Towards self-aware performance and resource management in modern service-oriented systems

Samuel Kounev, Fabian Brosig, Nikolaus Huber and Ralf Reussner
Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany
{kounev,fabian.brosig,nikolaus.huber,reussner}@kit.edu

***Abstract*—Modern service-oriented systems have increasingly complex loosely-coupled architectures that often exhibit poor performance and resource efficiency and have high operating costs. This is due to the inability to predict at *run-time* the effect of dynamic changes in the system environment (e.g., varying service workloads) and adapt the system configuration accordingly. In this paper, we describe a long-term vision and approach for designing systems with built-in *self-aware* performance and resource management capabilities. We advocate the use of *architecture-level* performance models extracted dynamically from the evolving system configuration 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 performance requirements are continuously satisfied.**

I. Introduction

In today's data centers, IT services and applications based on the Service-Oriented Architecture (SOA) paradigm are typically hosted on dedicated server machines with over-provisioned capacity to ensure adequate performance at peak usage times. Driven by the pressure to improve energy efficiency and reduce operating costs, enterprises are increasingly adopting server virtualization and consolidation technologies. The adoption of virtualization, however, comes at the cost of increased system complexity and dynamicity. The increased complexity is caused by the introduction of virtual resources and the resulting gap between logical and physical resource allocations. The increased dynamicity is caused by the lack of direct control over the underlying physical hardware and by the complex interactions between the applications and workloads sharing the physical infrastructure. The inability to predict such interactions and adapt the system accordingly makes it hard to provide quality-of-service guarantees in terms of performance and availability. Service providers are often faced with questions such as: What performance would a new service deployed on the virtualized infrastructure exhibit and how much resources should be allocated to it? What would be the effect of migrating a service from one virtual machine (VM) to another? How should the system configuration be adapted to avoid performance problems arising from changing customer workloads? Answering such questions requires the ability to predict at *run-time* how the performance of running services would be affected if the system configuration or the workload changes. We refer to this as *online performance prediction*. Predicting the performance of a SOA application, however, even in an offline scenario is a challenging task. Consider the architecture of a typical modern service-oriented system as depicted in Figure 1. For a given set of hardware and software platforms at each layer of the architecture, Figure 1 shows some examples of the degrees of freedom at each layer and the factors that may affect the performance of SOA services. Predicting the performance of a service requires taking these factors into account as well as the dependencies among them. Therefore, a detailed *performance model* capturing the performance-relevant aspects of both the software architecture and the multi-layered execution environment is needed.

This work was funded by the German Research Foundation (DFG) under grant No. KO 3445/6-1.

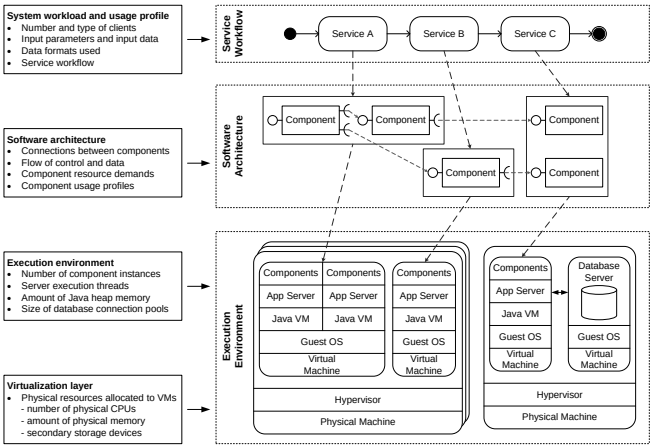

Figure 1. Modern Service-Oriented System

II. Performance Models

We distinguish between *descriptive* architecture-level performance models and *predictive* performance models. The former describe performance-relevant aspects of software architectures and execution environments (e.g., UML models augmented with performance annotations). The latter capture the temporal system behavior and can be used for performance prediction by means of analytical or simulation techniques (e.g., queueing networks). Over the past decade, a number of architecture-level performance meta-models have been developed by the performance engineering community, the most prominent examples being the UML SPT and MARTE profiles [1]. Other proposed meta-models include

CSM [2], PCM [3] and KLAPER [4]. Architecture-level performance models are built during system development and are used at design and deployment time to evaluate alternative system designs and/or predict the system performance for capacity planning purposes. While a number of model-based performance prediction techniques exist, most of them suffer from two significant drawbacks which render them impractical for use at run-time: i) models are expensive to build and provide limited support for reusability and customization, ii) models are static and maintaining them manually during operation is prohibitively expensive [5].

