Modeling of I/O Performance Interference in Virtualized Environments with Queueing Petri Nets

Qais Noorshams, Kiana Rostami, Samuel Kounev, Ralf Reussner
Karlsruhe Institute of Technology, Germany
E-mail: [firstname.lastname]@kit.edu

***Abstract*—Virtualization technology allows to share the physical resources used in IT infrastructures for efficient and flexible system operation. Sharing of physical resources, however, comes usually at the cost of performance and poses significant challenges to respect the Quality-of-Service of consolidated data-intensive applications due to the mutual performance interference among the applications. The non-trivial impact of workload consolidation on the I/O performance can be anticipated using explicit performance analysis techniques. In current practice, however, explicit modeling of I/O performance interference effects in virtualized environments is usually avoided due to their complexity. In this paper, we present an explicit performance modeling approach of I/O performance interference in virtualized environments with queueing Petri nets (QPNs). More specifically, we first highlight major challenges when modeling I/O performance in virtualized environments. Then, we create a single-VM I/O performance model calibrated with response time measurements to capture the complex behavior of a representative, real-world environment based on IBM System z and IBM DS8700 server hardware. Finally, we use the I/O performance model to evaluate the I/O performance when the workload is distributed heterogeneously on co-located virtual machines. Overall, we effectively create an I/O performance interference model capturing the I/O performance effects in a multi-VM environment with less than 10% prediction error on average.**

***Keywords*-I/O; Storage; Virtualization; Performance; Interference; Prediction; Modeling; Queueing Petri Net**

I. Introduction

Reliable and efficient IT systems are of central importance in today's data centers. Virtualization technology is an enabling mechanism for reliable and efficient data center operation by allowing to decouple logical from physical resources with an additional abstraction layer and granting virtual machines (VMs) the consolidated access to pooled resources.

In virtualized environments, the running VMs are isolated regarding operating system or application errors, for instance. Isolation regarding performance, however, is widely an open challenge especially for data-intensive applications [1], [2], [3]. The co-located VMs cause complex I/O performance interference effects among their workloads and disturb the overall system performance significantly. With the drastic increase in I/O resource demands over the past years, applications deployed in virtualized environment are expected to get throttled more and more if this trend is not taken into account.

Performance modeling and evaluation techniques capturing the I/O performance interference effects can help to run the system environment efficiently while respecting Quality-of-Service requirements. In current practice, however, explicit modeling of I/O performance interference effects in virtualized environments is avoided due to the high complexity of today's virtualized environments. Only few explicit modeling approaches for I/O performance interference in virtualized environments have been proposed (e.g., [3]), however, with a focus on consolidation scenarios only or with validation limited to basic environments. The complex system environments and logical layers between the applications and the physical storage lead to complex performance interference effects posing significant challenges for explicit modeling approaches to create practical performance models.

To address these problems, in this paper, we present an explicit performance modeling approach of I/O performance interference in virtualized environments with queueing Petri nets (QPNs). More specifically, we first highlight major challenges when modeling I/O performance in virtualized environments. Then, we create a single-VM I/O performance model to capture the complex behavior of a representative, real-world environment based on IBM System z and IBM DS8700 server hardware. For the calibration of the model, we use response time measurements only that does not rely on hardware-specific or low-level monitoring data. Finally, we use the I/O performance model to evaluate the I/O performance when the workload is distributed heterogeneously on co-located virtual machines. Overall, we effectively create an I/O performance interference model capturing the I/O performance effects in a multi-VM environment with less than 8% and 11% mean prediction error for read and write requests, respectively.

To summarize, the contribution of this paper is two-fold: i) We present a modeling approach of I/O performance interference effects in virtualized environments calibrated with response time measurements from one virtual machine to estimate the interference effects of multiple virtual machines. ii) We evaluate our approach in a real-world environment based on the state-of-the-art server technology of the IBM System z and IBM DS8700. We extend our previous work [4], which is limited to homogeneous workload, and create a heterogeneous I/O performance model capturing the I/O performance interference effects in a multi-VM environment with different workload intensities in the VMs.

This paper is structured as follows: In Section II, we show an example to further motivate our approach. Then, we give a brief introduction to queueing Petri nets in Section III. Section IV highlights the challenges when modeling I/O performance interference. In Section V, we create and evaluate the per-

Postprint version. To be published in Proceedings of the IEEE 22nd International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2014), Paris, France, 2014.

