Modeling IaaS Usage Patterns for the Analysis of Cloud Optimization Policies

Sebastian D. Krach, Christian Stier
{krach, stier}@fzi.de
FZI Research Center for Information Technology

Athanasios Tsitsipas
athanasios.tsitsipas@uni-ulm.de
OMI, Ulm University

Abstract

Infrastructure as a Service (IaaS) operators need to balance multiple adversarial goals, such as data center performance and energy efficiency. Automated resource management policies implemented in IaaS Cloud middleware allow the operators to automate trade-off decisions. Simulation-based analyses are viable means to validate that the utilized policies achieve the goals of the operator. For an IaaS operator to perform meaningful analyses, changes in the workload mix of active VMs need to be considered. Current Cloud simulation approaches neglect the influence of VM submissions and terminations, or require the workload to be specified with little abstraction. In this paper, we present a unified approach for modeling IaaS workloads. Our workload model describes the IaaS workload as a sequence of time-triggered, event-driven external influences. We implement our model as an extension to Palladio and SimuLizar. Finally, we illustrate how historical real-world measurements are leveraged to evaluate resource management policies.

1 Introduction

IaaS operators provide computing infrastructures for their customers with limited knowledge of the application workload. They pursue adversary goals which are subject to their business model, e.g. minimizing the power consumption to cut operation costs or provide spare resources for unforeseen load peaks. Overprovisioning reduces operation costs, but carries the risk of not being able to serve spiking workloads of individual VMs. Automated resource management policies implemented in IaaS Cloud middleware aim to react on such bursty workloads, e.g. by optimizing VM allocations or scaling resources. Evaluating policies in the context of a real data center requires extensive experiments. This is cumbersome and expensive, as small testbeds are unsuitable for QoS related research, while public infrastructure providers do not offer the required degree of control [6].

Simulation offers an alternative, since it requires less resources to execute and facilitates running evaluations with a significantly larger number of infrastructure components. However, in order for the simulations to provide meaningful results, appropriate workload models are required. Infrastructure load resulting from requests of IaaS users to spawn multiple VMs may have significant impact on the overall data center load. Consequently, capturing such user actions in a user behaviour model is important to assess the performance of the optimization policies.

Existing simulation approaches, like CloudSim [2], provide means to predict the impact of user actions on the data center performance, but they require programmatical implementations of the workload models and the experiment scenarios. Describing simulation-ready models requires both domain knowledge, as well as a lot of insight into the simulation framework.

Vögele et al. [10] extend the modeling capabilities of the Palladio Component Model (PCM) [1] for describing user behavior on an architectural level. While their concept includes reusable behavior specification and refined probabilistic decision models based on guard semantics, it does not allow for simulation-time references. Time-controlled behavior is essential to evaluate policies using scenarios derived from recorded data. Heinrich et al. [9] extend the workload models of the PCM with representations of users' manual actions. Their approach regards personnel as schedulable resources and not as a driver for time-controlled workload changes. Groenda and Stier [8] present a PCM-based approach to describe VM workload patterns over time. While the authors' approach is able to model VM resource demands, it provides no support for user actions, as VM start requests or scheduled server shut-downs due to maintenance.

In this paper we present our extension to the workload models of Groenda and Stier [8] with support for time-controlled event-based experiment scenarios. We evaluate the approach in a scientific computing use case by comparing predicted utilization against the actually measured one. Finally, we quantitatively evaluate the prediction accuracy using a CPU-utilization based energy consumption prediction approach.

2 Timeline-based IaaS Workload Description

Workload in a data center is impacted by user actions, in particular, requests to start or stop VMs. In order to enable simulation-based reasoning on the

data center performance we regard these actions as event-based influence, captured in a separate scenario model.

The core element of our extension is called *Experiment Scenario Timeline*. The timeline allows user actions to be modeled by either specifying an absolute point in simulation time (*AbsoluteTimeEvent*) or relative to another event (*RelativeTimeEvent*). The nature of the event is encapsulated as *Request* model element contained in the element of the event, e.g *StartApplicationRequest*. Thus, event-specific information, e.g., point in time of execution, is separated from request-specific details, e.g., the identification of the VM Image to start, to provide future extensibility.

The *StartApplicationRequest* encapsulates a user's request to start a new VM instance. Depending on the type of application, it can consist of one or multiple VM instances. Following the execution of the request during simulation, the workload descriptions associated with the respective VMs are interpreted. Among others, we additionally model VM migrations or changes to the power state of a server. Further details can be found in [11].

