Providing Model-Extraction-as-a-Service for Architectural Performance Models

Jürgen Walter, Simon Eismann, Nikolai Reed, and Samuel Kounev
Chair of Software Engineering, University of Würzburg, Germany
(juergen.walter, simon.eismann, nikolai.reed, samuel.kounev)
@uni-wuerzburg.de

Abstract

Architectural performance models can be leveraged to explore performance properties of software systems during design-time and run-time. We see a reluctance from industry to adopt model-based analysis approaches due to the required expertise and modeling effort. Building models from scratch in an editor does not scale for medium and large scale systems in an industrial context. Existing open-source performance model extraction approaches imply significant initial efforts which might be challenging for layman users. To simplify usage, we provide the extraction of architectural performance models based on application monitoring traces as a web service. Model-Extraction-as-a-Service (MEaaS) solves the usability problem and lowers the initial effort of applying model-based analysis approaches.

1 Introduction

Architectural performance models can be leveraged to explore performance properties of software systems at design-time and run-time. While design-time analysis requires a fully parameterized model, approaches for proactive model-based resource management often depend on a model skeleton as input [10].

The creation of architectural performance models can be based on manual construction or scripting of model extraction code tailored to a specific application. Both requires significant effort which ranges between weeks to months for experienced performance engineers [2, 8].

We see a reluctance from industry about model-based analysis approaches due to modeling efforts including understanding and writing model creation scripts. We assume that building models in an editor from scratch does not scale in an industrial context due to modeling overhead.

To improve acceptance in industry, model-based approaches need to be easy to use. Novel model extraction approaches aim at automatic extraction of models from application performance monitoring data[7, 11, 12, 14]. However, applying research software is still challenging for layman users. Therefore, this paper proposes to provide MEaaS. At this, we explain the creation of model extraction web services based on our Performance Model Extractor (PMX) [14] tool. The web services receive application monitoring traces and (optional) parameters as input and returns an architectural performance model.

Provisioning of model extraction as a web service offers many benefits for users, developers, and maintainers of that service. Users of the web service circumvent cumbersome and error-prone set up of research software. Moreover, users do not have to update their software, as the web service always provides the latest software version. In contrast to running on multiple customer environments, maintenance needs to be done only for a single run-time environment. Researchers developing and maintaining performance model extraction tooling benefit, as we configured the web service to collect submitted monitoring traces as training data for model extraction. Furthermore, a web service allows to investigate how the software is used by external users. Based on the submitted monitoring traces we may learn and improve automated model extraction mechanisms. Finally, an increased number of users and experiments improves external validity of model extraction software.

2 State of the Art

Related approaches can be divided into (i) works providing modeling and simulation as services [4, 9] and (ii) approaches for model learning from application performance monitoring traces [7, 11, 12, 14].

There are several attempts to lower the effort for modeling and simulation tools [4, 9]. This includes Simulation-as-a-Service [9] as well as providing modeling tools as web services [4]. Recent works often provide services using REST interfaces and Docker container technology. None of the existing Software-as-a-Service (SaaS) solutions provides the extraction of architectural performance models based on application performance monitoring trace input.

In research, it is still quite popular to create performance models manually. While for some domains and applications this might be sufficient, we argue that this does not scale for medium and large-scale systems. Existing architectural performance model

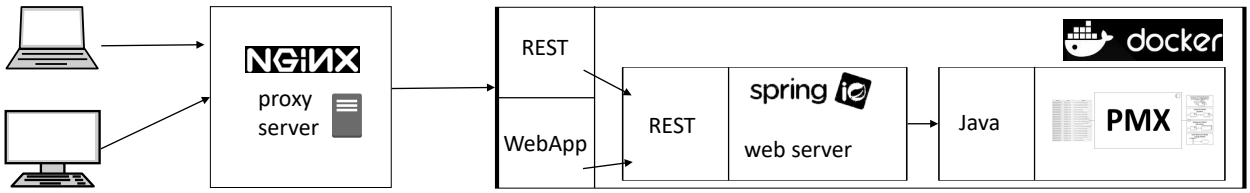

Figure 1: MEaaS Architecture.

extraction tools [7, 11, 12, 14] are either not open source or lack a public and easy to use description on how to setup and run model extraction. None of the performance model extraction approaches provides a public web service.

