2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) | 978-1-6654-1219-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICSE-Companion52605.2021.00107

# White-Box Performance-Influence Models: A Profiling and Learning Approach (Replication Package)

Max Weber
Leipzig University
Germany

Sven Apel
Saarland University
Saarland Informatics Campus
Germany

Norbert Siegmund
Leipzig University
Germany

***Abstract*—These artifacts refer to the study and implementation of the paper 'White-Box Performance-Influence Models: A Profiling and Learning Approach'. In this document, we describe the idea and process of how to build white-box performance models for configurable software systems. Specifically, we describe the general steps and tools that we have used to implement our approach, the data we have obtained, and the evaluation setup. We further list the available artifacts, such as raw measurements, configurations, and scripts at our software heritage repository.**

## I. Research Overview

Many modern software systems are highly configurable, allowing the user to tune them for performance, energy consumption, and other non-functional properties. Current performance modeling approaches aim at finding performance-optimal configurations by building black-box models. While such models provide accurate estimates, they cannot pinpoint causes of observed performance behavior to specific code regions. This does not only hinder system understanding, but it also complicates tracing the influence of configuration options to individual methods.

We propose white-box performance-influence models [1], an approach that models configuration-dependent software properties, such as performance, at the method level. This allows us to predict the influence of configuration decisions on individual methods, supporting system understanding and performance debugging. To this end, our approach combines profiling information of runs of different software configurations with machine learning. By means of 9 real-world Java software systems, we demonstrate that our approach can efficiently identify configuration-relevant methods and learn accurate performance-influence models [2] at the method level.

To enable reproducibility of our approach, we show in Figure 1 the overview of all steps. First, we sample a set of configurations of the software system we want to analyze. Then, we use a coarse-grained profiler (jProfiler) to obtain profiling data for each sampled configuration. With this data we learn performance-influence models via classification and regression trees for all methods. This way, we identify methods that are highly configuration- and performance-sensitive, possibly causing inaccurate predictions. Finally, we re-profile these inaccurate methods with a fine-grained profiler (Kieker) to learn more accurate models.

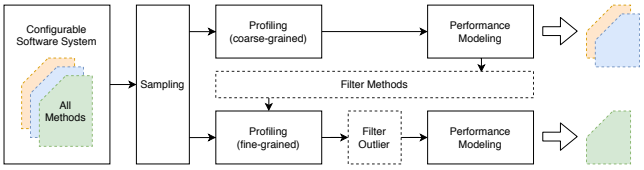


Fig. 1. Method-level white-box-modeling pipeline for configurable software systems

## II. Artifacts

We provide all necessary artifacts of our approach in a publicly available repository[1].

We have executed our approach on 9 real-world Java subject systems and make all binaries of the subject systems available to enable re-running our profiling experiments. Additionally, we provide the profiling results, such that the reproduction of only the learning step is possible as well.

To obtain the sample set from the 9 subject systems, we used the tool SPL Conqueror[2] [3]. In our experiments, we used feature-wise and pair-wise sampling. The corresponding configurations are also available at our repository.

For profiling the subject systems, we used the tools jProfiler[3] (for coarse-grained profiling) and Kieker[4] [4] (for fine-grained profiling). To run the performance measurements, we invested 19 years of CPU time. To shorten replication of our experiments, we provide all performance measurements in the mentioned repository.

The performance modeling part of our approach is responsible for learning a performance model per method based on the profiling data. The scripts and instructions are also available in the repository.

[1]Software Heritage: https://archive.softwareheritage.org/browse/revision/2e61f8ce57498194c2af0cd76e87498a174f07fa/
[2]SPLConqueror: https://github.com/se-sic/SPLConqueror
[3]jProfiler: https://www.ej-technologies.com/
[4]Kieker: http://kieker-monitoring.net/

978-1-6654-1219-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ICSE-Companion52605.2021.00107

We executed the experiments on a cluster of 27 desktop computers, each of them equipped with an Intel Quad-Core processor, an SSD running Ubuntu 18.04.03 LTS headless, an HDD to store the profiling data, and 8GB or 16GB of RAM (we grouped the cluster such that each group a consistent amount of RAM). To start and distribute the performance measurement experiments, we used the slurm workload manager[5].

## REFERENCES


[1] M. Weber, S. Apel, and N. Siegmund, "White-box performance-influence models: A profiling and learning approach," in *Proc. Int. Conf. Software Engineering (ICSE)*, 2021.

[2] N. Siegmund, A. Grebhahn, S. Apel, and C. Kästner, "Performance-influence models for highly configurable systems," in *Proc. Europ. Software Engineering Conference and ACM SIGSOFT Symp. Foundations of Software Engineering (ESEC/FSE)*. ACM, 2015, pp. 284–294.

[3] N. Siegmund, M. Rosenmüller, M. Kuhlemann, C. Kästner, S. Apel, and G. Saake, "Spl conqueror: Toward optimization of non-functional properties in software product lines," *Software Quality Journal*, vol. 20, no. 3, pp. 487–517, 2012.

[4] A. Van Hoorn, J. Waller, and W. Hasselbring, "Kieker: A framework for application performance monitoring and dynamic software analysis," in *Proc. Int. Conf. Performance Engineering (ICPE)*. ACM, 2012, pp. 247–248.


[5]Slurm: https://slurm.schedmd.com/