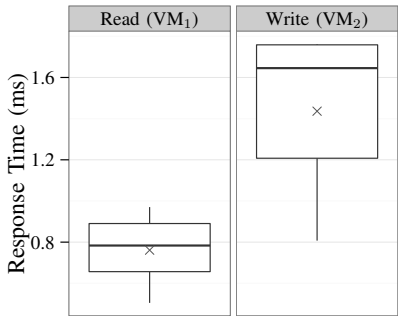

Fig. 1: Measurements of Performance Interference Effects

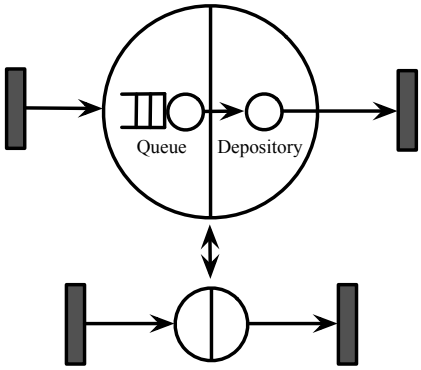

Fig. 2: A queueing place and its shorthand notation (from: [9]).

formance model capturing the I/O performance interference effects observed in the considered system environment. Finally, Section VI reviews related work followed by Section VII summarizing the paper.

II. Motivating Example

To illustrate typical I/O performance interference effects, consider the following setup whose measurements are shown in Figure 1. Two workloads with 10 clients each are running in separate virtual machines, VM_1 is running a read-intensive workload and VM_2 is running a write-intensive workload. In a first scenario, the workload of VM_2 increases up to 50 clients causing the performance of VM_1 to drop (left part of Figure 1). In a second scenario, the opposite occurs and the workload of VM_1 increases up to 50 clients while disturbing the performance of VM_2 (right part of Figure 1). In these simple scenarios, the performance of the read and write workload, which is kept constant in the respective VM, spreads by 92.1% and 118.3%, respectively, depending on the workload running on the co-located virtual machine. This simple observation is the motivation for our approach and demonstrates the need for performance models allowing to analyze and evaluate the performance interference of co-located data-intensive virtual machines.

III. Modeling with Queueing Petri Nets

Queueing Petri nets (QPNs) are a powerful modeling technique combining the queueing network and the Petri net formalism. In the following, we give a brief introduction to QPNs, which is taken from [5]. Detailed information and formal definitions of QPNs can be found in [6].

An ordinary *Petri net (PN)* is a bipartite directed graph comprised of places, drawn as circles, and transitions, drawn as bars. In PNs, token reside in the places and travel through the net using the firing transitions to describe a certain behavior. Multiple extensions to PNs have been developed in order to increase the modeling power, such as *Colored PNs (CPNs)* introduced by K. Jensen [7]. CPNs allow a type (color) to be attached to a token to classify them into groups. In addition to introducing token colors, CPNs also allow transitions to fire in different *modes* (transition colors).

Other extensions to PNs allow temporal (timing) aspects to be integrated into the net description. In particular, *Stochastic PNs (SPNs)* [8] attach an exponentially distributed *firing delay* to each transition, which specifies the time the transition waits after being enabled before it fires. *Generalized Stochastic PNs (GSPNs)* allow two types of transitions to be used: immediate and timed. Once enabled, immediate transitions fire in zero time. If several immediate transitions are enabled at the same time, the next transition to fire is chosen based on *firing weights* (probabilities) assigned to the transitions. Timed transitions fire after an exponentially distributed firing delay as in the case of SPNs. The firing of immediate transitions always has priority over that of timed transitions.

While CGSPNs have proven to be a very powerful modeling formalism, they do not provide any means for direct representation of queueing disciplines, which has led to *Queueing PNs (QPNs)*. QPNs allow queues to be integrated into places of CGSPNs. A place of a CGSPN that has an integrated queue is called a *queueing place* and consists of two components, the *queue* and a *depository* for tokens which have completed their service at the queue. This is depicted in Figure 2.

