A Generic Approach for Architecture-Level Performance Modeling and Prediction of Virtualized Storage Systems

Qais Noorshams, Andreas Rentschler, Samuel Kounev, Ralf Reussner
Chair for Software Design and Quality
Karlsruhe Institute of Technology
Karlsruhe, Germany
{noorshams, rentschler, kounev, reussner}@kit.edu

ABSTRACT

Virtualized environments introduce an additional abstraction layer on top of physical resources to enable the collective resource usage by multiple systems. With the rise of I/O-intensive applications, however, the virtualized storage of such shared environments can quickly become a bottleneck and lead to performance and scalability issues. The latter can be avoided through careful design of the application architecture and systematic capacity planning throughout the system life cycle. In current practice, however, virtualized storage and its performance-influencing design decisions are often neglected or treated as a black-box. In this work-in-progress paper, we propose a generic approach for performance modeling and prediction of virtualized storage systems at the software architecture level. More specifically, we propose two performance modeling approaches of virtualized systems. Furthermore, we propose two approaches how the performance models can be combined with architecture-level performance models. The goal is to cope with the increasing complexity of virtualized storage systems with the benefit of intuitive software architecture-level models.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling techniques;
D.2.11 [**Software Engineering**]: Software Architectures

Keywords

I/O, Storage, Performance, Virtualization

1. INTRODUCTION

In recent years, virtualization technology has been widely adopted. According to analysts, this development has yet to reach its peak. Over the period 2012–2016, the server virtualization market is expected to grow annually by more than 31% [18]. In a parallel development, the I/O resource demands of modern IT systems have grown exponentially over the past couple of decades [17]. Until the year 2020, the amount of digital data is expected to double every year, 40% of which is expected to be either processed or stored in the cloud [5]. The danger of this trend is, however, that the virtualized storage of shared environments becomes a bottleneck, thus leading to significant performance and scalability issues as there are many performance-influencing factors. To avoid such issues, performance modeling and evaluation techniques are needed for capacity planning at deployment time as well as to guarantee performance requirements during operation upon changes in the workload.

In current practice, however, virtualized storage and its performances influences on the overall system performance are often neglected or treated as a black-box due to their complexity. Since traditional storage systems were not designed for the intricate data consumption in virtualized environments [7], virtualized storage systems have evolved significantly to sophisticated systems with multiple caching and virtualization layers.

Existing approaches model the storage performance of I/O-intensive applications in virtualized environments, e.g., [1, 10], however, with validation limited to basic environments or with focus on consolidation scenarios only. Moreover, it is difficult to include such models into software architecture models, because the required information between the two modeling abstraction levels needs to be synchronized. Software architecture models are on a higher abstraction level making them simpler and more intuitive to use. However, they may have several access points to the storage resources, thus, having mutual performance effects with low-level storage models. A media streaming portal, for instance, can have multiple different requests at the software architecture level, e.g., watermarking and media downloading. Those requests translate to intertwined read and write accesses at the storage system.

In this work-in-progress paper, we propose a generic approach for performance modeling and prediction of virtualized storage systems and combining high-level software architecture models with low-level storage models. More specifically, we propose a black- and a white-box performance modeling approach for virtualized storage systems. Furthermore, we propose two approaches how the performance models can be combined with architecture-level performance models. The goal is to cope with the increasing complexity of virtualized storage systems with the benefit of intuitive software architecture models. The overall methodology, however, is generic and can be applied for enhancing any software architecture models by more complex resource or subsystem models.

ICPE'13, April 21–24, 2013, Prague, Czech Republic.

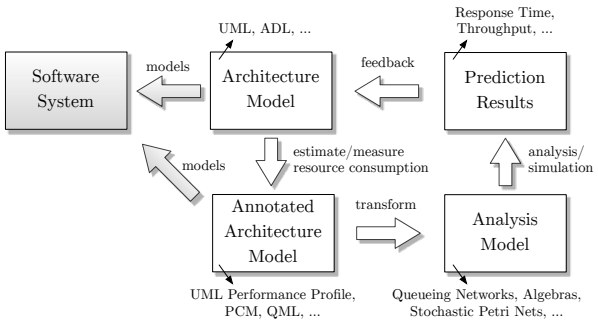


Figure 1: Model-based Performance Prediction (Source: [2])

2. PROPOSED APPROACH

In this section, we first present the background of our work originated from model-based performance prediction. Then, we present our performance modeling methodology as well as our combination approach integrating performance models of virtualized storage systems into software architecture models.

