cluster is used for the new trace.

A. Feature set

In the following, we introduce the feature set, we consider to be important for the clustering of different resource demand estimation traces. We experimented with different feature settings and combinations. In total, we evaluated 24 different features. They were gathered based on other results from previous work [12], [26]. In order to keep the feature space small, we finally decided to reduce the relevant feature set to the following four features:

- Number of resources: The number of processing units, e.g., CPUs, of the given trace.
- Number of workload classes: The number of different request classes or workload classes of the trace.
- Variance inflation factor: A measure for collinearity between the measurement series of one trace [28].
- Arithmetic mean for each resource utilization: The average resource utilizations per resource.

We aim for a trade-off between small feature size and high cluster quality, where the presented set has empirically proven to be sufficient, as most bigger feature sets actually result in a worse clustering. Reducing the feature set as much as possible is desirable, since it reduces the risk of over-fitting and furthermore increases explainability and interpretability for the human.

B. Brute force clustering

Our first approach utilizes clustering, based on the actual optimization behavior of the traces. Intuitively, we could say we cluster based on the graphical clusters, observable in Figure 1. However, note that this can be time intensive as this requires one execution for each parameter set for every training trace to generate the corresponding estimation error.

As depicted in Figure 3, our approach therefore first executes a brute force optimization algorithm. The brute force algorithm, as the name already suggests, iterates over the whole parameter space to find the best parameter setting. Additionally, we store the optimal parameter setting for each trace.

We now apply the clustering based on the actual optimization behavior of these traces. Each found cluster gets optimized separately. However, as we already went through all traces with our brute-force approach, it is trivial to choose the optimal values for each cluster. Additionally, we calculate the features for the resulting clusters. For every unseen trace, we calculate the required features for the given trace and assign it to the cluster with the closest feature distance. The estimation of that trace will then be executed with the respective optimal parameter setting of that cluster.

C. Feature-based clustering

A downside of brute force clustering is the computational intensity of the training phase. As applying brute force is part of the clustering process itself, this method can easily become infeasible in practical settings, since brute force is not always applicable [14]. We therefore introduce the so-called feature-based clustering, depicted in Figure 4.

In contrast to the brute force clustering, we firstly extract the features for each of the traces. Now, the actual clustering process is based on these feature distances, instead of the optimization behavior. The resulting clusters can now be optimized separately, to obtain an optimal parameter setting for each of the individual clusters. As the brute force search is no longer required to generate the clustering itself, we can choose more efficient optimization algorithms that do no longer guarantee an optimal solution, but consume much less computation power. As the S3 algorithm [27] has proven to be sufficient [14], we will use S3 in the following experiments.

V. Case Study

This section presents a preliminary case study using the proposed clustering methods. We first describe the experiment setup, then present results of the two proposed techniques and finally discuss our findings.

A. Setup

The considered training set consists of measurements obtained from running micro-benchmarks on a real system. In total, 210 traces of approximately one hour run time were collected, resulting in a total experiment time of around 210 hours. The micro-benchmarks generate a closed workload with exponentially distributed think times and resource demands. As mean values for the resource demands, we selected 14 different subsets of the base set [0.02s; 0.25s; 0.5s; 0.125s; 0.13s] with number of workload classes $C = \{1; 2; 3\}$. The subsets were arbitrarily chosen from the base set so that the resource demands are not linearly growing across workload classes. The subsets intentionally also contained cases where two or three workload classes had the same mean value as resource demand. We varied the number of workload classes

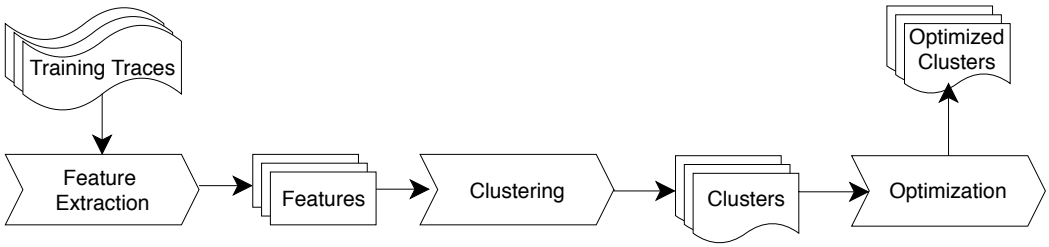

Fig. 4. Clustering traces using features.