The data center models are not simulated directly, but transformed into a Palladio Component Model (PCM) [1] instance which is then simulated using SimuLizar [5]. VM workloads are represented as *Assemblies*, *Usage Scenarios* and *Usage Evolutions* [8].

A dedicated event scheduler processes the *AbsoluteTimeEvents* ordered by ascending simulation time. For the closest event, a specific handler is registered with the underlying discrete event simulation framework which, when invoked, marks the event as *scheduled*. The actual request processing is realized using SimuLizar's reconfiguration support. The event handler triggers the execution of the reconfiguration transformations selected by the scheduled *Request*.

User actions, particularly VM bootups may take time to execute. To support those occurring relative to the completion time of a previous *Request*, the actual processing logic is encapsulated into separate *Adaptation Behaviors* [12] which are executed asynchronous to the continuation of the simulation. *RelativeTimeEvents* are scheduled once the event to which they refer is marked as successfully processed.

For each VM, a separate workload driver specified in the VM's Usage Scenario issues demand on the system, as described in [8] for Black Box applications. We have extended SimuLizar to support the addition and the removal of VMs, by adding support for a varying number of active Usage Scenarios. VM starts are realized by dynamically adding the respective PCM elements, particularly the VM's Usage Scenario, to the in-memory PCM instance of the active simulation. Subsequently, workload users for newly added scenarios are being created automatically. Similarly, VM stop requests are realized by removing the representing PCM elements, particularly the Usage Scenario, which in turn prevents new users from being issued. Users which have already started a scenario execution are being removed after completing their run.

3 Evaluation

In this section we conduct an evaluation of the presented concepts for the domain of High Performance Computing (HPC) applications. In the testbed we run CactoScale [7] to monitor the application behavior and the infrastructure performance data, which uses HBase, a scalable, column-oriented database, to store the historic monitoring data. In order to facilitate the experiment scenario modeling, we devised an approach to extract models based on the historical monitoring data of real-world IaaS Cloud infrastructure recorded by CactoScale. Section 3.1 discusses the collected information in an IaaS infrastructure. In Section 3.2, we summarize the nature of the employed application example. Finally, in Section 3.3 we evaluate the prediction accuracy in comparison to real-world measurements.

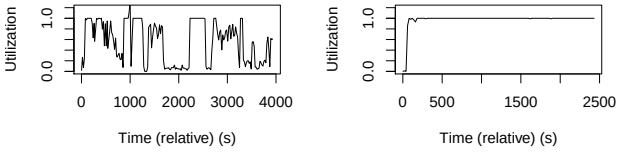
3.1 Monitoring in an IaaS Cloud Infrastructure

IaaS infrastructure monitoring tools have to support varying numbers of VMs while maintaining high data collection throughput and respond in real-time [4].

An HPC application requests CPU time or places system calls for filesystem interaction and allocates memory. In order to construct black box behaviour models of HPC workloads, the hypervisor hosting the VM, needs to be equipped with monitoring agents. Application specific details are only visible from inside the VM. Monitoring data provided by the application itself, its state or additional tags (e.g. application type), can be used to increase the simulation accuracy.

3.2 Evaluation Scenario

Our evaluation use case is based on the Molpro application [3], used in physical chemistry to calculate wave functions for small molecules. Molpro adheres to a run-to-completion semantics, as is common for HPC applications. Each Molpro execution is confined to a single VM. In the evaluation experiment we employ two Molpro configurations LCCSD and DFT, for each of which the CPU load profile is depicted in Figure 1. The load intensity is measured as single core utilization relative to the respective VM start. We consecutively started 26 Molpro instances on a cluster consisting of 8 servers.



(a) Molpro LCCSD VM (b) Molpro DFT VM

Figure 1: CPU Resource Demand of Separate VMs

3.3 Evaluation Results

We have extracted the model specifications for the resource environment and the actual application utilization based on historical measurements of the conducted experiment. Four VMs (one LCCSD at 197s, three DFT at 506s, 776s and 976s of experiment time) get allocated to the same server. In Figure 2 we graphically compare the CPU utilization over time. It shows the measured (blue) and predicted (red) CPU utilization over the course of the experiment. Both traces reflect the points in time at which the VMs are started (additionally visualized by dashed markers). Shortly after each start request the overall utilization increases by about 3%, which is the equivalent of one core of a 32 core CPU being fully utilized. Similarly, the phase where the LCCSD VM requires less CPU (second 1705 until 2208 relative to VM start, see Figure 1a) is visible starting at 1900s of experiment time in Figure 2.

