TOWARDS PERFORMANCE PREDICTION FOR CLOUD COMPUTING ENVIRONMENTS BASED ON GOAL-ORIENTED MEASUREMENTS

Michael Hauck
FZI Research Center for Information Technology, Karlsruhe, Germany
hauck@fzi.de

Jens Happe
SAP Research, Karlsruhe, Germany
jens.happe@sap.com

Ralf H. Reussner
Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
reussner@kit.edu

Keywords: Performance Prediction, Measurements, Cloud Computing, Virtualization, Modelling

Abstract: Scalability and performance are critical quality attributes of applications developed for the cloud. Many of these applications have to support hundreds or thousands of concurrent users with strongly fluctuating workloads. Existing approaches for software performance evaluation do not address the new challenges that arise for applications executed in cloud computing environments. The effects of virtualization on response times, throughput, and resource utilisation as well as the massive number of resources available require new platform and resource models for software performance evaluation. Modelling cloud environments using established approaches for software performance prediction is a cumbersome task that requires a detailed understanding of virtualization techniques and their effect on software performance. Additional complexity comes from the fact that cloud environments may combine multiple virtualization platforms which differ in implementation and performance properties.

In this position paper, we propose an approach to infer performance models of cloud computing environments automatically through goal-oriented measurements. The resulting performance models can be directly combined with established model-driven performance prediction approaches. We outline the research challenges that have to be addressed in order to employ the approach for design-time performance predictions of software systems running in cloud computing environments.

1 INTRODUCTION

Many applications developed for the cloud have high requirements with respect to performance and scalability in order to support hundreds or thousands of users. Response time and throughput of such applications are critical and thus have to be considered in early stages of the software life-cycle. Software architects must take into account the specific properties of cloud environments. Most fundamental is a detailed understanding of different virtualization platforms and their effect on software performance. Furthermore, virtualization platforms provide means to scale up (increase the size of a virtual machine by assigning it more memory or cpus) and scale out (adding additional virtual machines to an application). Both must be supported by the application. Performance standards in practice are high. An application is considered scalable only if its throughput increases linearly with the resources assigned to it.

Software performance engineering (SPE) and model-driven performance engineering (MDPE) enable the prediction of software performance at an early stage of the software development process. Software architects provide architectural models with performance annotations of their application, which are then transformed into performance analysis models, such as queueing networks, stochastic Petri nets, or simulation models (see overview in (Balsamo et al., 2004; Koziolek, 2010)). The results of the performance analysis enable software architects to reason on the application's performance, for example to evaluate different design alternatives, to plan resource capacities or to estimate the performance of a software system in a cloud.

For accurate performance predictions, the performance-relevant properties of cloud environments have to be reflected by the performance models. However, current approaches provide only

limited support for performance properties of cloud computing environments and, more specifically, virtualization platforms. Therefore, the integration of such properties is a cumbersome, manual task, which requires detailed knowledge about virtualization techniques and their effect on software performance. In many cases, performance-relevant properties of virtualization platforms have to be retrieved by inspecting system documentation. Such documentation may not be available in all cases. Furthermore, the effect of specific implementation details on performance is often unknown. However, neglecting the performance impact of virtualization platforms can lead to inaccurate prediction results. For example, scheduling implementations can influence the response time of an application by several orders of magnitude (Schroeder et al., 2006). If no information about the virtualization platform is available, measurements have to be conducted in order to include its influence into the performance model. Such measurements involve a high manual effort to design and set up the measurements and analyze the results.

Moreover, cloud computing environments can make use of multiple virtualization techniques, which differ in their performance-relevant properties (Adams and Agesen, 2006), (Matthews et al., 2007). Thus, a performance model of a specific virtualization platform cannot necessarily be reused in a different setting.

To avoid the efforts of creating the performance model manually, we propose an approach to detect performance-relevant properties of cloud computing environments and virtualization environments automatically. This approach makes use of goal-oriented measurements which are conducted on the target platform to detect certain characteristics of performance-relevant properties of the virtualization environment. The environment is regarded as a "black-box", which means that the performance-relevant properties of the environment are not known a priori. Instead, the properties are derived by measurements that are executed on the platform, and fed into a performance model. The automation of both the measurements and the derivation of a performance model aims at easing the burden of performance predictions of software running in the cloud.

The remainder of this position paper is structured as follows. Section 2 discusses research questions regarding performance prediction of software running in cloud computing or virtualized environments based on measurements. The approach is presented in detail in Section 3. Section 4 presents related work, and Section 5 concludes the paper.