$C = \{1; 2; 3\}$ and the load level $U = \{20\%; 50\%; 80\%\}$ between experiments. These traces were already used in previous studies [12], [14], [26].

The error of a resource demand estimate is defined by the sum of the relative response time error E_R and the absolute utilization error E_U:

$$E_R = \frac{1}{C} \sum_{c=1}^{C} \left| \frac{\tilde{R}_c - R_c}{R_c} \right|, \qquad E_U = \left| \sum_{c=1}^{C} (X_c \cdot \tilde{D}_c) - U \right|, \tag{1}$$

with C being the number of workload classes, R_c the average measured response time of workload class c over all resources, $\tilde{R}_c$ the predicted average response time based on the estimated resource demands, X_c the measured throughput of workload class c, $\tilde{D}_c$ the estimated resource demand of workload class c and U the average measured utilization over all resources.

We experimented with different variants of the iterative k-means [29], the iterative k-medoids [30] and DB-SCAN [31]. In the following, we present our results based on the iterative k-means using the silhouette coefficient [32] as stopping criterion for evaluating the cluster quality for a given k. The silhouette coefficient is a measure of how similar an object is to its own cluster compared to how similar it is to the other clusters [32]. Note that this implies that not all parameter optimizations result in the same amount of clusters, since we do not want to restrict the clustering process to any specific cluster number k. As distance measure, we use the sum of all point-wise distance of the optimization curves for the brute-force clustering, and the euclidean feature distance for feature clustering.

B. Results

Figure 5 shows the clustering of the brute-force clustering approach. It shows the clustered traces for the bounds factor of two different Kalman filters: (1) KF1 by Wang et el. [10] to the left and KF2 by Zheng et al. [11] on the right. The left figure represents the clustered version of Figure 1. A visual inspection confirms that the algorithm identified reasonable clusters. Furthermore, we note that the black cluster means show different behaviors. Therefore, we conclude that the idea of clustering different estimation traces is feasible, as we can separate different groups of behavior. On KF2 optimization (right), we can see that the two clusters have two different optima. The turquoise cluster has its optimum at approx. 0.1, while the red cluster has its optimum at 1.0. For a human, finding these clusters graphically can be challenging, as the clusters found by the clustering approach are not trivial to observe.

TABLE I
SILHOUETTE COEFFICIENT OF THE TWO CLUSTERING APPROACHES FOR THE OPTIMIZABLE PARAMETERS.

Parameter	Brute-force	Feature-based
KF1 Step Size	0.74	0.12
KF1 Observe Noise Co-variance	0.71	-0.07
KF1 State Noise Co-variance	-0.21	-0.63
KF1 State Noise Coupling	0.01	-0.68
KF1 Initial Bounds Distance	0.20	-0.21
KF1 Bounds Factor	0.64	0.20
KF2 Step Size	0.74	0.02
KF2 Observe Noise Co-variance	0.73	0.04
KF2 State Noise Co-variance	0.72	-0.06
KF2 State Noise Coupling	0.72	0.13
KF2 Initial Bounds Distance	0.55	0.15
KF2 Bounds Factor	0.56	-0.03

Table I compares the silhouette coefficient of both clustering approaches for six different parameter clusterings of both Kalman filters. For the brute-force algorithm, we conclude that there are some parameters with good clustering results. However, there are also other examples, i.e., KF1 State Noise Co-variance and KF1 State Noise Coupling, where the resulting clusters are not very useful. For the feature-based clustering, we observe that the quality of the found clusters decreases. This is expected, as the feature-based clustering has less input information available; however, the accuracy degradation is unexpectedly drastic. Additionally, Figure 6 shows the found clusters of the feature-based algorithm of the same traces as in Figure 1. We can confirm our observation from Table I that the clustering of the feature-based approach is significantly worse than the brute force approach. Especially the clustering of the KF2 (right) looses its expressiveness in comparison to the clusters found by the brute force approach. For KF1 (left) however, we still observe a reasonable clustering, although there are some wrong assignments especially in the yellow and the turquoise clusters.