2.1 Background

In general, our approach can be applied to any performance-aware software architecture models. We apply our proposed approach to the Palladio Component Model (PCM) [3], a modeling language for component-based software architectures realizing a model-based performance prediction approach. The PCM is aligned with the component-based software engineering (CBSE) development process and provides modeling constructs to describe the i) software components, ii) system architecture, iii) resource allocations, and iv) usage profile of a component-based software system.

Using model transformations, e.g., to queueing networks or performance prototypes, a PCM model instance can be analytically solved or simulated to predict the system performance. The performance prediction serves as feedback and enables a model-based quality assessment of software systems. This process is illustrated in Figure 1.

2.2 Performance Modeling Methodology

Illustrated in Figure 2, in this section we show how our approach fits into the model-based performance prediction approach by first analyzing performance-influencing factors and then creating black- and white-box storage analysis models, i.e., performance prediction models. The storage analysis models are combined with the software architecture level model in the next section.

2.2.1 System Under Study

As a proof-of-concept of our approach, we study a representative virtualized environment based on the IBM mainframe *System z* and the storage system *DS8700*. They are state-of-the-art high-performance virtualized systems with redundant or hot swappable resources for high availability. The System z combined with the DS8700 represents a typical virtualized environment that can be used as a building block of cloud computing infrastructures. It supports on-demand elasticity of pooled resources with a pay-per-use accounting system (cf. [12]). The System z provides processors and memory, whereas the DS8700 provides storage space.

2.2.2 Performance-Influencing Factors

Aligned with our modeling methodology, in [16] we identified and evaluated the performance-influencing factors of applications deployed in the considered environment. The factors comprise both workload-specific characteristics and system-specific configuration parameters. An important aspect is to find an appropriate abstraction level for the factors. On the one hand, too coarse-grained factors lead to inaccuracies in the performance models. On the other hand, too fine-grained factors cannot be specified by software developers and deployers. Furthermore, we develop a fully automated approach to extract the workload characteristics for a given application. Tools like *blktrace*[1] are able to provide detailed workload-level information, e.g., the read/write ratio and the requested block addresses. Based on this information, we calculate important request metrics, e.g., the average size. Moreover, we develop a heuristic classification strategy to identify multiple, possibly intertwined, sequential accessing clients, since this has a significant performance impact [16].

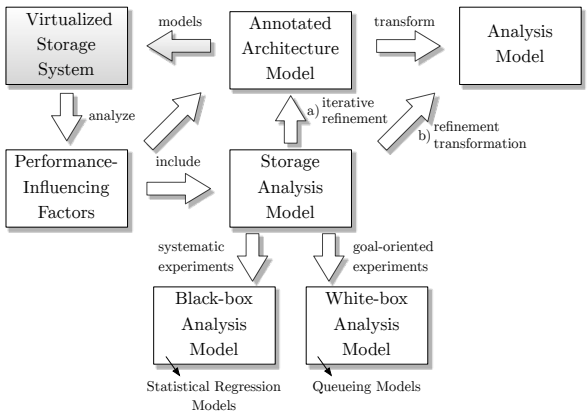


Figure 2: Proposed Approach

2.2.3 Black-box Analysis Model

In [15] we created black-box analysis models by applying multiple statistical regression techniques on systematic measurements. Regression techniques are very powerful in terms of prediction accuracy for interpolation scenarios. Furthermore, we proposed a general heuristic optimization algorithm to optimize the parameters of the regression techniques. The main advantage of our black-box modeling approach is that it is fully automated as part of our tool *SPA* [4], thus, lifting the burden for performance engineers to manually develop the models. We are working on extending our automated methodology to further evaluate the approach with another load-driver and in another system environment. Moreover, we plan to use the automatically extracted workload characterization, which will be integrated into SPA, as input for the regression-based models to be able to predict arbitrary workloads, cf. Figure 3. This approach aims at creating the black-box models once and map different applications to the required black-box model input parameters.