2 RESEARCH QUESTIONS

In this section, several open research questions are presented that need to be addressed in order to apply performance predictions for cloud computing environments based on goal-oriented measurements. The first three questions deal with design-based performance prediction for software running in virtualized environments, the latter two questions deal with goal-oriented measurements to derive performance-relevant properties.

What are the relevant performance properties of virtualization platforms?

Running software in virtualized environments leads to a complex technology stack which influences the software performance. To enable reasonable performance analyses, the key performance-relevant properties of virtualization environments have to be identified.

Such properties include for example virtualization overheads and virtualization implementation properties. Virtualization overhead has to be taken into account, as resource demands of software may lead to a different performance if the software is running on a virtual machine compared to its execution in a non-virtualized environment. Virtualization implementation properties play for example a crucial role in handling contention effects of multiple virtual machines running on the same physical server. This is done by a virtualization software called hypervisor, which provides sophisticated scheduling algorithms that affect the performance. Even for the same virtualization technique, different scheduler implementations may lead to different performance effects (Cherkasova et al., 2007). Hypervisors also provide a more complex logic to handle I/O requests compared to traditional OS schedulers, as virtual I/O devices have to be mapped to physical I/O devices.

Further performance-relevant properties that have been identified for software running in a cloud computing environment include network I/O properties. For example, some parts of a software may run locally with a local network connection, whereas other parts may run in the cloud where internet network connection properties have to be regarded.

In addition to virtualized platforms, cloud computing environments hide the server structure on which the application is running on. Even more, after deploying an application in the cloud, it may be moved dynamically to a different server with different performance properties after deployment. In our presented approach we want to focus on scenarios where the application is running on the same infrastructure on which the black-box measurements have

been taken. However, we acknowledge the fact that cloud computing environments may imply more complex influencing factors for performance modelling, which are subject to later enhancements of the approach.

How to reflect relevant performance properties of cloud computing environments / virtualization platforms in software performance models?

For design-time performance prediction, models of the software have to be provided, and to some extent models of the virtualized environment. Here, the granularity of the model mainly depends on the approach to be taken:

In a forward-engineering approach, the implementation of the application often does not exist yet, and the models tend to be rather abstract. Here, the challenge is to find a model granularity that does not require knowledge which not available during the design phase, but still allows to yield reasonable performance prediction results. Another option would be to use abstract models and automatically add additional information by using performance completions or coupled transformations, as proposed in (Happe et al., 2010).

In a reverse-engineering approach, the code of the software can be taken into account when creating the models. This allows to create more fine-grained models, which can be partially created automatically by tool support. Here, also more fine-grained performance-relevant information can be put into the model. Such an approach might be relevant for a scenario in which the performance of legacy systems should be analyzed w. r. t. a deployment in a virtualized environment.

Independent of the software model granularity, existing models for performance predictions have to be enhanced in order to reflect virtualization and cloud computing infrastructure properties, such as virtualization hypervisor properties, different network connections, virtual and physical servers, or server clusters.

How to integrate performance-relevant properties of cloud computing environments into performance analyses?

Additionally to the enhancements of software performance models, performance analysis tools have to be enhanced as well in order to support performance-relevant properties of cloud computing environments. For performance prediction, we aim at using a configurable analysis tool that can be configured depending on the detected properties (e.g. performance overheads or hypervisor properties). As the complete approach should run in an automated way where possible, analysis tool configuration should also happen in an automated way.

To implement performance prediction for our approach, we plan to enhance the Palladio Component Model (PCM) (Becker et al., 2009). The PCM supports design-based performance prediction based on software models and provides different performance analysis tools, such as analytical solvers and a discrete-event simulation.

How to design experiments to detect performance-relevant properties in a technology-independent way?

In order to apply performance prediction to different cloud environments, the measurements of the experiments have to be designed in a generic way and must not be tailored towards a specific virtualization technique or hypervisor scheduler implementation. It is also an open question which measurement metrics can be taken into account. While every system provides means to measure time spans, some virtualization solutions or cloud computing providers might not provide certain measurement metrics such as resource utilizations.

In (Hauck et al., 2010), we proposed generic experiments to detect load-balancing properties of general-purpose operating system (GPOS) schedulers. The experiments are based on response time measurements only and yielded promising results.