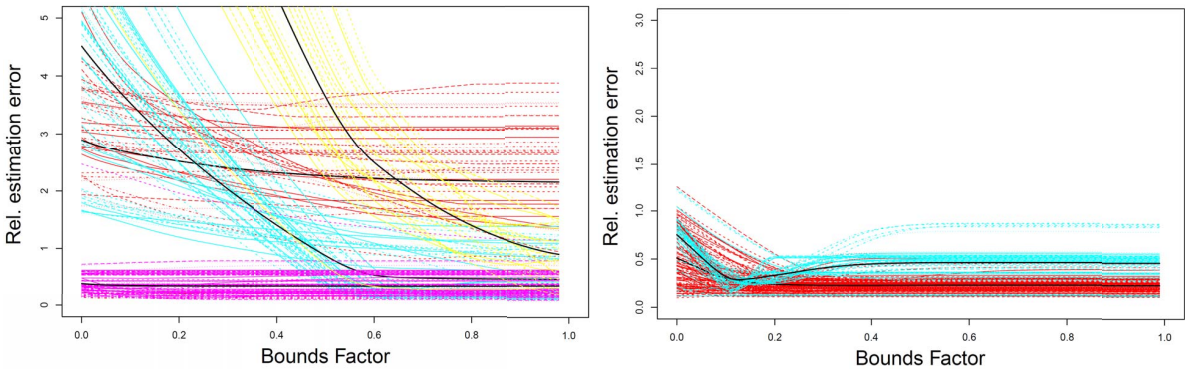

Fig. 5. Results of the brute force clustering for the bounds factor parameter of KF1 (left) and KF2 (right). Each color represents a different cluster, black lines represent the respective cluster means.

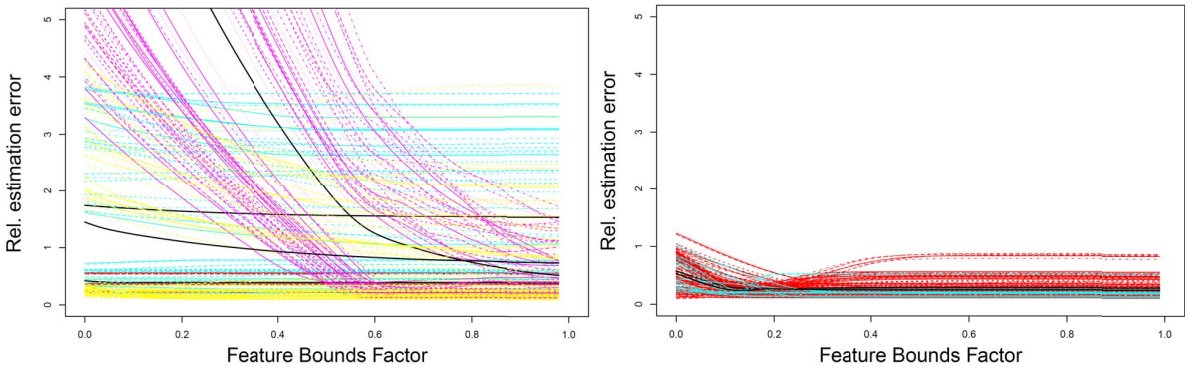

Fig. 6. Results of the feature-based clustering for the bounds factor parameter of KF1 (left) and KF2 (right). Each color represents a different cluster, black lines represent the respective cluster means.

C. Discussion

Summarizing, we can say that there is a clear trade-off between accuracy and required computation time. While the brute force approach produces significantly better clusters, it has severe conceptual disadvantages over the feature-based clustering, including higher and unpredictable computation time.

Both approaches determine different clusters in the optimization behaviors that can be used to systematically optimize the groups separately in order to improve estimation accuracy of the individual cluster. Additionally, both approaches are able to quickly optimize new traces, by assigning them to the closest cluster and applying the optimal parameter set for the respective cluster. The brute force clustering leads to better clusters and optimal parameter settings of that clusters, but takes significantly more time. The main advantage of the feature-based clustering is its lower and configurable run-time, as any optimization algorithm can be used to optimize the clusters separately.

VI. Future Work

We see a significant decrease of the cluster quality using the feature-based approach. We are currently investigating two possibilities to increase clustering accuracy. First, the performance of the feature-based approach naturally depends on the quality of the chosen features, the clustering algorithm and its hyper-parameters. Hence, an in-depth study of the different influencing factors could lead to significant accuracy gains. Second, in the current version, we calculate the pairwise euclidean difference of the optimization behavior. However, we are usually more interested in the relative increase or decrease in accuracy than the actual distance of the errors. Hence, a comparison based on the slope of the functions could be meaningful, i.e., of the first derivative.