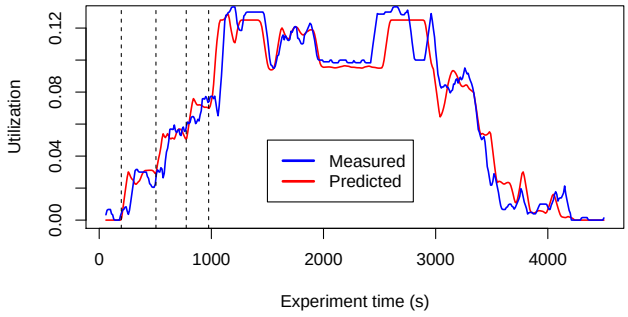


Figure 2: CPU Resource Demand of Separate VMs

In order to quantify the prediction accuracy over the duration of the experiment, we leverage a CPU utilization-based power consumption prediction method and compare the accumulated energy consumption of the server. For this, we employed a model which assumes a linear relationship between CPU utilization and the power consumption and derived the configuration parameter from historical measurements. Integrating the power prediction over the time of the experiment we were able to predict the energy consumption with an error of less than 7%.

4 Conclusion

This paper presents our model-based description of time-triggered events to take into account user actions which impact application workload profiles and allows us to evaluate VM placement. The analysis provides IaaS operators with the means to reason on the performance of optimization policies employed by their cloud middleware in a realistic experiment context. Employing our approach we were able to predict the energy consumption of the compute node under test with an error of less than 7%.

Our approach of monitoring the data center and storing historical data in a dedicated database (HBase) allows us to derive workload models of individual VMs, as well as, experiment scenarios. The cost of using the approach is limited to operating the database. Overhead resulting from the required monitoring can be neglected, as model extraction is based on VM performance metrics measured by default, particularly VM allocation and VCPU utilization.

So far, the experiment scenario support for VMs is limited to start and stop requests. A strategy to cope with resource congestion in HPC environments is to pause non-critical VMs for the duration of the shortage. As part of future work, we plan to extend the experiment scenario simulation with support for pausing and resuming a VM execution.

Acknowledgments

The research leading to these results has received funding from the European Union's Seventh Framework Programme under grant agreement no. 610711.

References

[1] S. Becker, H. Koziolek, and R. Reussner. "The Palladio component model for model-driven performance prediction". In: *Journal of Systems and Software* 82.1 (2009). Special Issue: Software Performance - Modeling and Analysis, pp. 3–22.

[2] R. N. Calheiros et al. "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms". In: *Softw. Pract. Exper.* 41.1 (2011), pp. 23–50.

[3] H.-J. Werner et al. "Molpro: a general-purpose quantum chemistry program package". In: *Wiley Interdisciplinary Reviews: Computational Molecular Science* 2.2 (2012), pp. 242–253.

[4] T. W. Wlodarczyk. "Overview of Time Series Storage and Processing in a Cloud Environment". In: *CloudCom 2012*. 2012, pp. 625–628.

[5] M. Becker, M. Luckey, and S. Becker. "Performance Analysis of Self-Adaptive Systems for Requirements Validation at Design-Time". In: *Proceedings of QoSA'13*. 2013.

[6] G. Sakellari and G. Loukas. "A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing". In: *Simulation Modelling Practice and Theory* 39 (2013). S.I.Energy efficiency in grids and clouds, pp. 92–103.

[7] P. O. Östberg et al. "The CACTOS Vision of Context-Aware Cloud Topology Optimization and Simulation". In: *CloudCom 2014*. 2014, pp. 26–31.

[8] H. Groenda and C. Stier. "Improving IaaS Cloud Analyses by Black-Box Resource Demand Modeling". In: *Symposium on Software Performance 2015*. 2015.

[9] R. Heinrich et al. "Integrating business process simulation and information system simulation for performance prediction". In: *Software & Systems Modeling* (2015), pp. 1–21.

[10] C. Vögele et al. "Modeling Complex User Behavior with the Palladio Component Model". In: *Softwaretechnik-Trends*. Vol. 35(3). 2015.

[11] CACTOS Consortium. *D6.4: Supporting Documentation for the Software Deliverable: CactoSim Simulation Framework Final Prototype*. Tech. rep. 2016.

[12] C. Stier and A. Koziolek. "Considering Transient Effects of Self-Adaptations in Model-Driven Performance Analyses". In: *Proceedings of QoSA'16*. accepted, to appear. 2016.