Furthermore, measurements aiming at detecting the virtualization overhead of certain resource demands (CPU, memory, network I/O) have already been conducted in prior work. We plan to automate the evaluation of the measurements in order to automatically derive performance models from the measurement results which can be used for performance prediction.

Depending on the scenario, different additional kinds of experiments might be possible. For example, one could think of taking measurements on multiple machines in the cloud to gain experiment results that allow to reason about the contention effects that occur.

How to conduct accurate measurements in virtualized environments?

In virtualized environments, additional challenges arise when taking measurements. First, the timer accuracy might suffer when accessing timer information from inside a virtual machine. Virtualization implementations provided enhanced timer accuracy in the last years, but initial experiment results showed that timer accuracy might suffer when measuring response

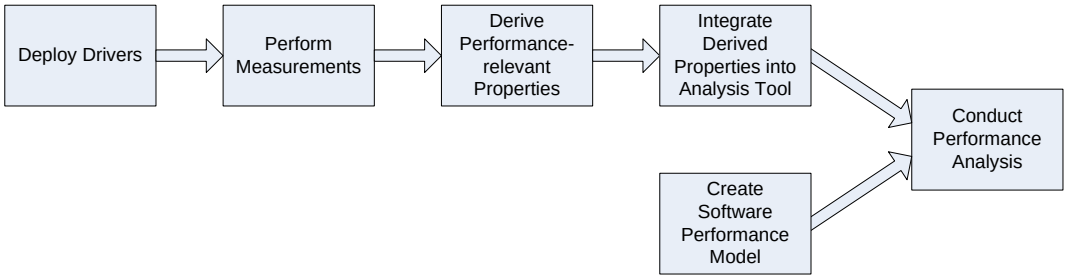

Figure 1: Workflow of the automatic derivation of performance-relevant properties.

time in virtual machines especially when the system is under load.

Another issue when taking measurements is that additional load might disturb the measurement results. In virtualized environments, additional load on the system might always and unpredictably occur (noisy neighbours). This can lead to disturbed measurement results, because performance isolation across virtual machines might not be the case (Gupta et al., 2006). To derive performance-relevant properties correctly, the measurements have to designed robust w. r. t. measurement noise.

3 APPROACH

In the following, we present our approach to derive performance-relevant properties automatically in more detail.

An overview on the approach is given in Figure 1 and consists of the following steps:

1. *Deploy Drivers.* The approach is based on goal-oriented measurements, which requires the target environment to be available. On this environment, i. e. on the virtual machines in which the target application is to be run on, a load driver has to be installed. For the load driver, we plan to use resource demands libraries which allow to generate different patterns of CPU load and I/O load.
2. *Perform Measurements.* Once the driver is deployed and calibrated, experiments are to be performed, which include issuing different patterns of CPU and I/O load and measuring certain parts of the issued load. To derive performance-relevant properties, the load patterns have to be designed in a way that the measured results allow to infer implementation properties of the platform through statistical analyses. The measurements should be robust w. r. t. measurement noise, and should be generic, i. e. they should be applicable to miscellaneous virtualization platforms, and not be targeted towards a specific platform. Possible generic measurements include virtualization overhead for certain resource demands, network I/O throughput and latency rates, as well as storage access properties (for different read-write patterns). The measurements are initiated by an experiment controller, which also collects the measurement results from the load drivers after the measurements have been completed.
3. *Derive Performance-relevant Properties.* The measurement results serve as input for the analysis. The analyzer derives properties by using statistical methods, and triggers further experiments if necessary.
4. *Integrate Derived Properties into Analysis Tool.* Finally, the detected properties have to be integrated into the analysis model. Configurable performance properties can be implemented in a performance analysis tool for example by using a configuration model. In this case, the detected properties would be passed to the analysis tool as a configuration instance. Depending on the property configuration, the analysis tool has to provide different analyzer logic in order to reflect the performance properties.

Once the performance analysis tool is configured based on the detected performance properties, the software architect can conduct a performance analysis by creating a model of the application software, which is then transformed into an analysis model. The analysis tool uses this model to perform the analysis, taking into account the experimentally derived performance properties.

Our approach aims at deriving performance-relevant properties for a wide range of virtualized environments. Besides virtualization environments consisting of a single hypervisor machine running a couple of virtual machines, the approach is also intended to be applicable on larger environments, for example cloud computing environments like Amazon EC2 (Amazon.com, 2006). Note that the generic approach could also be adapted for different properties of non-virtualized software systems, such as performance-relevant properties of messaging systems.