An attempt to address the first issue was made by recent efforts in the area of *component-based performance engineering* [6]. The latter deals with techniques and tools for building performance models of software components that are parameterized to explicitly capture the influences of the component execution context. While techniques for component-based performance engineering have contributed a lot to facilitate model reusability, there is still much work to be done on further parameterizing performance models before they can be used for online performance prediction. In particular, current techniques do not provide means to model the layers of the component execution environment (e.g., the virtualization layer) explicitly [7]. The performance influences of the individual layers, the dependencies among them and the resource allocations at each layer should be captured as part of the models.

As to the second issue indicated above, the heart of the problem is in the fact that *architecture-level* performance models are normally designed for *offline use* and as such they do not capture dynamic aspects of the environment. In a virtualized SOA environment changes are common, e.g., new services are deployed on the virtualized infrastructure, service workloads change, VMs are migrated between servers. Given the frequency of such changes, the amount of effort involved in maintaining performance models is prohibitive and therefore in practice such models are rarely used after deployment. Even though some techniques have been proposed to automatically track predictive performance models at run-time (e.g., [8]–[10]), such techniques abstract the system at a very high level without taking into account its software architecture and configuration.

III. Vision and Approach

We now present a long-term vision and approach for designing systems with built-in online performance prediction capabilities enabling *self-aware* performance and resource management. The idea is to make *architecture-level* performance models (e.g., PCM) usable at run-time by enhancing them to capture dynamic aspects of the environment and making them an integral part of the system. To achieve this, models should be integrated into the system components they represent and execution platforms should be enhanced with functionality to track dynamic changes in the environment and automatically maintain models during operation. The assumption is that the initial models are either built manually during system design or they are extracted at run-time based on online monitoring and measurement data [11]. We will refer to the new models as *dynamic service performance models* since they will be continuously updated, refined and calibrated during operation based on online monitoring and measurement data.

The new models should be designed to encapsulate all information, both static and dynamic, relevant to predicting a service's performance on-the-fly. This includes information about the service's software architecture, its workload and its execution environment. Current architecture-level performance models for component-based architectures, surveyed in [6] provide a foundation to build on. The models will be used to answer performance-related queries arising during operation such as the ones mentioned in Sect. I. We refer to such queries as *online performance queries*.

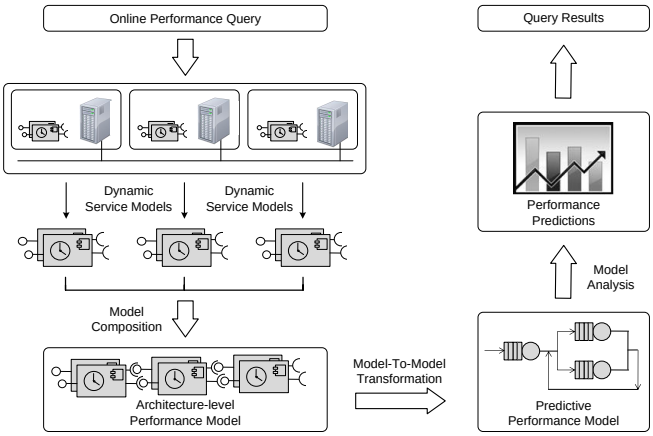

Figure 2. Online Performance Prediction Process

Figure 2 illustrates the process that will be followed in order to provide an answer to a query. First, the models of all involved services will be retrieved and combined by means of model composition techniques into a single architecture-level performance model encapsulating all information relevant to answering the performance query. This model will then be transformed into a predictive performance model by means of an automatic *model-to-model transformation*. Existing model-to-model transformations for static architecture-level performance models will be used as a basis, e.g., [2]–[4]. 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 the time available for the analysis. Different model types (layered queueing networks, queueing Petri nets and general-purpose simulation models) and model solution techniques (exact analytical techniques, numerical approximation techniques, simulation and bounding techniques) should be exploited here in order to provide flexibility in trading-off

between prediction accuracy and analysis overhead.

The ability to answer online performance queries during operation will provide the basis for implementing techniques for self-aware performance and resource management. Such techniques will be triggered automatically during operation in response to observed or forecast changes in service workloads. The goal will be to proactively adapt the system to such changes in order to avoid anticipated performance 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 performance query mechanism to predict the effect of such reconfigurations before making a decision. Figure 3 illustrates the online reconfiguration process. The latter is based on the generic model of a control loop from [12] which we have extended to integrate the use of the online performance query mechanism. In addition to the main control loop, two additional loops are running in the background, one for continuously refining and calibrating online models and one for forecasting the workload evolution.