The behavior of the net is as follows: tokens, when fired into a queueing place by any of its input transitions, are inserted into the queue according to the queue's scheduling strategy. Tokens in the queue are not available for output transitions of the place. After completion of its service, a token is immediately moved to the depository, where it becomes available for output transitions of the place. This type of queueing place is called *timed* queueing place. In addition to timed queueing places, QPNs also introduce *immediate* queueing places, which allow pure scheduling aspects to be described. Tokens in immediate queueing places can be viewed as being served immediately. Scheduling in such places has priority over scheduling/service in timed queueing places and firing of timed transitions. The rest of the net behaves like a normal CGSPN.

IV. Challenges

In this section, we discuss important challenges when modeling I/O performance interference as well as how we approach them.

1) *Performance Evaluation of a Representative Setup*
An important aspect is evaluating a representative experimental setup, i.e., both system and workload setup, which makes performance modeling more difficult. In our approach, we will use a representative system environment. Considering the workload, it usually suffices to obtain best and worst case performance estimations with synthetic workloads, which is done in practice to evaluate storage systems. Thus, we will use such a benchmark in in our approach.
2) *Modeling the Observation vs. Modeling the Behaviour*
To build a performance model, we distinguish two ideas, i) modeling experiment observations and ii) modeling the internal behavior. While both require knowledge of the environment, we reason that it is more useful to develop a process how to execute experiments and model the observations. This allows to abstract from details and implementation specifics. While it can be easier to model the internal behavior of an environment, this requires deep internal knowledge, documentation, and even instrumentation for parameterization, which is not necessarily always available.
3) *Determining the Topology*
One part of performance modeling is to determine the model topology covering the software and hardware resources of the system environment. Our goal is to keep the model as simple as possible to be able to reasonably parameterize the model. More detailed models might increase the model accuracy with the potential sacrifice in applicability, such that the two goals, accuracy and applicability, might result in a trade-off.
4) *Estimating the Service Times*
After the definition of the model topology, the queueing models need to be parameterized by estimating the service times. The most common approach is to use utilization law, however, in contrast to the utilization of CPUs, the notion of utilization for storage systems in virtualized environments is not well-defined. Therefore, we use an approach to estimate the service times based on response time measurements.

V. Modeling I/O Performance Interference

In this section, we present the development of our heterogeneous QPN model of a representative system environment. For modeling and model solving, we use the *Queueing Petri net Modeling Environment (QPME)* [9].

A. System Under Study

For our I/O performance analysis, we consider a representative virtualized environment based on the IBM mainframe *System z* and the storage system *DS8700*, which we have analyzed in our previous work [10]. They are state-of-the-art high-performance virtualized systems with redundant and 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. [11]). The System z provides processors and memory, whereas the DS8700 provides storage space. The structure of this environment is illustrated in Figure 3.

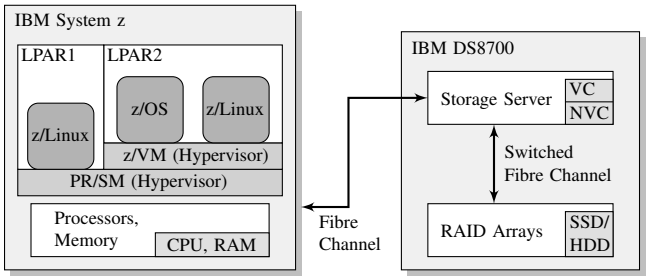

Fig. 3: IBM System z and IBM DS8700

The *Processor Resource/System Manager (PR/SM)* is a hypervisor managing logical partitions (*LPARs*) of the machine (therefore also called *LPAR hypervisor*) and enabling CPU and storage virtualization. For memory virtualization and administration purposes, IBM introduces another hypervisor, *z/VM*. The System z supports the classical mainframe operating system *z/OS* and special Linux ports for System z commonly denoted as *z/Linux*. The System z is connected to the DS8700 via fibre channel. Storage requests are handled by the storage server, which is equipped with a volatile cache (VC) and a non-volatile cache (NVC). The storage server is connected via switched fibre channel to SSD- or HDD-based RAID arrays.

The storage server applies several pre-fetching and destaging algorithms for optimal performance, cf. [12]. When possible, read requests are served from the volatile cache, otherwise, they are served from the RAID arrays and stored together with pre-fetched data in the volatile cache for future accesses. Write-requests are propagated both to the volatile and non-volatile cache, they are then destaged to the RAID arrays asynchronously from the volatile cache and discarded from the non-volatile cache.

