Towards Online Performance Model Extraction in Virtualized Environments*

Simon Spinner[1], Samuel Kounev[1], Xiaoyun Zhu[2], and Mustafa Uysal[2]

[1] Karlsruhe Institute of Technology (KIT)
{simon.spinner,kounev}@kit.edu
[2] VMware Inc.
{xzhu,muysal}@vmware.com

Abstract. Virtualization increases the complexity and dynamics of modern software architectures making it a major challenge to manage the end-to-end performance of applications. Architecture-level performance models can help here as they provide the modeling power and analysis flexibility to predict the performance behavior of applications under varying workloads and configurations. However, the construction of such models is a complex and time-consuming task. In this position paper, we discuss how the existing concept of virtual appliances can be extended to automate the extraction of architecture-level performance models during system operation.

1 Introduction

Modern IT systems have increasingly complex layered architectures composed of loosely-coupled components deployed in virtualized environments. The use of virtualization provides increased flexibility and efficiency by enabling the sharing of resources among independent applications. However, managing the end-to-end performance of applications in virtualized environments while ensuring efficient resource usage is a challenge due to the increased system complexity and dynamics. Questions such as the following arise frequently during operation: How quickly and at what granularity (e.g., vCores, virtual machine instances) should resources be allocated/deallocated to applications as workloads change? How much resources are required to ensure both efficient operation and compliance with Service Level Agreements (SLAs)? To answer such questions, it is crucial to be able to predict *at run-time* the system performance under varying workloads and system configurations, so that resource allocations can be adapted dynamically to enforce SLAs while optimizing efficiency.

Existing approaches to online performance and resource management are typically based on coarse-grained performance models abstracting applications and system layers at a high level. Individual effects and complex interactions between the application workloads and the system layers are considered as static

* This work was funded by VMware Inc. We acknowledge the many fruitful discussions with Pradeep Padala.

and viewed as a black box. This hinders fine-grained performance predictions that are necessary for efficient resource management (e.g., predicting the effect on the response time, if a virtual machine of an application tier is replicated or migrated). Therefore, newer approaches to online performance and resource management (e.g. DMM [1,2]) are based on the more powerful *architecture-level* performance models for fine-grained performance predictions. However, building architecture-level performance models that accurately capture the different aspects of system behavior is a time-consuming and challenging task when applied manually to large real-world systems [3]. Often, no explicit architecture documentation of the system exists, and hence, the model must be built from scratch. Additionally, experiments and measurements must be conducted to parameterize and calibrate the model, such that it reflects the system behavior accurately. Moreover, a major challenge is to ensure that models derived based on measurements of the system in an offline setting would be representative of the actual system behavior in the real production environment. Given the high costs of building performance models, techniques for automated model extraction based on observation of the system at run-time are highly desirable.

The contributions of this position paper are: *a*) we describe an extension of the notion of virtual appliance with integrated logic for performance model extraction, *b*) we propose an approach for how an end-to-end architecture-level performance model can be obtained in virtualized environments with a heterogeneous software stack, and *c*) we present a research roadmap for implementing the proposed approach. The paper is structured as follows: Section 2 gives a brief overview of related work in the field of automatic model extraction and of our preliminary work. Section 3 describes the vision and the approach in detail and identifies research challenges.

2 State-of-the-Art

Related Work Current performance monitoring and management tools in industry (e.g., Hyperic or Dynatrace Diagnostics) can provide large amounts of raw performance data, however, they lack the ability to generate performance abstractions of the monitored systems and applications. Approaches such as [4,5] use systematic measurements to build black-box mathematical models. However, they only serve as interpolation of the measurements. Predictive performance models are extracted for example in [6], where run-time monitoring data is used to derive the model parameters of predefined queueing Petri net models. Extraction of structural information is considered for example for UML sequence diagrams [7], and for LQNs [8,9].

Existing work on extracting architecture-level performance models is either based on static code analysis or assumes a strictly controlled environment. In [10], behavior models are extracted via static and dynamic analysis, however, this is done in an offline setting requiring fine-grained manual instrumentation of applications. The described approaches are focused on the application level and do not explicitly consider the influences of the lower system layers. To quantify

the impact of the virtualization platform on the application performance, micro-benchmarks are used in [11,12]. However, no explicit model of the performance influence of the virtualization platform is proposed.