2.2.4 White-box Analysis Model

The black-box modeling approach requires systematic experiments that cover the whole parameter space. To create performance models with a set of goal-oriented experiments and gain further understanding of the system environment, we develop an iterative white-box performance modeling methodology and model our virtualized storage system in a queueing network, cf. Figure 4. Our approach follows the

[1] `http://www.cse.unsw.edu.au/~aaronc/iosched/doc/blktrace.html`

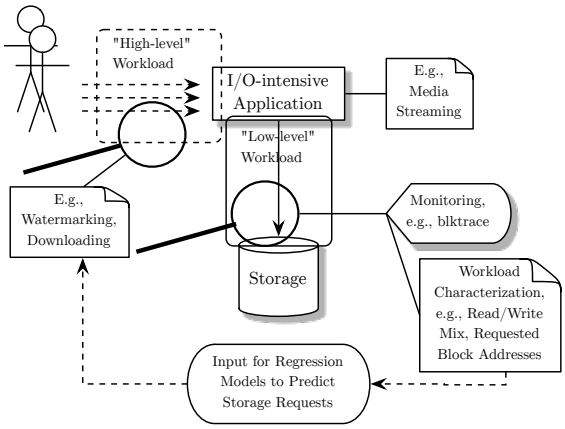


Figure 3: Analysis Model Generalization Approach

common generic steps in classical performance engineering comprised of performance model *creation*, *calibration*, and *validation* established by existing approaches [13, 14]. We first analyze the system environment identifying the system-specific performance-relevant aspects including the generic performance-influencing factors. Furthermore, we identify the workload scenarios we want capture as part of the model initially. We then create the model calibrated with experiments that we iteratively refine to account for more complex scenarios. Our preliminary results model the caching and the RAID array resource of our environment in homogeneous queueing models. The mean service times μ of the service stations are parameterized and calibrated based on response time measurements, such that

$$\mu(t,s,d) = c_1\, ts \cdot \ln(d) + c_2\, t \cdot \ln(d) + c_3\, s \cdot \ln(d) + c_4\, ts + c_5\, t + c_6\, s + c_7 \ln(d) + c_8; \quad c_i \in \mathbb{R},$$

where t is the number of simultaneous requests (clients), s is the request sizes, and d is the overall amount of data the workload operates on. Shown in Figure 5, the results demonstrate the high prediction accuracy of our white-box models for both interpolation and extrapolation (or consolidation) scenarios. Figure 5a shows the relative error for 1200 completely random configurations. The mean error (depicted as cross) is 7.22% and 2.39% for read and write requests, respectively. Figure 5b shows the relative error for 480 workload consolidation scenarios with 2 and 3 VMs. Here, the mean error is 8.22% and 2.74% for read and write requests, respectively. Next, we plan to extend the queueing models to account for mixed workloads. Furthermore, we plan to generalize the white-box model similar to the black-box model generalization approach.

2.3 Combination Approach

We propose two approaches to combine the low-level storage models with high-level software architecture models, *iterative refinement* and *refinement transformations*. In both approaches, it is important to design the *storage interface* between the two modeling abstraction levels and integrate the interface into the architecture model.

2.3.1 Storage Interface Design

The storage interface includes the required dynamic information (i.e., the workload-specific performance-influencing factors) in the software architecture model that is passed as input to the storage analysis model (e.g., the distinction

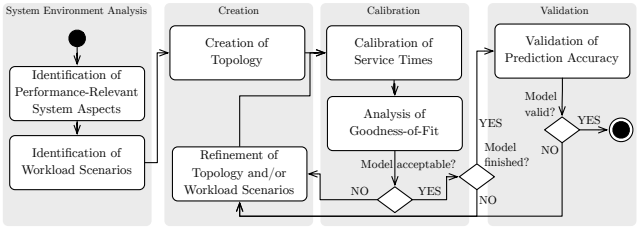


Figure 4: Iterative White-box Modeling Methodology

between read and write requests and their access pattern). Furthermore, the set of static storage-relevant information (i.e., the system-specific performance-influencing factors) is used to configure or initialize the storage analysis model (e.g., the chosen I/O scheduler). At this point, the main research question is what to include in the storage interface and at which abstraction level.