In our system environment, the DS8700 contains 2 GB NVC and 50 GB VC with a RAID5 array containing seven HDDs. Calibration and validation measurements are obtained in z/Linux virtual machines (VM) with shared cores and 4 GB of memory each. The file system is configured to the de facto standard *EXT4* and as I/O scheduler, we use the default scheduler in virtualized environments *NOOP*, cf. [13]. Further, we focus our measurements on the storage performance using POSIX configuration. As a basis for our experimental analysis, we use the open source *Flexible File System Benchmark* (FFSB) [14] due to its fine-grained configuration possibilities. FFSB runs at the application layer and measures the end-to-end response time covering all system layers from the application all the way down to the physical storage. For a given configuration of a benchmark run, a set of files is created first. Then, the target number of workload threads (i.e., clients) are launched and they begin issuing read and write requests of the specified size, which is configured to 4 KB in this paper, directed to a

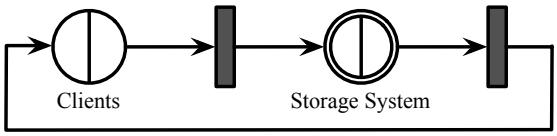

Fig. 4: Initial QPN Model

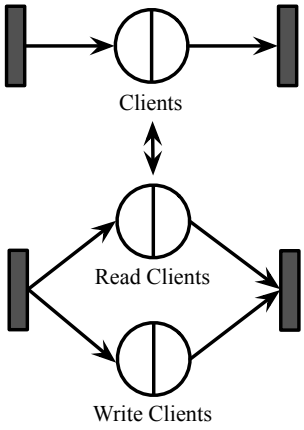

Fig. 5: Client Places for Heterogeneous Workload

randomly chosen target from the file set. Each thread issues a request as soon as the previous one is completed. For any configuration, we use a one minute warm-up time and a five minute measurement time. During this time, the benchmark typically gathers millions of measurement samples.

B. Initial Model

The starting point for our I/O performance interference model is a homogeneous, single-class queueing model for read and write requests as described in our previous work in an iterative, step-by-step process [4].

As network topology, we use a closed model as indicated in Figure 4. In general, closed models are the most popular for software systems since real applications are layered and the interactions between layers is subject to admission control or finite threading limits [15]. The *Clients queueing place* with infinite server queue and service time equal to the client think time represents the arriving requests at the system. The *Storage System subnet place* captures the I/O performance in a separate queueing Petri subnet. Initially, this subnet consists of one *queueing place*. The places are connected using *immediate transitions* that fire as soon as they are enabled.

For the calibration, we model the read request and write request service times using a *gamma-distribution* whose parameters (k, θ) are estimated from the measurements. To estimate the standard deviation σ, we measure the system under low workload-intensity with one thread. Then, we scale the load up to 100 threads in steps of 10 for both read and write requests, where we observe a strong correlation between the number of workload threads t and the mean read and write response time measurements ρ^m, respectively, in the form

$$\rho^m \approx c_1\, t + c_2,\ \ c_i \in \mathbb{R}. \tag{1}$$

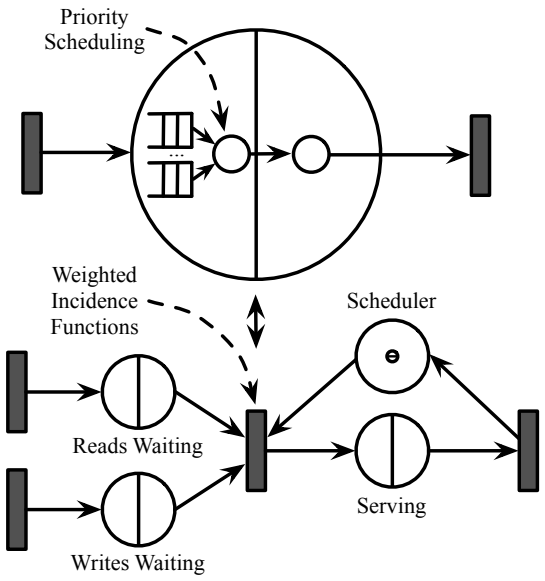

Fig. 6: Explicit QPN Model of Request Scheduling

Using mean value analysis (MVA) in our topology, we can easily estimate the mean read and write request service time μ, respectively, with a given value of t using

$$\rho^m(t) = \mu[1 + \bar{t}(t-1)], \tag{2}$$

where $\bar{t}(t-1)$ is the average number of threads found in the queue by an arriving thread. For our estimation, we use $t = 25$. Then, the parameters of the respective gamma-distribution of read and write requests, respectively, are simply

$$k := \left(\frac{\mu}{\sigma}\right)^2 \text{ and } \theta := \frac{\sigma^2}{\mu}. \tag{3}$$