Preliminary Work The approach we propose in this position paper is based on the experiences we gained in our preliminary work. In [1,2], we describe the Descartes Meta-Model (DMM) which is an architecture-level modeling language for online performance and resource management. It enables to describe the performance influence of different system layers in independent sub-models, which can be automatically composed at run-time enabling online performance prediction. The combined model can be automatically transformed to different alternative underlying stochastic models (queueing networks, stochastic Petri nets, and fine-grained custom simulation models), which in turn can be solved using different solution techniques (exact analytical techniques, numerical approximation techniques, simulation and bounding techniques). While DMM provides a powerful and flexible tool for online predictions, the manual creation of these models can be complex and time-consuming. Therefore in [13], we investigated the feasibility of extracting architecture-level performance models at system run-time. We used low-level monitoring data obtained through application instrumentation to extract an architecture-level performance model of the SPECjEnterprise2010 standard benchmark. While the resulting models were able to predict the system performance within an acceptable error margin (mostly 10-20 percent) [13], this approach has two major drawbacks, limiting its practical applicability: (i) the extraction is focused on the application level and does not construct detailed models of the lower layers of a system (e.g., virtualization and middleware), and (ii) the approach is restricted to a specific software stack (i.e., Java EE application server, WebLogic Diagnostic Framework for the application instrumentation). The former limits the prediction accuracy of the extracted models in virtualized environments and their usage for configuration-based what-if analysis. The latter hinders its application in heterogeneous environments with different software stacks. The goal of the proposed approach is to overcome these limitations and enable the automatic model extraction in virtualized environments with heterogeneous software stacks.

3 Vision and Approach

Vision To simplify the creation and maintenance of architecture-level performance models, we envision a novel class of virtualization platforms with integrated capabilities for the automatic extraction of such models at system run-time. We assume an environment where a *virtual machine monitor (VMM)* hosts a set of *virtual appliances (VA)* with heterogeneous software stacks. VAs are pre-packaged VM images each containing a complete software stack ready to run on a virtualization platform. VAs are becoming increasingly popular in system management since they significantly reduce the effort and knowledge needed for deploying software systems. For instance, there are VAs available providing a

pre-configured Tomcat application server or Zimbra collaboration server. These VAs are built by experts of the respective system and can then be shared with others (e.g., through online marketplaces, such as VMware Solution Exchange[3]).

We argue that the notion of a VA should be extended to include additional logic for extracting performance models of the application as well as the middleware layers during run-time. A performance engineer, who has expertise in performance modeling, can specifically design the extraction logic for the respective software stack. When such a VA is deployed in a virtualized environment, the model extraction logic will start to monitor the application serving real production workloads and will automatically built a performance model of the VA.

A virtualization platform that is aware of the model extraction logic within the VA can then exploit the extracted performance models for online performance and resource management. However, to evaluate the performance impact of changes at the VMM level, we also need a model of the performance-relevant factors of the VMM and their influence on the VAs. Therefore, the VMM also needs to be extended with the capability of creating such models, so that an end-to-end performance model of the VA and the VMM can be extracted.

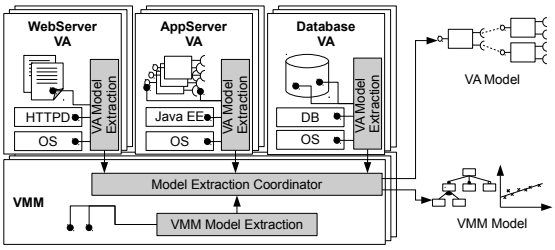

Fig. 1. Overview of proposed architecture

Approach Figure 1 gives an overview of the proposed architecture. The major components are VA Model Extraction, VMM Model Extraction and Model Extraction Coordinator. The *VA Model Extraction* component is delivered as part of each VA. It contains pre-defined model skeletons that capture structural knowledge of the software stack, a set of monitoring probes, and an executable extraction process. The extraction process describes how to compose and parameterize an end-to-end performance model of the VA based on model skeletons, static configuration data (e.g., server configuration files or deployment descriptors) and dynamic monitoring data (e.g., call path traces). Typically, an extraction process consists of the extraction of the static and dynamic architecture (e.g., application components, active and passive resources, inter- and intra-component control flow) and the model parameterization (e.g., resource demands, and branching probabilities). The degree to which this information is known beforehand and can be integrated as model skeletons, heavily depends on the type of VA. For instance, if the VA contains a complete application (e.g.,

[3] `https://solutionexchange.vmware.com/store`

a wiki or a mail server), the architecture can be provided beforehand and at run-time it is only necessary to estimate the model parameters for the current environment. In contrast, in case of a Java EE application server, the creator of the VA does not know the applications that will run on top of it. Therefore, he needs to integrate logic to determine the current application components and instrumentation probes to observe the control flow of the application.

The *VMM Model Extraction* is tightly coupled with the VMM and observes its internal state and configuration to build a model that captures the overhead of the VMM and contention effects due to the sharing of physical resources. We plan to derive regression-based models describing the overhead and contention effects depending on the current utilization of the physical resources and the VMM configuration (e.g., caps, priority and affinity settings of the scheduler). The models will be extracted based on online monitoring data provided by the VMM. If necessary, we also consider to use micro-benchmarks in order to determine certain performance characteristics of the VMM (e.g., to determine the overhead for certain workload mixes). Such micro-benchmarks can either be run in an initial step, when installing a new virtual host, or during system operation in phases of low workload intensity.