2.3.2 Iterative Refinement

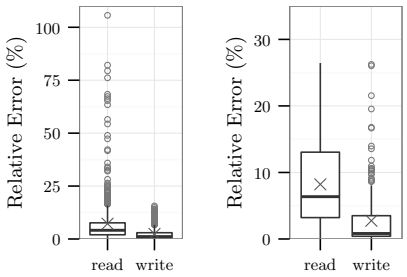
An iterative refinement approach allows for combining a specific software architecture model instance with the storage analysis model. The software architecture model instance contains multiple accesses to different active and passive resources (e.g., thread pools) and, more specifically, contains storage resource consumptions that are represented by a parametric delay, e.g., when simulating the architecture model instance. The architecture model instance is coupled with the storage analysis model as follows (cf. Figure 6):

1. The storage delays are initialized with a small value.
2. The architecture model instance is simulated to obtain the performance characteristics of the model instance. During the simulation, the storage accesses of the model instance over the storage interface are monitored to extract the storage workload profile.
3. The workload profile and the static storage-relevant information is used as input for the storage analysis model to obtain specific storage delays for the architecture model.
4. The storage delays in the simulation are updated with the delays obtained by the storage analysis model.
5. Step 2–4 is repeated until the performance characteristics of the model instance converge.

The caveat is, however, that the performance characteristics are not guaranteed to converge. Moreover, it is an open question to what extent this approach can be automated.

2.3.3 Refinement Transformations

A second approach is to include the storage analysis model with refinement transformations. Illustrated in Figure 7, the transformations include both i) a *completion transformation* [19] as well as ii) an *analysis transformation*. The



(a) Interpolation (b) Consolidation

Figure 5: Prediction Accuracy of Queueing Models

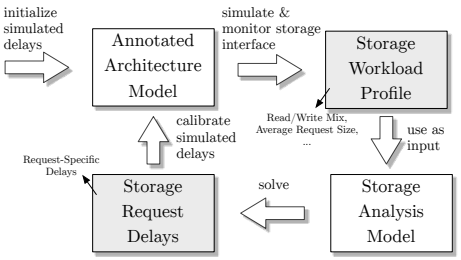


Figure 6: Iterative Refinement Approach

completion transformation refines the architecture model to fill in the static storage-relevant information and is configured, e.g., using feature tree models, cf. [8]. The analysis transformation uses the architecture model and creates the target analysis model to predict the system performance, cf. Figure 1. This transformation is extended to combine the target analysis model with the storage analysis model over the storage interface using a bridge or adapter. Generally, this approach allows for an arbitrary target analysis model. However, the conceptual interoperability of different modeling formalisms is still subject to research. Thus, if different formalisms are to be combined, their interoperability is required to be well-defined.

3. RELATED WORK

Several existing approaches have been proposed for performance analysis of I/O-intensive applications in virtualized environments. However, none of these approaches considers the system performance at the software architecture level as realized in the approach presented in this paper. Most related, Kraft et al. [10] propose a queueing network-based approach to predict the I/O performance of consolidated virtual machines. They monitor and model low-, block-level I/O requests. Their approach is focused on predicting consolidation scenarios. Further, Ahmad et al. [1] analyze the I/O performance in VMware's ESX Server virtualization. They compare virtual against native performance using benchmarks. They create mathematical models scaling the native performance down for I/O throughput predictions in virtualization. To analyze performance interference in a virtualized environment, Koh et al. [9] manually run CPU bound and I/O bound benchmarks. While they develop mathematical models for prediction, they explicitly focus on the consolidation of different types of workloads, i.e., CPU and I/O bound. By applying machine learning techniques, Kundu et al. [11] use artificial neural networks and support vector machines for dynamic capacity planning in virtualized environments. In [6], Gulati et al. present a study on storage workload characterization in virtualized environments, however, they do not propose any performance prediction techniques.

4. CONCLUSION

In this work-in-progress paper, we proposed a methodology for performance modeling and analysis of a) I/O-intensive applications in b) virtualized environments at the c) software architecture level. To the best of our knowledge, there are no other approaches that combine a), b), and c) to our extent. For this approach, we will extend the Palladio Component Model, a model-based performance prediction approach for component-based software architectures. For performance modeling, we create both black-box models based on statistical regression techniques and white-box models based on queueing networks. We apply our approach in real world environments as the IBM System z and the IBM DS8700. Our main contribution is that we provide a generic approach for combining software architecture level performance models with resource-specific storage models. More specifically, our contributions are the following:

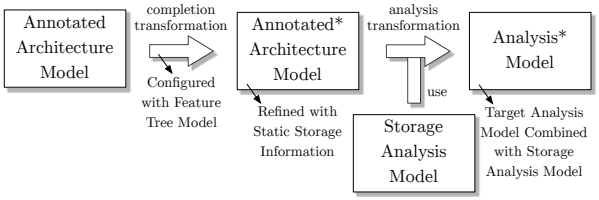


Figure 7: Refinement Transformation Approach

– We analyze virtualized storage systems and identify the performance-influencing factors.
– We create both black- and white-box analysis models of virtualized storage systems.
– We combine high-level software architecture models and low-level storage performance models.

Acknowledgments This work was funded by the German Research Foundation (DFG) under grant No. RE 1674/5-1 and KO 3445/6-1. We especially thank the Informatics Innovation Center (IIC) – `http://www.iic.kit.edu/` – for providing the system environment of the IBM System z and the IBM DS8700.

5. REFERENCES

[1] I. Ahmad, J. Anderson, A. Holler, R. Kambo, and V. Makhija. An analysis of disk performance in vmware esx server virtual machines. In *WWC-6*, 2003.
[2] S. Becker. *Coupled model transformations for QoS enabled component-based software design*. PhD thesis, Universität Oldenburg, 2008.
[3] S. Becker, H. Koziolek, and R. Reussner. The palladio component model for model-driven performance prediction. *J. of Systems and Software*, 82(1), 2009.
[4] D. Bruhn, Q. Noorshams, S. Kounev, and R. Reussner. Storage Performance Analyzer (SPA). `http://sdqweb.ipd.kit.edu/wiki/SPA`, 2012.
[5] J. Gantz and D. Reinsel (IDC). THE DIGITAL UNIVERSE IN 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East. `http://idcdocserv.com/1414`, 2012. Last accessed: Jan 2013.
[6] A. Gulati, C. Kumar, and I. Ahmad. Storage workload characterization and consolidation in virtualized environments. In *VPACT '09*.
[7] InformationAge. The year of virtual storage. `http://www.information-age.com/channels/the-cloud-and-virtualization/perspectives-and-trends/1596523/the-year-of-virtual-storage.thtml`, 2011. Last accessed: Jan 2013.
[8] L. Kapova and T. Goldschmidt. Automated feature model-based generation of refinement transformations. In *SEAA '09*.
[9] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu. An analysis of performance interference effects in virtual environments. In *ISPASS '07*.
[10] S. Kraft, G. Casale, D. Krishnamurthy, D. Greer, and P. Kilpatrick. Performance models of storage contention in cloud environments. *Springer Journal of Software and Systems Modeling*, 2012.
[11] S. Kundu, R. Rangaswami, A. Gulati, M. Zhao, and K. Dutta. Modeling Virtualized Applications using Machine Learning Techniques. In *VEE '12*.
[12] P. Mell and T. Grance. The nist definition of cloud computing. *National Institute of Standards and Technology*, 53(6), 2009.
[13] D. Menascé and V. Almeida. *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning*. Prentice Hall, 2000.
[14] D. Menascé, V. Almeida, L. Dowdy, and L. Dowdy. *Performance by Design: Computer Capacity Planning by Example*. Prentice Hall science explorer. Prentice Hall, 2004.
[15] Q. Noorshams, D. Bruhn, S. Kounev, and R. Reussner. Predictive Performance Modeling of Virtualized Storage Systems using Optimized Statistical Regression Techniques. In *ICPE '13*.
[16] Q. Noorshams, S. Kounev, and R. Reussner. Experimental Evaluation of the Performance-Influencing Factors of Virtualized Storage Systems. In *EPEW '12*, volume 7587 of *LNCS*. Springer.
[17] S. Oliveira, K. Furlinger, and D. Kranzlmuller. Trends in computation, communication and storage and the consequences for data-intensive science. In *IEEE HPCC-ICESS'12*, pages 572 –579.
[18] TechNavio. Global Server Virtualization Market 2012-2016. `http://www.technavio.com/content/global-server-virtualization-market-2012-2016`, 2013. Last accessed: Jan 2013.
[19] M. Woodside, D. Petriu, and K. Siddiqui. Performance-related completions for software specifications. In *ICSE '02*.