3 Model-Extraction-as-a-Service

The goal of the presented work is to relieve the performance engineer from the complexity of model extraction *and* setup. The non-web-service version of PMX [14] already provides performance engineers with a solution that integrates established tooling for monitoring and log processing [3], system model and call graph creation [5], and resource demand estimation [6]. The presented web service relieves the user from setup and updating.

3.1 Design

Figure 1 depicts the coarse-grained architecture of our web service. Our MEaaS includes several software components, which we describe in the following.

Java extraction module The Java extraction module transforms application performance monitoring data into architectural performance models. The model extraction logic, separating generic and formalism specific parts, has been described in [14]. Generic modeler extraction parts and formalism specific object creation routines can be packed into an executable jar file including all dependencies. The Java extraction module can be executed on the command line.

Spring I/O web server A web server allows to execute model extraction within a browser. The web interface provides an interaction layer to the user to interact with the model extraction layer (PMX) using a graphical interface. The web server contains

- REST service implementation
- HTTP pages user interface
- software to trigger model extraction service
- software to return results (file download)

We decided for the Spring Framework to set up the web server, as it allows for easy implementation of the REST API that is used to upload files, trigger the model extraction, and download. Although the framework does not impose any specific programming model, it has become popular in the Java community as an alternative to, replacement for, or even addition to the Enterprise JavaBeans (EJB) model.

Docker container The docker container simplifies and speeds up deployment by packing web server and java application into a single container. The docker container allows deploying the application and access it locally within a web browser. We decided for Docker technology, as it provides light-weight easy to deploy images.

NGINX proxy server The proxy provides an interface between external users and docker container and organizes communication. For example, it provides a reverse proxy to do port forwarding and a secure layer for https. We decided for NGINX because it is light-weight, easy to use, and very popular. By also providing load balancing functionality, NGINX allows for an increased PMX user base.

3.2 Usage

Input The model extraction requires Kieker [3] application monitoring traces as input. To receive accurate models, the traces have to contain monitoring information of a low load run (avoiding contention) or provide additional resource utilization information to improve resource demand estimation accuracy [6]. Additional parameters are optional and can be found at the web-service website.

Provided Web Services Kieker application performance monitoring traces and additional extraction parameters can be passed using two different services: First, we provide an interactive web interface including an upload dialog. Second, we also provide a pure REST interface suited to be used in automated processes. Passing parameters requires to expand the URL, for example:

```
http://extraction-service-url/conf?id=
traceurl/traces.zip&?core=server=4
```

The result of the REST call is a zip file containing several files describing the performance model (resource landscape, software architecture, deployment, repository) as well as logging information.

To illustrate and evaluate our approach, we provide web service implementations for the Descartes Modeling Language (DML)[13] and the Palladio Component Model (PCM) [1] modeling formalism. [1] Both for-

[1] https://descartes.tools/pmx/

malisms provide complementing toolchains. Palladio focuses on design-time analysis, while DML focuses on run-time scenarios.

3.3 Limitations

Performance Model Extraction Despite more than two years of research and development have been spent on the creation of PMX, there are several known limitations. We assume CPU bound processes. Memory bound processes have not been investigated so far. Also, the extraction of explicit parametric dependencies is not yet supported. In case performance prediction capabilities of extracted models do not meet accuracy requirements, the classical software performance engineering model refinement process can be applied.

Software as a Service There are open issues concerning SaaS and scalability. The model extraction might take longer than HTTP timeouts (especially if input traces are huge). Future web service versions should apply asynchronous communication to resolve this problem. There is a vulnerability for denial-of-service attacks. The large resource demand caused by a single model extraction request makes the system prone to such attacks. To cope with this security breach we might be forced to use login mechanisms. Privacy for monitoring data could be ensured running a separate service instance behind the firewall of a company.

