Online Performance Prediction with Architecture-Level Performance Models

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

Abstract: Today's enterprise systems based on increasingly complex software architectures often exhibit poor performance and resource efficiency thus having high operating costs. This is due to the inability to predict at run-time the effect of changes in the system environment and adapt the system accordingly. We propose a new performance modeling approach that allows the prediction of performance and system resource utilization online during system operation. We use architecture-level performance models that capture the performance-relevant information of the software architecture, deployment, execution environment and workload. The models will be automatically maintained during operation. To derive performance predictions, we propose a tailorable model solving approach to provide flexibility in view of prediction accuracy and analysis overhead.

1 Introduction

Modern enterprise systems based on the Service-Oriented Architecture (SOA) paradigm have increasingly complex architectures composed of loosely-coupled services that are subjected to time-varying workloads and evolve independently. In such dynamic environments, managing system resources so that end-to-end performance requirements are met while resources are efficiently utilized is a challenge. Due to the inability to keep track of dynamic changes in the system environment and predict their effect, today's SOA systems often exhibit poor performance and resource efficiency thus having high operating costs.

System usage profiles, software deployments or execution environments change over time: Services are subjected to time-varying workloads. New services are deployed or existing services are newly composed. The service-providing components are deployed on distributed, heterogeneous system environments including different middleware configurations and virtualization layers. System administrators are often faced with questions such as: What performance would a new service deployed on the virtualized infrastructure exhibit? What would be the effect of migrating a service from one virtual machine to another? How should the workloads of a new service and existing services be partitioned among the available resources so that performance objectives are met and resources are utilized efficiently? How should the system configuration be adapted to avoid anticipated performance problems arising from growing customer workloads? In order to provide for an efficient resource and performance management honoring service level agreements (SLAs), the ability to predict resource utilization and system performance on-line, i.e.,

continuously during system operation, is essential.

Existing approaches to run-time performance prediction (e.g., [MG00, NKJT09]) are based on predictive performance models such as (layered) queueing networks or queueing petri nets abstracting the system at a high level without taking the software architecture or, e.g., the virtualization layer into account. These approaches are often restricted to homogeneous servers, single workload classes and abstract individual services as black boxes.

Approaches to performance prediction at design and deployment time (e.g., [BKR09, SLC+05, Obj]) are mostly based on architecture-level models that are annotated with descriptions of the system's performance-relevant behavior. The underlying modeling languages account for modeling systems at different levels of abstraction and detail as well as modeling performance-relevant context dependencies between execution environments, software deployments and usage profiles. However, most of them suffer from a significant drawback which render them impractical for use at run-time: Given that they are designed for off-line use, the models are *static*, i.e., do not capture the dynamic aspects of systems. Maintaining them manually during operation is prohibitively expensive [Hel09].

We will provide *dynamic service performance models* that describe performance-relevant system design abstractions and are automatically maintained during operation, thus can capture also the dynamic aspects of modern enterprise system environments, where configuration and deployment changes are common. For instance, new services are deployed on-the-fly, virtual machines are migrated or servers are consolidated during operation. The dynamic service performance models will include all information relevant to predict a service's performance on-the-fly. The models will be tied to their deployment platform in order to keep them up to date. A deployment platform will provide interfaces describing their performance-relevant aspects and have built-in self-monitoring functionality. Following [WFP07], we propose a convergence of performance monitoring, modeling and prediction as interrelated activities.

The following goals will be pursued: i) Development of a meta-model for dynamic service performance models capturing static and dynamic performance-relevant information about the software architecture and deployment, the execution environment and the workload. ii) Development of a methodology to integrate the dynamic service performance models with their deployment platforms. The focus will be on automated model maintenance including model parameter estimation. iii) Development of techniques for solving the dynamic service models. To derive performance predictions from dynamic service models at run-time, the models will be transformed into predictive performance models that are then solved by analytical or simulation techniques. We will use multiple solving techniques to provide flexibility in view of prediction accuracy and analysis overhead.

The proposed thesis is part of the Descartes Research Project [KBHR10] which envisions a new kind of SOA platforms that are self-aware of their performance and resource efficiency. They are self-aware in the sense that they will be aware of changes in their environment and be able to predict the effects of such changes on their performance. They will automatically adapt to ensure that system resources are utilized efficiently and performance requirements are continuously satisfied.

2 Related Work

A survey of model-based performance prediction techniques was published in [BDMIS04]. A number of techniques utilizing a range of different performance models have been proposed including queueing networks (e.g., [MG00]), layered queueing networks (e.g., [WZL06]), queueing Petri nets (e.g., [Kou06]), stochastic process algebras [GHKM05], statistical regression models (e.g., [EFH04]) and learning-based approaches (e.g., [EEM10]). Those models capture the temporal system behavior without taking into account its software architecture and configuration. We refer to them as predictive performance models.