C. Mixed Workload Model

In this section, we extend the queueing model to account for mixed read/write requests, where we consolidate read and write workloads with different workload intensities, i.e., number of threads. The workload that is used to create the model consists of different numbers of read and write threads t_r and t_w within one VM and both are varied between 10 and 50 in steps of 10.

In the model, the request types are distinguished using different token classes (or colors). For the two request types, we first introduce separate places for the tokens, cf. Figure 5. When mixing the requests, we observe that the mean read and write request response times are systematically over- and underestimated, respectively. We capture this observation in our QPN model by using a priority scheduling mechanisms and explicitly assigning relative priorities w_r and w_w to the read and write requests, respectively. As shown in Figure 6, we model this behavior explicitly by separating the queueing place of the storage system into *Waiting* queueing places and a *Storage* queueing place with a connecting transition whose incidence functions are weighted. A *Scheduler* place with one token ensures that only one request is served at a time and schedules the next request according to the relative priorities.

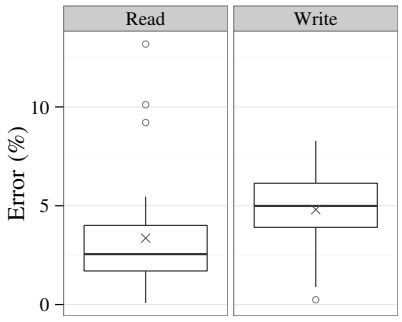

Fig. 7: Goodness-of-fit for Heterogeneous Workload Model

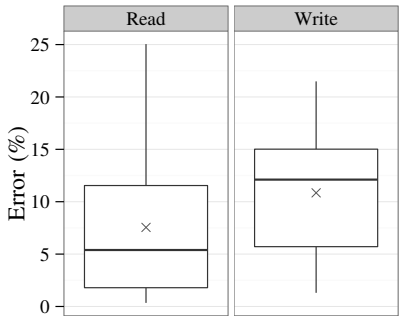

Fig. 8: Prediction Error for Multi-VM Workload

We extend the network topology of our initial QPN model and use the service times estimated as described in the previous section for the calibration. To determine the weights of the incidence functions, we search for appropriate values that minimize the difference between the measurements of the considered configurations and the simulation results. To predict the performance of unseen configurations, we use statistical modeling techniques to built a function f for the relative read and write request weights $\frac{w_r}{w_w}$ depending on the relative number of read and write workload threads $\frac{t_r}{t_w}$, i.e., $\frac{w_r}{w_w} = f(\frac{t_r}{t_w})$.

To evaluate how well the QPN model reflects the measurements, Figure 7 shows the calibration error, i.e., the difference between measurements and simulation results. Overall, we obtain a very good fit of the model with an average error of the mean response times of 3.36% for read requests and 4.80% for write requests.

D. Evaluating Multi-VM Workload

We now use our model of the previous section to estimate the impact of workloads from multiple VMs and predict their performance behavior and interference.

As prediction configuration, we distribute the workload on two virtual machines and each running a read-intensive and write intensive workload, respectively. We vary the workload threads in each VM from 10 to 50 in steps of 10 covering also the examples scenarios shown in the motivating example in Section II. For the model, we use the service times estimated in isolation and relative priority estimated using one VM measurements. We compare the mean response time measurements with the simulation results.

The prediction results are shown in Figure 8. Overall, the results are promising with a median error of 5.39% and 12.11% and a mean error of 7.55% and 10.85%, respectively. Especially the maximum prediction errors are encouraging with 25.05% und 21.49% for read and write requests, respectively.

Overall, there are some interesting conclusions. Albeit the workload is relatively simple, the model could be kept compact to capture the performance behavior and interference effects between the request types in a sufficiently complex environment. This makes it easier to parameterize the model and employ it in similar contexts.

