Self-Aware QoS Management in Virtualized Infrastructures

Samuel Kounev
Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany
kounev@kit.edu

Fabian Brosig
Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany
fabian.brosig@kit.edu

Nikolaus Huber
Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany
nikolaus.huber@kit.edu

ABSTRACT

We present an overview of our work-in-progress and long-term research agenda aiming to develop a novel methodology for engineering of *self-aware* software systems. The latter will have built-in *architecture-level* QoS models enhanced to capture dynamic aspects of the system environment and maintained automatically during operation. The models will be exploited at run-time to adapt the system to changes in the environment ensuring that resources are utilized efficiently and QoS requirements are satisfied.

Categories and Subject Descriptors

C.4 [**PERFORMANCE OF SYSTEMS**]: Modeling techniques

General Terms

Performance, Management

1. INTRODUCTION

With the increasing adoption of virtualization and the transition towards Cloud Computing platforms, modern enterprise software systems based on the SOA paradigm are becoming increasingly complex and dynamic. The lack of direct control over the underlying physical hardware and the complex interactions between the applications sharing the physical infrastructure pose some major challenges in providing Quality-of-Service (QoS) guarantees, e.g., performance, availability and reliability.

Service providers are often faced with questions such as: What would be the effect on the QoS of running applications if a new application is deployed in the virtualized infrastructure or an existing application is migrated from one physical server to another? How much resources need to be allocated to a newly deployed application to ensure that service-level agreements are satisfied? How should the system configuration (e.g., resource allocations, load balancing strategies) be adapted to avoid QoS issues or inefficient resource usage arising from changing customer workloads? Answering such *online QoS queries* requires the ability to predict at *run-time* how the QoS of running applications would be affected if the system configuration or the workload changes. We refer to this as *online QoS prediction*.

Predicting the QoS of a SOA application, even in an offline scenario, is a challenging task. A detailed *system QoS model* capturing the QoS-relevant aspects of both the software architecture and the multi-layered execution environment is needed. While a number of architecture-level model-based prediction techniques exist, e.g., [5], most of them suffer from two significant drawbacks which render them impractical for use at run-time: i) QoS models are expensive to build and provide limited support for reusability and customization, ii) QoS models are static and maintaining them manually during operation is prohibitively expensive [3, 6].

2. RESEARCH AGENDA & VISION

The Descartes Research Group[1] was started in July 2009, funded by the German Research Foundation (DFG) as well as by European projects and industrial partners. The group is working on a novel software and systems engineering methodology comprised of a set of methods, techniques and tools for the engineering of so-called *self-aware* software systems. The latter will have built-in online QoS prediction and self-adaptation capabilities addressing the challenges described in Sect. 1. Self-awareness is defined by the combination of three properties: i) *Self-reflective*: systems should be aware of their software architecture, execution platform and the hardware infrastructure on which they are running as well as of dynamic changes that occur during operation, ii) *Self-predictive*: systems should be able to predict the effect of dynamic changes (e.g., changing user workloads) on their behavior, as well as predict the effect of undertaken system adaptation actions in response to such changes, and iii) *Self-adaptive*: systems should proactively adapt as the environment evolves in order to ensure that their QoS requirements are continuously satisfied in a cost- and energy-efficient manner.

To realize this vision, we advocate the use of online models integrated into the system components and capturing the QoS-relevant system aspects during operation. The models will serve as a "mind" to the system controlling its behavior, i.e., resource allocations and scheduling decisions. In analogy to Descartes' *dualism principle*, the link between the models and the system components they represent will be bidirectional.

We are currently working on developing a new meta-model

[1] http://www.descartes-research.net

for online QoS models designed to encapsulate all information, both static and dynamic, relevant to predicting a system's QoS on-the-fly. While, initially, we are focusing on performance, availability and efficiency aspects, long term we are planning to consider further QoS properties such as reliability and fault-tolerance. Our meta-model is based on existing architecture-level[2] performance meta-models for component-based architectures surveyed in [5], in particular the Palladio Component Model (PCM) [1]. It captures the performance influences of the platforms used at each layer of the execution environment (e.g., hardware, virtualization, middleware) taking into account resource allocations.

Unlike conventional architecture-level QoS models, the instances of the meta-model we are developing will be *dynamic* in the sense that they will be maintained and updated automatically to reflect the evolving system environment. To realize this, we are working on enhancing execution platforms with functionality to automatically extract and maintain online models during operation. Depending on the type of system considered (e.g., newly developed system vs. legacy system) and the availability of monitoring and instrumentation frameworks, the degree of automation of the initial model extraction will vary.