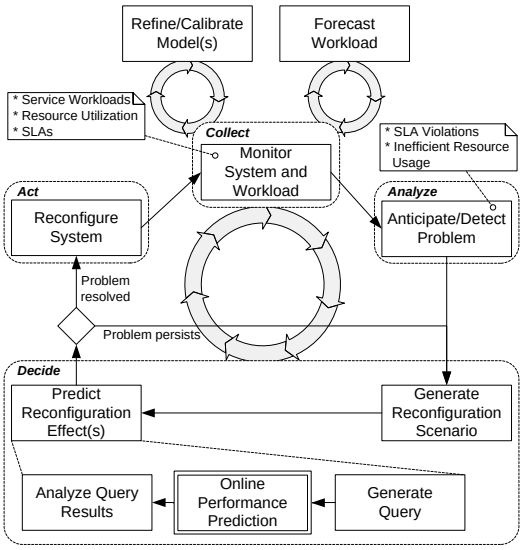

Figure 3. Online Reconfiguration Process

While model-based self-adaptation techniques have been studied in the software engineering and autonomic computing communities (see e.g. [8], [12]–[14]), the use of dynamic *architecture-level* performance models at run-time for online performance and resource management is a new research direction. The described approach raises several big challenges that will be subject of long-term fundamental research (the references below provide some further motivation and initial steps in the respective direction):

- Abstractions for modeling performance-relevant aspects of services in *dynamic virtualized* environments. Individual layers of the software architecture and execution environment [7], context dependencies and dynamic system parameters should be modeled explicitly.
- Automatic online model extraction, maintenance, refinement and calibration *during operation* [11], [15]. Models should be tightly coupled with the system components and maintained automatically at run-time.
- Efficient resolution of service context dependencies including dependencies between service input parameters, resource demands, invoked third-party services and control flow of underlying components [16].
- Automatic generation of predictive performance models for online performance prediction [15]. The model type and level of abstraction as well as the model solution technique should be determined automatically.
- Efficient heuristics exploiting the online performance prediction techniques for dynamic system reconfiguration and utility-based optimization [17].
- Techniques for self-aware performance and resource management guaranteeing SLAs while improving energy efficiency and lowering costs [12].

IV. Initial Steps

Initial steps towards the realization of the described long-term vision have been started as part of a new five-year research project [18] funded by the German Research Foundation (DFG). The project, named after René Descartes, aims to develop a set of novel techniques for *self-aware* performance and resource management in modern service-oriented systems. Self-awareness in this context is meant in the sense that systems should be aware of changes that occur in their environment (changing workloads or resource allocations) and should be able to predict the effect of such changes on their performance (*"thought is what happens in me such that I am immediately conscious of it" – René Descartes*). Furthermore, systems should automatically adapt as the environment evolves in order to ensure that infrastructure resources are utilized efficiently and performance requirements are continuously satisfied (*"for it is not enough to have a good mind: one must use it well" – René Descartes*). The idea is to use dynamic performance models that 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 mind controls the body, but the body can also influence the mind"*), the link between the performance models and their respective system components should be bidirectional. In the rest of this section, we describe two initial case studies that were carried out before the initiation of the Descartes Project as a preliminary proof-of-concept of the proposed research directions.

In the first case study, we studied a complex Java EE application showing how detailed architecture-level performance models can be extracted automatically at run-time based on online monitoring data [11]. The Java EE application we studied was a beta version of the new SPEC-

jEnterprise2010 standard benchmark deployed on Oracle WebLogic Server (WLS) with Oracle Database 11g. We employed the WebLogic Diagnostics Framework (WLDF) as a monitoring and instrumentation tool and the Palladio Component Model (PCM) [3] as a performance meta-model.

The extraction method we implemented has three main steps: i) the extraction of the effective application architecture, ii) the extraction of the performance-relevant control flow of components and iii) the extraction of resource demands. The extraction is based on trace data reflecting the observed call paths during execution as well as the measured response times and resource utilization. To validate the extraction method, we compared predictions derived from the extracted PCM models with measurements on the real system. We considered a number of different scenarios, on the one hand, varying the operation mix and throughput level under which the PCM models were extracted, and on the other hand, varying the operation mix and throughput level for which performance predictions were made. In all cases, the prediction error was between 20% and 30% [11]. Even though the current version of the extraction method is not 100% automated, the case study demonstrated that the existing gap between low-level monitoring data and high-level performance models can be closed.