Furthermore, while not too surprising, it was still interesting to see that the scheduling in the operating system – with the shifting of the default I/O scheduler – could be abstracted away. This reduces the complexity in the model and allows to draw reasonable conclusions from one VM workloads to multi-VM workloads. In other words, it can be reasonable to aggregate homogeneous VMs to mixed workload VMs for performance evaluation and avoiding to build and measure a large amount of VMs if it only slightly improves prediction accuracy.

The model can serve as a starting point for explicit modeling of I/O performance interference effects in virtualized environments. It is also now to us to extend the workload scenarios and further improve these results.

VI. Related Work

The work closely related to the approach presented in this paper can be classified into two groups. The first group analyzes I/O performance interference effects in virtualized environments. Here, Chiang et al. [2] use linear and second degree polynomials to model I/O performance interference. They use the models for scheduling algorithms to manage task assignments in virtualized environments. As input in their model, they use read and write request arrival rates as well as local and global CPU utilization. In [16], Yang et al. present a framework that uses a set of pre-defined workloads to identify characteristics of the hypervisor I/O scheduler. Furthermore, they show how this information can be exploited to deteriorate the I/O performance of co-located virtual machines. To analyze performance interference also across resources, Koh et al. [1] manually run CPU-bound and I/O-bound benchmarks. They develop mathematical models for prediction of normalized performance compared to the isolated performance of the benchmark. In an experimental study, Pu et al. [17] analyze CPU and network I/O performance interference in a Xen-based environment. They conclude that the least performance degradation occurs for workloads with different resource demands, i.e., CPU and network I/O demand or mixing small with large network demands. In summary, however, none of these approaches uses explicit, queueing theory-based modeling approaches to capture the I/O performance interference effects. Furthermore, some approaches use vendor-specific monitoring

tools (e.g., *xen-top*) to obtain an internal view of the global system environment, which is not necessarily always feasible.

The second group is focused on modeling storage performance in virtualized environments. Storage modeling approaches in native environments, e.g., [18], [19], [20], [21], [22], are intentionally left out of scope as such approaches strongly rely on low-level instrumentation and monitoring data, e.g., allocation of data to disk sectors, disk seek times, disk rotation time, or single disk utilization. In typical virtualized environments, such information is hardly available for storage users, if available at all, hampering the reliable parametrization of these models. Closest to our work, Kraft et al. [3] present two approaches based on queueing theory to predict the I/O performance of consolidated virtual machines. Both methods use monitored measurements on the block layer that is lower than typical applications run. Moreover, both methods are focused on performance prediction of consolidation scenarios only without considering the performance and interference effects due to changes in the workload intensity across multiple VMs. In [23], Ahmad et al. analyze the I/O performance of VMware's ESX Server virtualization. They compare virtual to native performance using benchmarks. They further create mathematical models for the virtualization overhead used for the prediction of I/O throughput degradation. By applying different machine learning techniques, Kundu et al. [24] use artificial neural networks and support vector machines for dynamic capacity planning in virtualized environments. Further, Gulati et al. [25] present a study on storage workload characterization in virtualized environments, but perform no performance analysis.

VII. Conclusion

We presented an explicit performance modeling approach of I/O performance interference in virtualized environments with queueing Petri nets (QPNs). We created an I/O performance model to capture the complex behavior of a representative, real-world environment based on IBM System z and IBM DS8700 server hardware. We calibrated the model with response time measurements obtained in a single VM setup and refrained from using hardware-specific or low-level monitoring tools. We use the I/O performance model to evaluate the I/O performance when the workload is running with different intensities on two co-located virtual machines. We effectively captured the I/O performance effects in the considered environment with less than 8% and 11% mean prediction error for read and write requests, respectively. The results are generally encouraging as the QPN model is kept simple to be able to calibrate it in an automated manner. It is now to us to extend the workload setup and further improve these results. We envision that our approach can be beneficial in a multiple application scenarios. Such I/O performance models can support system developers and data center operators in capacity planning and virtual machine consolidation decisions, for instance. Our approach does not depend on hardware-specific or low-level monitoring tools allowing it to be generally applied to any system environment.

Acknowledgments This work was partially supported by the German Research Foundation (DFG) under grant No. KO 3445/6-1, and the German Federal Ministry of Economics and Energy (BMWI), grant No. 01MD11005 (PeerEnergyCloud). 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.

References

[1] 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*.

[2] R. C. Chiang and H. H. Huang, "TRACON: Interference-aware Scheduling for Data-intensive Applications in Virtualized Environments," in *SC '11*.