To describe performance-relevant system aspects at a different abstraction level, there are a number of meta-models for describing architecture-level performance models specifying the performance and resource-demanding behavior of software systems [Koz09]. The most prominent meta-models are the UML SPT and MARTE profiles [Obj], both of which are extensions of UML as the de facto standard modeling language for software architectures. The architecture-level performance models are built during system development and are used at design and deployment time to evaluate alternative system designs or predict the system performance for capacity planning purposes. Over the past decade, with the increasing adoption of component-based software engineering [CSSW05], the performance evaluation community has focused on supporting component-based systems. Proposed meta-models include SPE-MM [SLC$^+$05], CSM [PW07], KLAPER [GMS07] and PCM [BKR09]. In the latter, the authors advocate an explicit context model as part of the component specification that captures the dependencies of functional and extra-functional properties on the components connections to other components, the execution environment, the allocated hardware and software resources and the usage profile. A modeling notation is proposed allowing component developers to explicitly specify the influence of parameters on the component resource demands as well as on their usage of external services. Component service specifications describe the service's behavior and control flow in an abstract and parametric manner.

The common goal of these architecture-level modeling efforts is to enable the automated transformation into predictive performance models making it possible to predict the system performance and resource consumption for a fixed workload and configuration scenario. However, the existing architecture-level performance models are normally designed for offline use, assume a static system architecture and do not support modeling dynamic environments [KBHR10]. In a virtualized environment changes are common, e.g., service workloads change over time, new services are deployed, or VMs are migrated between servers. The amount of effort involved in maintaining performance models is prohibitive and therefore in practice such models are rarely used after deployment.

3 Goals and Approach

We aim at establishing service-providing systems that are aware of changes in their environment (e.g., service workload, software deployment and execution environment) and

have the ability to predict the effect of those changes on their performance. We intend to provide a new method for online performance prediction by integrating dynamic service performance models with system execution environments. In this section, the main goals and tasks are discussed.

Meta-model for Dynamic Service Performance Models We will develop a meta-model for dynamic service performance models capturing static and dynamic performance-relevant information about service workload, the service's software architecture and the execution environment. Since we intend to keep meta-model instances continuously up to date during system operation, we refer to the models as *dynamic* service models. The modeling language should make it possible to specify information at different levels of abstraction and granularity, since we will use the models in different scenarios with different goals and constraints ranging from quick performance bounds analysis to accurate performance prediction.

Suitable abstractions for capturing the dynamic aspects of multi-layered execution environments (virtualization, operating system, Java virtual machine, application server) have to be developed. The meta-model should allow to describe the layers (e.g., middleware, virtualization) and their performance-influence explicitly [HKKR09] to facilitate modeling an exchange of individual execution layers.

We will use the PCM [BKR09] as a basis, given that it provides support for explicitly modeling the performance-influencing factors of services, i.e., their implementation, execution environment, workload and external service calls. PCM will be augmented by dynamic model parameters and model entities allowing the specification of dynamic system behavior.

Automated Model Maintenance at Run-time We will develop a methodology to integrate dynamic service models with service execution environments. This will include a technique to extract the models semi-automatically based on monitoring data. In order to keep the dynamic service performance models up to date during system operation, we develop methods to automatically maintain them at run-time. Therefore, we plan to integrate the models with the system, i.e., we will connect the performance-relevant system properties with the model instances.

We develop a platform-neutral interface for communicating which components should be monitored, according to which performance metrics should be measured and how the measurement data needs to be processed. This includes application-level monitoring (e.g., method response times), middleware-level monitoring (e.g., database connections) as well as monitoring at operating systems or virtualization layers (e.g., CPU utilization). Given that we observe the system during operation, techniques reducing the monitoring overhead such as sampling and filtering need to be considered.

In particular, the estimation of service resource demands will be of great importance for our approach. In the literature, there exist already many different resource demand estimation methods differing in their accuracy, their robustness and their applicability. For instance, there are notable differences in the amount and type of measurement data that

is required as method input. We aim at providing a resource demand estimation approach that supports tailoring towards specific constraints like available measurement data or the time horizon. For that reason, we work on a classification scheme which will facilitate evaluating the existing resource demand estimation approaches regarding their benefits and drawbacks.

Automated Performance Prediction To derive performance predictions from dynamic service models at run-time, the models have to be composed and transformed (in an automated manner) into a predictive performance model that is then solved by analytical or simulation techniques. The predictive performance models will be generated by means of model transformations using one or more dynamic service models as input. We intend to use multiple existing model solving techniques to provide flexibility in terms of prediction accuracy and analysis overhead. For instance, for a detailed performance analysis a fine-grained simulation could be conducted, whereas an analytical solver based on simple queueing network models or even a coarse-grained bounds analysis would be more appropriate in cases where a quick response is required.