To relieve the software architect of the burden of deriving performance-relevant properties manually, we plan to automate large parts of the approach. Both taking the measurements and evaluating experiment results can be done in an automated way if measurements can be designed in a generic way in order to yield robust results. If certain properties cannot be unambiguously identified through measurements, additional reasoning has to be done manually in order to fine-tune the performance model. To support automation, we are currently implementing a measurements framework that is able to conduct and control the measurement experiments, as well as analysing the experiment results and deriving the performance-relevant properties.

4 RELATED WORK

Design-time performance prediction has been in the focus of research for the last years (surveyed in (Balsamo et al., 2004) and (Koziolek, 2010)), but virtualization and cloud computing as a novel trends propose new challenges for software performance prediction. An approach to model the performance impact of server consolidation with virtual machines is presented in (Menasce and Bennani, 2006), (Menasce, 2005). Analytical queueing models are used to provide performance estimates (i. e. response time, throughput, utilization) of virtual machines running on a single hypervisor. However, the system is modelled at a very abstract level and only rough performance estimates (average times only) are possible. Other approaches provide virtualization models, but focus only on the deployment of virtual machines or virtual clusters (Sotomayor et al., 2006), (Yamasaki et al., 2007). These approaches do not consider the performance of a virtualized application.

Various approaches make use of measurements to detect performance properties. Cherkasova and Gardner (Cherkasova and Gardner, 2005) propose measurements to detect a certain kind of CPU overhead for I/O that occurs for a specific virtualization solution (Xen). They use a framework similar to the one that is needed for our approach w. r. t. measuring and monitoring, however the approach is only applicable to the Xen hypervisor. Besides, the measurements only focus on a certain scheduling property of the hypervisor. Microbenchmarks are used in (Wood et al., 2008) to estimate performance overheads that occur when deploying an application into a virtualized environment. Different kinds of virtualization overhead are regarded to detect resource requirements, but the approach cannot be applied to analyze response times of applications running in virtualized environments. In (Woodside et al., 2001), resource functions are defined to derive resource demands of a software system by using regression splines. Zheng et. al. (Zheng et al., 2008) use Kalman filters to estimate parameters of the performance model. Both approaches aim at detecting input parameters for the software model, whereas our approach focusses on deriving performance-relevant properties of the virtualization environment. (Iosup et al., ress) provide initial measurements of performance properties of cloud computing platforms. The authors consider cloud computing services for scientific computing, an integration of measurement results into analysis tools, such as performance prediction tools, is however not in the focus.

Challenges of measurements of MPI performance characteristics are presented in (Gropp and Lusk, 1999). This work covers interesting issues that might lead to incorrect measurements, but does not present solutions for all issues. Besides, this work does not deal with measurement challenges specific to virtualization, such as inaccurate timers. Frameworks for automated benchmarks in distributed environments or grid environments have been proposed in (Kalibera et al., 2006), (Tsouloupas and Dikaiakos, 2006), but these frameworks do not cover performance properties specific to virtualization environments or cloud computing environments. Also, an automated evaluation of the benchmark results is not supported.

5 CONCLUSIONS

In this position paper, we presented a novel approach to enable performance prediction of applications running in cloud computing and virtualized environments. The approach applies goal-oriented measurements to automatically derive performance-relevant properties of the environment. A "white-box" perspective on the virtualized system is not necessary, i. e. detailed knowledge about the behaviour of virtualization and cloud technologies is not required. Instead, such properties are to be detected by goal-oriented measurements on the target platform.

We are currently implementing a framework for automated measurements in order to infer prediction models of virtualised environments by systematic measurements.

For this purpose, it has to be validated if the approach can to derive performance properties with reasonable accuracy. Based on our previous experience (Hauck et al., 2010), the approach is a promising starting point for an inclusion of performance properties of cloud computing environments or more detailed virtualization effects into design-time software performance prediction.

ACKNOWLEDGEMENTS

The work described in this paper was partially supported by the German Federal Ministry of Education and Research (BMBF) under grant 01IC10S01A.

REFERENCES

Adams, K. and Agesen, O. (2006). A Comparison of Software and Hardware Techniques for x86 Virtualization. In *ASPLOS-XII: Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM.

Amazon.com, I. (2006). Amazon EC2. Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2.

Balsamo, S., Di Marco, A., Inverardi, P., and Simeoni, M. (2004). Model-based Performance Prediction in Software Development: A Survey. *IEEE Transactions on Software Engineering*, 30(5):295–310.