Additionally, we want to investigate the trade-off between computation time and cluster quality in more depth by analyzing additional optimizable parameters as well as a larger set of measurement traces. By systematically analyzing the trade-off between computation time and increased accuracy, we want to derive a more conclusive assessment of the two approaches. A larger data set furthermore enables us to find further examples of clusters with opposed parameter optima. This helps to convincingly demonstrate the benefit of clustering traces.

The presented case study focuses on optimizing a single parameter at a time. However, in practice, the parameters can not be analyzed in isolation as they also impact each other. Therefore, the clustering – as well as the underlying optimization – should analyze the multiple parameter dimensions

together. This multi-dimensional cluster analysis practically prohibits the use of the brute force clustering, as the search space grows exponentially. Hence, increasing the effectiveness of the feature-based approach is of paramount importance.

Lastly, we want to transfer the proposed techniques to other domains of self-aware computing. Apart from the domain-specific feature set, the proposed techniques should be easily transferable.

VII. Conclusion

In this paper, we propose two different automated approaches of clustering measurement traces for resource demand estimation. This knowledge about clusters can be used to improve the hyper-parameter tuning of the different estimation approaches in a self-aware manner for the given set of traces by either (1) reducing the estimation error or (2) accelerating the online optimization of those traces. We present a preliminary case study that demonstrates how the clustering of trace groups can be performed. We use these results to derive a set of open issues and challenges to address in future work.

Acknowledgment

This work was co-funded by the German Research Foundation (DFG) under grant No. (KO 3445/11-1) and by Google Inc. (Faculty Research Award). The authors would like to thank Karsten Schaefers for his contribution of the experimental evaluation.

References

[1] D. A. Menascé, L. W. Dowdy, and V. A. F. Almeida, *Performance by Design: Computer Capacity Planning By Example*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004.

[2] A. Bauer, J. Grohmann, N. Herbst, and S. Kounev, "On the Value of Service Demand Estimation for Auto-Scaling," in *19th International GI/ITG Conference on Measurement, Modelling and Evaluation of Computing Systems (MMB 2018)*. Springer, February 2018.

[3] N. Huber, F. Brosig, S. Spinner, S. Kounev, and M. Bähr, "Model-Based Self-Aware Performance and Resource Management Using the Descartes Modeling Language," *IEEE Transactions on Software Engineering*, vol. 43, no. 5, pp. 432–452, 2017.

[4] R. H. Reussner, S. Becker, J. Happe, R. Heinrich, A. Koziolek, H. Koziolek, M. Kramer, and K. Krogmann, *Modeling and Simulating Software Architectures: The Palladio Approach*. MIT Press, 2016.

[5] S. Kounev, P. Lewis, K. Bellman, N. Bencomo, J. Camara, A. Diaconescu, L. Esterle, K. Geihs, H. Giese, S. Götz, P. Inverardi, J. Kephart, and A. Zisman, "The Notion of Self-Aware Computing," in *Self-Aware Computing Systems*, S. Kounev, J. O. Kephart, A. Milenkoski, and X. Zhu, Eds. Berlin Heidelberg, Germany: Springer Verlag, 2017.

[6] S. Spinnner, J. Grohmann, S. Eismann, and S. Kounev, "Online model learning for self-aware computing infrastructures," *Journal of Systems and Software*, vol. 147, pp. 1 – 16, 2019.

[7] F. Willnecker, M. Dlugi, A. Brunnert, S. Spinner, S. Kounev, and H. Krcmar, "Comparing the Accuracy of Resource Demand Measurement and Estimation Techniques," in *EPEW 2015*, ser. Lecture Notes in Computer Science, M. Beltrán, W. Knottenbelt, and J. Bradley, Eds., vol. 9272. Springer, August 2015, pp. 115–129.

[8] J. Rolia and V. Vetland, "Parameter estimation for performance models of distributed application systems," in *CASCON '95*. IBM Press, 1995, p. 54.

[9] F. Brosig, S. Kounev, and K. Krogmann, "Automated Extraction of Palladio Component Models from Running Enterprise Java Applications," in *VALUETOOLS '09*, 2009, pp. 1–10.

[10] W. Wang, X. Huang, X. Qin, W. Zhang, J. Wei, and H. Zhong, "Application-Level CPU Consumption Estimation: Towards Performance Isolation of Multi-tenancy Web Applications," in *IEEE CLOUD 2012*, Jun. 2012, pp. 439 –446.