4 Concluding Remarks

This paper presents MEaaS for architectural performance models. Based on an existing model extraction software, this paper presents additional software components to provide a web service. Provisioning as a web service reduces efforts to derive architectural performance models for all kinds of users. Moreover, a central service allows to collect monitoring traces and share them with the research community.

Acknowledgments This work is supported by the German Research Foundation (DFG) in the Priority Programme "DFG-SPP 1593: Design For Future—Managed Software Evolution" (KO 3445/15-1)

References

[1] S. Becker, H. Koziolek, and R. Reussner. "The Palladio Component Model for Model-driven Performance Prediction". In: *Journal of Systems and Software* 82.1 (Jan. 2009), pp. 3–22.

[2] N. Huber et al. "Performance Modeling in Industry: A Case Study on Storage Virtualization". In: *ACM/IEEE 32nd International Conference on Software Engineering (ICSE 2010), Software Engineering in Practice Track*. Cape Town, South Africa: ACM, Mar. 2010, pp. 1–10.

[3] A. van Hoorn, J. Waller, and W. Hasselbring. "Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis". In: *3rd ACM/SPEC Int. Conf. on Perf. Eng. (ICPE '12)*. 2012, pp. 247–248.

[4] E. Cayirci. "Modeling and simulation as a cloud service: A survey". In: *2013 Winter Simulations Conference (WSC)*. Dec. 2013, pp. 389–400.

[5] A. van Hoorn. *Model-Driven Online Capacity Management for Component-Based Software Systems*. Kiel Computer Science Series 2014/6. Dissertation, Faculty of Engineering, Kiel University. Kiel, Germany: Department of Computer Science, Kiel University, 2014.

[6] S. Spinner et al. "Evaluating Approaches to Resource Demand Estimation". In: *Performance Evaluation* 92 (Oct. 2015), pp. 51–71.

[7] P. C. Brebner. "Automatic Performance Modelling from Application Performance Management (APM) Data: An Experience Report". In: *Proceedings of the 7th ACM/SPEC International Conference on Performance Engineering, ICPE 2016, Delft, The Netherlands, March 12-16, 2016*. 2016, pp. 55–61.

[8] S. Lehrig and S. Becker. "Using Performance Models for Planning the Redeployment to Infrastructure-as-a-Service Environments: A Case Study". In: *12th International ACM SIGSOFT Conference on Quality of Software Architectures, QoSA 2016, Venice, Italy, April 5-8, 2016*. 2016, pp. 11–20.

[9] S. Shekhar et al. "A simulation as a service cloud middleware". In: *Annals of Telecommunications* 71.3 (Apr. 2016), pp. 93–108.

[10] S. Spinner, J. Walter, and S. Kounev. "A Reference Architecture for Online Performance Model Extraction in Virtualized Environments". In: *Proceedings of the 2016 Workshop on Challenges in Performance Methods for Software Development (WOSP-C'16) co-located with 7th ACM/SPEC International Conference on Performance Engineering (ICPE 2016)*. Delft, the Netherlands, Mar. 2016.

[11] F. Willnecker and H. Krcmar. "Optimization of Deployment Topologies for Distributed Enterprise Applications". In: *2016 12th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA)*. Apr. 2016, pp. 106–115.

[12] A. Brunnert and H. Krcmar. "Continuous performance evaluation and capacity planning using resource profiles for enterprise applications". In: *Journal of Systems and Software* 123 (2017), pp. 239–262.

[13] N. Huber et al. "Model-Based Self-Aware Performance and Resource Management Using the Descartes Modeling Language". In: *IEEE Transactions on Software Engineering (TSE)* 43.5 (2017).

[14] J. Walter et al. "An Expandable Extraction Framework for Architectural Performance Models". In: *Proceedings of the 3rd International Workshop on Quality-Aware DevOps (QUDOS'17)*. l'Aquila, Italy: ACM, Apr. 2017.