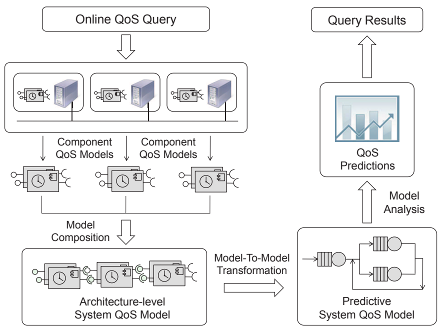

Figure 1: Online QoS Prediction Process

The online QoS models will be used during operation to answer QoS-related queries such as the ones discussed in Sect. 1. Fig. 1 illustrates the process that will be followed in order to provide an answer to a query. First, the QoS models of all involved system components will be retrieved and combined by means of model composition techniques into a single architecture-level QoS model encapsulating all information relevant to answering the QoS query. This model will then be transformed into a predictive QoS model by means of an automatic *model-to-model transformation*. The target predictive model type and level of abstraction as well as the solution technique will be determined on-the-fly based on the required accuracy and time available for the analysis.

The ability to answer online QoS queries during operation provides the basis for implementing techniques for self-aware QoS management. Such techniques will be triggered automatically during operation in response to observed or forecast changes in application workloads. The goal will be to proactively adapt the system to such changes in order to avoid anticipated QoS problems or inefficient resource usage. The adaptation will be performed in an autonomic fashion by considering a set of possible system reconfiguration scenarios (e.g, changing VM placement and/or resource allocations) and exploiting the online QoS query mechanism to predict the effect of such reconfigurations.

[2]We distinguish between *descriptive* architecture-level QoS models and *predictive* QoS models. The former describe QoS-relevant aspects of software architectures and execution environments (e.g., UML models augmented with QoS-related annotations). The latter capture the temporal system behavior and can be used for QoS prediction by means of analytical or simulation techniques (e.g., Markov chains, layered queueing networks or stochastic Petri nets).

The presented research agenda and vision lies at the intersection of several computer science disciplines including software architecture, computer systems modeling, autonomic computing, distributed systems, and more recently, Cloud Computing and Green IT.

3. CASE STUDIES

We are currently working on two case studies carried out in the environment depicted in Fig. 2. We study a complex Java EE application, the new SPECjEnterprise2010 benchmark, deployed in a cluster of Oracle WebLogic Server running in a virtualized environment. The aim of the first case study is to show how detailed architecture-level performance models can be extracted automatically at run-time based on online monitoring data [2]. As a performance model we use the PCM [1]. To validate the extraction method, we compared predictions derived from the extracted PCM models with measurements on the real system. The second case study shows how the extracted performance models can be exploited for self-adaptive performance management at run-time by adding and removing resources on the virtualization and middleware layer under varying workloads [4]. In addition to the overall vision and approach, the poster will present our latest results of the two case studies as of the time of the conference.

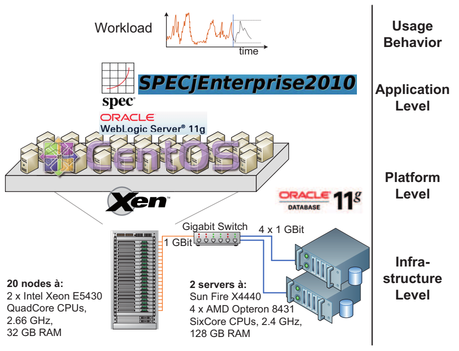

Figure 2: Experimental Environment

4. REFERENCES

[1] S. Becker, H. Koziolek, and R. Reussner. The Palladio component model for model-driven performance prediction. *Journal of Syst. and Softw.*, 82:3–22, 2009.

[2] F. Brosig, S. Kounev, and K. Krogmann. Automated Extraction of Palladio Component Models from Running Enterprise Java Applications. In *Proc. of ROSSA-2009*. ACM, 2009.

[3] J. Hellerstein. Engineering Autonomic Systems. In *Proc. of ICAC'2009*, pages 75–76. ACM, 2009.

[4] N. Huber, F. Brosig, and S. Kounev. Model-based Self-Adaptive Resource Allocation in Virtualized Environments. In *SEAMS'11*.

[5] H. Koziolek. Performance evaluation of component-based software systems: A survey. Performance Evaluation, 2009.

[6] M. Woodside, G. Franks, and D. Petriu. The future of software performance engineering. In *Future of Software Engineering (FOSE'07)*, 2007.