[3] S. Kraft, G. Casale, D. Krishnamurthy, D. Greer, and P. Kilpatrick, "Performance Models of Storage Contention in Cloud Environments," *SoSyM*, 2012.

[4] Q. Noorshams, K. Rostami, S. Kounev, P. Tůma, and R. Reussner, "I/O Performance Modeling of Virtualized Storage Systems," in *MASCOTS '13*.

[5] S. Kounev, "Performance Modeling and Evaluation of Distributed Component-Based Systems using Queueing Petri Nets," *IEEE TSE*, vol. 32, no. 7, July 2006.

[6] F. Bause, "Queueing Petri Nets - A formalism for the combined qualitative and quantitative analysis of systems," in *Proc. of the 5th Intl. Workshop on Petri Nets and Performance Models*, 1993.

[7] K. Jensen, *Coloured Petri Nets and the Invariant Method*. Mathematical Foundations on Computer Science, LNCS 118:327-338, 1981.

[8] F. Bause and F. Kritzinger, *Stochastic Petri Nets - An Introduction to the Theory*, 2nd ed. Vieweg Verlag, 2002.

[9] S. Kounev, S. Spinner, and P. Meier, "Qpme 2.0 - a tool for stochastic modeling and analysis using queueing petri nets," in *From Active Data Management to Event-Based Systems and More*, ser. LNCS, K. Sachs, I. Petrov, and P. Guerrero, Eds. Springer, 2010, vol. 6462.

[10] Q. Noorshams, S. Kounev, and R. Reussner, "Experimental Evaluation of the Performance-Influencing Factors of Virtualized Storage Systems," in *EPEW '12*, ser. LNCS, vol. 7587. Springer, 2012.

[11] P. Mell and T. Grance, "The NIST definition of cloud computing," *National Institute of Standards and Technology*, vol. 53, no. 6, 2009.

[12] B. Dufrasne, W. Bauer, B. Careaga, J. Myyrrylainen, A. Rainero, and P. Usong, "IBM System Storage DS8700 Architecture and Implementation," http://www.redbooks.ibm.com/abstracts/sg248786.html, 2010.

[13] X. Ling, S. Ibrahim, H. Jin, S. Wu, and T. Songqiao, "Exploiting Spatial Locality to Improve Disk Efficiency in Virtualized Environments," in *MASCOTS '13*.

[14] "Flexible File System Benchmark (FFSB)," http://github.com/FFSB-prime (version with fixes and extensions of http://ffsb.sf.net), [Online; last accessed: May 2014].

[15] W. Wang and G. Casale, "Bayesian Service Demand Estimation Using Gibbs Sampling," in *MASCOTS '13*.

[16] Z. Yang, H. Fang, Y. Wu, C. Li, B. Zhao, and H. Huang, "Understanding the Effects of Hypervisor I/O Scheduling for Virtual Machine Performance Interference," in *CloudCom '12*.

[17] X. Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, and C. Pu, "Understanding Performance Interference of I/O Workload in Virtualized Cloud Environments," in *CLOUD '10*.

[18] J. S. Bucy, J. Schindler, S. W. Schlosser, G. R. Ganger, and Contributors, *The DiskSim Simulation Environment - Version 4.0 Reference Manual*, Carnegie Mellon University, Pittsburgh, PA, 2008.

[19] P. Harrison and S. Zertal, "Queueing models of RAID systems with maxima of waiting times," *Performance Evaluation*, vol. 64, 2007.

[20] A. S. Lebrecht, N. J. Dingle, and W. J. Knottenbelt, "Analytical and Simulation Modelling of Zoned RAID Systems," *The Computer Journal*, vol. 54, 2011.

[21] E. K. Lee and R. H. Katz, "An analytic performance model of disk arrays," *SIGMETRICS Perform. Eval. Rev.*, vol. 21, no. 1, 1993.

[22] E. Varki and S. X. Wang, "A performance model of disk array storage systems," in *Int. CMG Conference*, 2000.

[23] 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.

[24] S. Kundu, R. Rangaswami, A. Gulati, M. Zhao, and K. Dutta, "Modeling Virtualized Applications using Machine Learning Techniques," in *VEE '12*.

[25] A. Gulati, C. Kumar, and I. Ahmad, "Storage workload characterization and consolidation in virtualized environments," in *VPACT '09*.