As a second preliminary proof-of-concept demonstrating the benefits of online performance prediction, we conducted a case study of a SOA application running in a service-oriented Grid computing environment [19]. The latter was implemented using the Globus Toolkit middleware and the Xen virtualization platform. We augmented the Grid middleware with an online performance prediction component that can be called at run-time to predict the Grid performance for a given resource allocation and load-balancing strategy. The performance prediction was based on hierarchical queueing Petri net models dynamically composed at run-time to reflect the system configuration and workload. Based on the online performance prediction mechanism, we developed a methodology for designing autonomic QoS-aware resource managers that have the capability to predict the performance of the Grid components they manage and allocate resources in such a way that SLAs are honored. We conducted an extensive performance evaluation of our framework considering a number of different scenarios each focusing on selected aspects [19]. In all scenarios, the system was able to cope with variations in the service workloads and nearly all client SLAs were fulfilled. The results were very encouraging and demonstrated the benefits of online performance models for self-aware performance and resource management.

V. Conclusion

Architecture-level performance models provide a powerful tool for performance prediction, however, to make it possible to exploit this at run-time, models need to be integrated into the system components they represent and made to evolve together with the system environment. In this paper, we presented a vision and research agenda aiming to achieve this goal by developing novel techniques for building *dynamic* service performance models tightly coupled with the system components and automatically maintained during operation. The new models will provide the basis for implementing intelligent techniques for *self-aware* performance and resource management. We presented a roadmap to realize this vision and discussed two preliminary case studies conducted as initial steps in this direction. The proposed research direction promises a number of benefits such as better quality-of-service, lower operating costs and improved energy efficiency.

References

[1] Object Management Group (OMG), "UML SPT, v1.1 (January 2005) and UML MARTE (May 2006)."

[2] D. Petriu and M. Woodside, "An intermediate metamodel with scenarios and resources for generating performance models from UML designs," *Softw. and Syst. Modeling*, vol. 6, no. 2, pp. 163–184, 2007.

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

[4] V. Grassi, R. Mirandola, and A. Sabetta, "Filling the gap between design and performance/reliability models of component-based systems: A model-driven approach," *Jour. of Syst. and Softw.*, vol. 80, no. 4, pp. 528–558, 2007.

[5] J. L. Hellerstein, "Engineering autonomic systems," in *Proc. of the 6th Intl. Conf. on Autonomic Computing (ICAC)*. New York, NY, USA: ACM, 2009, pp. 75–76.

[6] H. Koziolek, "Performance evaluation of component-based software systems: A survey," *Perform. Eval.*, 2009, in Press.

[7] M. Hauck, M. Kuperberg, K. Krogmann, and R. Reussner, "Modelling layered component execution environments for performance prediction," in *CBSE'09*, 2009.

[8] M. Woodside, T. Zheng, and M. Litoiu, "Service System Resource Management Based on a Tracked Layered Performance Model," in *ICAC'06*, 2006, pp. 175–184.

[9] D. Menascé, H. Ruan, and H. Gomaa, "QoS management in service-oriented architectures," *Perform. Eval.*, vol. 64, no. 7-8, pp. 646–663, 2007.

[10] T. Israr, M. Woodside, and G. Franks, "Interaction tree algorithms to extract effective architecture and layered performance models from traces," *Journal of Sys. and Soft.*, vol. 80, no. 4, pp. 474–492, 2007.

[11] 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.

[12] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, "Software Engineering for Self-Adaptive Systems: A Research Roadmap," in *Software Engineering for Self-Adaptive Systems, LNCS 5525*, 2009, pp. 1–26.

[13] T. Vogel, S. Neumann, S. Hildebrandt, H. Giese, and B. Becker, "Model-driven architectural monitoring and adaptation for autonomic systems," in *ICAC'09*, 2009, pp. 67–68.

[14] F. Irmert, T. Fischer, and K. Meyer-Wegener, "Runtime adaptation in a service-oriented component model," in *SEAMS'08*, 2008, pp. 97–104.

[15] M. Woodside, G. Franks, and D. C. Petriu, "The future of software performance engineering," in *FOSE'07*, 2007.

[16] H. Koziolek, "Parameter Dependencies for Reusable Performance Specifications of Software Components," Ph.D. dissertation, University of Oldenburg, 2008.

[17] D. A. Menascé, J. M. Ewing, H. Gomaa, S. Malex, and J. P. Sousa, "A framework for utility-based service oriented design in SASSY," in *WOSP/SIPEW'10*, 2010, pp. 27–36.

[18] "Descartes Project," http://www.descartes-research.net, 2010.

[19] R. Nou, S. Kounev, F. Julia, and J. Torres, "Autonomic QoS control in enterprise Grid environments using online simulation," *Jour. of Syst. and Softw.*, vol. 82, 2009.