[11] T. Zheng, C. Woodside, and M. Litoiu, "Performance Model Estimation and Tracking Using Optimal Filters," *IEEE TSE*, vol. 34, no. 3, pp. 391–406, May 2008.

[12] S. Spinner, G. Casale, F. Brosig, and S. Kounev, "Evaluating Approaches to Resource Demand Estimation," *Perform. Evaluation*, vol. 92, pp. 51 – 71, October 2015.

[13] C. Bezemer, S. Eismann, V. Ferme, J. Grohmann, R. Heinrich, P. Jamshidi, W. Shang, A. van Hoorn, M. Villavicencio, J. Walter, and F. Willnecker, "How is Performance Addressed in DevOps?" in *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering, ICPE 2019, Mumbai, India*, 2019, pp. 45–50.

[14] J. Grohmann, N. Herbst, S. Spinner, and S. Kounev, "Self-Tuning Resource Demand Estimation," in *Proceedings of the 14th IEEE International Conference on Autonomic Computing (ICAC 2017)*, July 2017.

[15] J. Grohmann, S. Eismann, and S. Kounev, "The Vision of Self-Aware Performance Models," in *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*, April 2018, pp. 60–63.

[16] J. Rolia and V. Vetland, "Correlating resource demand information with ARM data for application services," in *Proceedings of the 1st international workshop on Software and performance*. ACM, 1998, pp. 219–230.

[17] G. Pacifici, W. Segmuller, M. Spreitzer, and A. Tantawi, "CPU demand for web serving: Measurement analysis and dynamic estimation," *Perform. Evaluation*, vol. 65, no. 6-7, pp. 531–553, 2008.

[18] G. Casale, P. Cremonesi, and R. Turrin, "Robust Workload Estimation in Queueing Network Performance Models," in *Euromicro PDP 2018*, Feb. 2008, pp. 183–187.

[19] ——, "How to Select Significant Workloads in Performance Models," in *CMG Conference Proceedings*, 2007.

[20] C. Stewart, T. Kelly, and A. Zhang, "Exploiting nonstationarity for performance prediction," *SIGOPS Oper. Syst. Rev.*, vol. 41, pp. 31–44, Mar. 2007.

[21] T. Zheng, J. Yang, M. Woodside, M. Litoiu, and G. Iszlai, "Tracking time-varying parameters in software systems with extended Kalman filters," in *CASCON '05*. IBM Press, 2005, pp. 334–345.

[22] D. Kumar, A. Tantawi, and L. Zhang, "Real-time performance modeling for adaptive software systems," in *VALUETOOLS '09*, 2009, pp. 1–10.

[23] W. Wang, X. Huang, Y. Song, W. Zhang, J. Wei, H. Zhong, and T. Huang, "A statistical approach for estimating cpu consumption in shared java middleware server," in *IEEE COMPSAC, 2011*. IEEE, 2011, pp. 541–546.

[24] S. Kraft, S. Pacheco-Sanchez, G. Casale, and S. Dawson, "Estimating service resource consumption from response time measurements," in *VALUETOOLS '09*, 2009, pp. 1–10.

[25] A. B. Sharma, R. Bhagwan, M. Choudhury, L. Golubchik, R. Govindan, and G. M. Voelker, "Automatic request categorization in internet services," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, pp. 16–25, Aug. 2008.

[26] J. Grohmann, N. Herbst, S. Spinner, and S. Kounev, "Using Machine Learning for Recommending Service Demand Estimation Approaches," in *Proceedings of the 8th International Conference on Cloud Computing and Services Science (CLOSER 2018)*, INSTICC. SciTePress, March 2018, pp. 473–480.

[27] Q. Noorshams, D. Bruhn, S. Kounev, and R. Reussner, "Predictive performance modeling of virtualized storage systems using optimized statistical regression techniques," in *ACM/SPEC ICPE 2013*, ser. ICPE '13. New York, NY, USA: ACM, 2013, pp. 283–294.

[28] M. Kutner, C. Nachtsheim, and J. Neter, *Applied Linear Regression Models*, ser. The McGraw-Hill/Irwin Series Operations and Decision Sciences. McGraw-Hill Higher Education, 2003.

[29] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, 1967, pp. 281–297.

[30] L. Kaufman and P. J. Rousseeuw, "Clustering by means of medoids, statistical data analysis based on the l1 norm and related methods," *edited by Y. Dodge, North-Holland*, pp. 405–416, 1987.

[31] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[32] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53 – 65, 1987.