The *Model Extraction Coordinator* controls the model extraction components in the VMM and VAs and triggers the initial extraction or the update of the performance models. It also validates the extracted models continuously by comparing the model predictions with observations on the real system. If the predictions deviate significantly from the actual performance, the current model will be updated by repeating the model parameterization or changing the model structure. Furthermore, it monitors the state of the environment and triggers the model extraction process if it observes any changes (e.g., configuration changes in the VA or the VMM).

The extracted models of the VMM and VA are based on the Descartes Meta-Model (DMM) [1,2]. The latter allows to dynamically compose the automatically extracted submodels of the VA and the VMM in order to answer configuration-based what-if questions. Using DMM as the output model for the model extraction offers the flexibility to employ different analysis techniques for model solution depending on the required accuracy and speed.

Research Challenges The described approach raises a number of research challenges targeted as part of our on-going work:

- A generic mechanism to package the model extraction logic in the VAs needs to be defined including an interface for exchanging information between the VMM and VAs during model extraction.
- New languages to simplify the implementation of the model extraction for various VAs will be designed (e.g., for specifying instrumentation probes in a technology-agnostic manner, or for specifying rules for abstracting the control flow of an application).
- Techniques for reliably estimating resource demands in virtualized systems are necessary (e.g., influence of virtualization effects, and parallel processing on multi-core processors).

- Methods for autonomic online validation and calibration of performance models are crucial to ensure the representativeness of the extracted models.
- Methods to quantify the performance influence of the virtualization platform during system operation are necessary to extract the VMM models.
- Automatic techniques to detect configuration changes and to determine their influence on the performance models are desirable.

References

1. Brosig, F., Huber, N., Kounev, S.: Architecture-Level Software Performance Abstractions for Online Performance Prediction. Elsevier Science of Computer Programming Journal (SciCo) (2013)
2. Huber, N., van Hoorn, A., Koziolek, A., Brosig, F., Kounev, S.: Modeling Run-Time Adaptation at the System Architecture Level in Dynamic Service-Oriented Environments. Service Oriented Computing and Applications (2013) In print.
3. Kounev, S.: Performance Modeling and Evaluation of Distributed Component-Based Systems Using Queueing Petri Nets. IEEE Trans. on Softw. Eng. **32**(7) (2006) 486–502
4. Westermann, D., Happe, J.: Towards Performance Prediction of Large Enterprise Applications Based on Systematic Measurements. In: Proc. of the 15th Intl. Workshop on Component-Oriented Programming. (2010)
5. Courtois, M., Woodside, M.: Using Regression Splines for Software Performance Analysis. In: Proc. of the 2nd Intl. Works. on Software and Performance. (2000)
6. Kounev, S., Bender, K., Brosig, F., Huber, N., Okamoto, R.: Automated Simulation-Based Capacity Planning for Enterprise Data Fabrics. In: 4th Intl. ICST Conf. on Simul. Tools and Techniques . (2011)
7. Briand, L.C., Labiche, Y., Leduc, J.: Toward the Reverse Engineering of UML Sequence Diagrams for Distributed Java Software. IEEE Trans. on Softw. Eng. **32**(9) (2006) 642 – 663
8. Hrischuk, C.E., Woodside, M., Rolia, J.A., Iversen, R.: Trace-Based Load Characterization for Generating Performance Software Models. IEEE Trans. on Softw. Eng. **25**(1) (1999) 122 – 135
9. Israr, T., Woodside, M., Franks, G.: Interaction Tree Algorithms to Extract Effective Architecture and Layered Performance Models from Traces. J. Syst. Softw. **80**(4) (2007) 474–492
10. Krogmann, K., Kuperberg, M., Reussner, R.: Using Genetic Search for Reverse Engineering of Parametric Behaviour Models for Performance Prediction. IEEE Trans. on Softw. Eng. **36**(6) (2010) 865–877
11. Wood, T., Cherkasova, L., Ozonat, K., Shenoy, P.: Profiling and Modeling Resource Usage of Virtualized Applications. In: Proc. of the 9th ACM/IFIP/USENIX Intl. Conf. on Middleware. (2008)
12. Lu, L., Zhang, H., Jiang, G., Chen, H., Yoshihira, K., Smirni, E.: Untangling Mixed Information to Calibrate Resource Utilization in Virtual Machines. In: Proc. of the 8th ACM Intl. Conf. on Autonomic Computing. (2011)
13. Brosig, F., Huber, N., Kounev, S.: Automated Extraction of Architecture-Level Performance Models of Distributed Component-Based Systems. In: 26th IEEE/ACM Intl. Conf. On Automated Softw. Eng. (2011)