Becker, S., Koziolek, H., and Reussner, R. (2009). The Palladio Component Model for Model-driven Performance Prediction. *Journal of Systems and Software*, 82:3–22.

Cherkasova, L. and Gardner, R. (2005). Measuring CPU Overhead for I/O Processing in the Xen Virtual Machine Monitor. In *USENIX 2005: Proceedings of the USENIX Annual Technical Conference*.

Cherkasova, L., Gupta, D., and Vahdat, A. (2007). Comparison of the Three CPU Schedulers in Xen. *SIGMETRICS Performance Evaluation Review*, 35(2):42–51.

Gropp, W. and Lusk, E. (1999). Reproducible Measurements of MPI Performance Characteristics. In *PVM/MPI 1999: Proceedings of the 6th European PVM/MPI Users' Group Meeting*. Springer-Verlag.

Gupta, D., Cherkasova, L., Gardner, R., and Vahdat, A. (2006). Enforcing Performance Isolation Across Virtual Machines in Xen. In *Middleware 2006: Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware*, New York, NY, USA. Springer-Verlag.

Happe, J., Becker, S., Rathfelder, C., Friedrich, H., and Reussner, R. H. (2010). Parametric Performance Completions for Model-driven Performance Prediction. *Performance Evaluation*, 67(8):694–716.

Hauck, M., Happe, J., and Reussner, R. H. (2010). Automatic Derivation of Performance Prediction Models for Load-balancing Properties Based on Goal-oriented Measurements. In *MASCOTS 2010: Proceedings of the 18th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE Computer Society.

Iosup, A., Ostermann, S., Yigitbasi, N., Prodan, R., Fahringer, T., and Epema, D. (in press). Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Transactions on Parallel and Distributed Systems*.

Kalibera, T., Lehotsky, J., Majda, D., Repcek, B., Tomcanyi, M., Tomecek, A., Tuma, P., and Urban, J. (2006). Automated Benchmarking and Analysis Tool. In *VALUETOOLS 2006: Proceedings of the 1st International Conference on Performance Evaluation Methodolgies and Tools*. ACM.

Koziolek, H. (2010). Performance Evaluation of Component-based Software Systems: A Survey. *Performance Evaluation*, 67(8):634–658.

Matthews, J. N., Hu, W., Hapuarachchi, M., Deshane, T., Dimatos, D., Hamilton, G., McCabe, M., and Owens, J. (2007). Quantifying the Performance Isolation Properties of Virtualization Systems. In *ExpCS 2007: Proceedings of the 2007 Workshop on Experimental Computer Science*. ACM.

Menasce, D. (2005). Virtualization: Concepts, Applications, and Performance Modeling. In *CMG 2005: Proceedings of the International CMG Conference*.

Menasce, D. and Bennani, M. (2006). Autonomic Virtualized Environments. In *ICAS 2006: Proceedings of the 2nd International Conference on Autonomic and Autonomous Systems*.

Schroeder, B., Wierman, A., and Harchol-Balter, M. (2006). Open Versus Closed: A Cautionary Tale. In *NSDI 2006: Proceedings of the 3rd Conference on Networked Systems Design & Implementation*. USENIX Association.

Sotomayor, B., Keahey, K., and Foster, I. (2006). Overhead Matters: A Model for Virtual Resource Management. In *VTDC 2006: Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*.

Tsouloupas, G. and Dikaiakos, M. D. (2006). Characterization of Computational Grid Resources Using Low-Level Benchmarks. In *E-SCIENCE 2006: Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing*. IEEE Computer Society.

Wood, T., Cherkasova, L., Ozonat, K., and Shenoy, P. (2008). Profiling and Modeling Resource Usage of Virtualized Applications. In *Middleware 2008: Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*. Springer-Verlag.

Woodside, C. M., Vetland, V., Courtois, M., and Bayarov, S. (2001). Resource Function Capture for Performance Aspects of Software Components and Sub-Systems. In *Performance Engineering, State of the Art and Current Trends*, pages 239–256. Springer-Verlag.

Yamasaki, S., Maruyama, N., and Matsuoka, S. (2007). Model-based Resource Selection for Efficient Virtual Cluster Deployment. In *VTDC 2007: Proceedings of the 3rd International Workshop on Virtualization Technology in Distributed Computing*.

Zheng, T., Woodside, C. M., and Litoiu, M. (2008). Performance Model Estimation and Tracking Using Optimal Filters. *IEEE Transactions of Software Engineering*, 34(3):391–406.