Tailoring towards required prediction accuracy and timing constraints could be accomplished by selecting the appropriate predictive performance model type and level of model granularity on-the-fly. Similarly, the way the predictive performance model is solved, by analysis or simulation, can be varied. We intend to use existing predictive performance models, existing model-to-model transformations, existing simulations and existing analytical techniques that we plan to assemble to a tailorable service model solving process. The methods will be selected based on their expressiveness, overhead, scalability and tool support.

4 Summary

In order to provide for an efficient resource and performance management of modern service-oriented systems, the ability to predict resource utilization and system performance online during system operation is essential. To enable performance predictions in such environments, we propose to use architecture-level performance models that capture the performance-relevant aspects of both the software architecture and the virtualized, multi-layered execution environment. The models will be automatically maintained at system run-time by integrating them with the deployment platform. For performance predictions, we propose a tailorable model solving approach that provides flexibility in view of prediction accuracy and analysis overhead.

References

[BDMIS04] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni. Model-Based Performance Prediction in Software Development: A Survey. *IEEE Trans. Softw. Eng.*, 30(5), 2004.

[BKR09] S. Becker, H. Koziolek, and R. Reussner. The Palladio component model for model-driven performance prediction. *Journal of Systems and Software*, 82, 2009.

[CSSW05] I. Crnkovic, H. Schmidt, J. Stafford, and K. Wallnau. Automated Component-Based Software Engineering. *Journal of Systems and Software*, 74(1), 2005.

[EEM10] Ahmed Elkhodary, Naeem Esfahani, and Sam Malek. FUSION: a framework for engineering self-tuning self-adaptive software systems. In *Proc. of FSE'10*. ACM, 2010.

[EFH04] E. Eskenazi, A. Fioukov, and D. Hammer. Performance Prediction for Component Compositions. In *Proc. of CBSE*, 2004.

[GHKM05] S. Gilmore, V. Haenel, L. Kloul, and M. Maidl. Choreographing Security and Performance Analysis for Web Services. In *Proc. of EPEW and WS-FM*, LNCS. 2005.

[GMS07] V. Grassi, R. Mirandola, and A. Sabetta. Filling the gap between design and performance/reliability models of component-based systems: A model-driven approach. *Journal on Systems and Software*, 80(4), 2007.

[Hel09] J. L. Hellerstein. Engineering autonomic systems. In *Proc. of the 6th Intl. Conf. on Autonomic Computing (ICAC)*, pages 75–76, New York, NY, USA, 2009. ACM.

[HKKR09] M. Hauck, M. Kuperberg, K. Krogmann, and R. Reussner. Modelling Layered Component Execution Environments for Performance Prediction. In *Proc. of CBSE*, 2009.

[KBHR10] S. Kounev, F. Brosig, N. Huber, and R. Reussner. Towards self-aware performance and resource management in modern service-oriented systems. In *Proc. of IEEE Intl Conf. on Services Computing (SCC 2010), USA*. IEEE Computer Society, 2010.

[Kou06] S. Kounev. Performance Modeling and Evaluation of Distributed Component-Based Systems using Queueing Petri Nets. *IEEE Trans. Softw. Eng.*, 32(7), July 2006.

[Koz09] H. Koziolek. Performance evaluation of component-based software systems: A survey. *Performance Evaluation*, 2009.

[MG00] D. A. Menascé and H. Gomaa. A Method for Design and Performance Modeling of Client/Server Systems. *IEEE Trans. Softw. Eng.*, 26(11), 2000.

[NKJT09] R. Nou, S. Kounev, F. Julia, and J. Torres. Autonomic QoS control in enterprise Grid environments using online simulation. *Journal of Systems and Software*, 82(3), 2009.

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

[PW07] D. Petriu and M. Woodside. An intermediate metamodel with scenarios and resources for generating performance models from UML designs. *Software and Systems Modeling*, 6(2), 2007.

[SLC+05] C. U. Smith, C. M. Llado, V. Cortellessa, A. Di Marco, and L. G. Williams. From UML Models to Software Performance Results: an SPE Process based on XML Interchange Formats. In *In Proc. of Intl. Works. Softw Perf. (WOSP)*, 2005.

[WFP07] M. Woodside, G. Franks, and D. C. Petriu. The Future of Software Performance Engineering. In *FOSE '07: 2007 Future of Software Engineering*, 2007.

[WZL06] M. Woodside, T. Zheng, and M. Litoiu. Service System Resource Management Based on a Tracked Layered Performance Model. In *ICAC'06*, 2006.