Rapid Testing of IaaS Resource Management Algorithms via Cloud Middleware Simulation

Christian Stier
FZI Research Center for Information Technology
Karlsruhe, Germany
stier@fzi.de

Jörg Domaschka
Institute of Information Resource Management, Ulm University
Ulm, Germany
joerg.domaschka@uni-ulm.de

Anne Koziolek
Karlsruhe Institute of Technology
Karlsruhe, Germany
koziolek@kit.edu

Sebastian Krach
FZI Research Center for Information Technology
Karlsruhe, Germany
krach@fzi.de

Jakub Krzywda
Department of Computing Science
Umeå University
Umeå, Sweden
jakub@cs.umu.se

Ralf Reussner
Karlsruhe Institute of Technology
Karlsruhe, Germany
reussner@kit.edu

ABSTRACT

Infrastructure as a Service (IaaS) Cloud services allow users to deploy distributed applications in a virtualized environment without having to customize their applications to a specific Platform as a Service (PaaS) stack. It is common practice to host multiple Virtual Machines (VMs) on the same server to save resources. Traditionally, IaaS data center management required manual effort for optimization, e.g., by consolidating VM placement based on changes in usage patterns. Many resource management algorithms and frameworks have been developed to automate this process. Resource management algorithms are typically tested via experimentation or using simulation. The main drawback of both approaches is the high effort required to conduct the testing. Existing Cloud or IaaS simulators require the algorithm engineer to reimplement their algorithm against the simulator's API. Furthermore, the engineer manually needs to define the workload model used for algorithm testing. We propose an approach for the simulative analysis of IaaS Cloud infrastructure that allows algorithm engineers and data center operators to evaluate optimization algorithms without investing additional effort to reimplement them in a simulation environment. By leveraging runtime monitoring data, we automatically construct the simulation models used to test the algorithms. Our validation shows that algorithm tests conducted using our IaaS Cloud simulator match the measured behavior on actual hardware.

ACM Reference Format:
Christian Stier, Jörg Domaschka, Anne Koziolek, Sebastian Krach, Jakub Krzywda, and Ralf Reussner. 2018. Rapid Testing of IaaS Resource Management Algorithms via Cloud Middleware Simulation. In *ICPE '18: ACM/SPEC International Conference on Performance Engineering, April 9–13, 2018, Berlin, Germany*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3184407.3184428

ICPE '18, April 9–13, 2018, Berlin, Germany

ACM ISBN 978-1-4503-5095-2/18/04…$15.00
https://doi.org/10.1145/3184407.3184428

1 INTRODUCTION

IaaS Cloud services allow users to deploy a distributed application in a virtualized environment without having to customize their application to a specific PaaS stack. It is common practice to host multiple VMs on the same server. The shared hosting of VMs reduces operational cost. Traditionally, IaaS data center management required manual effort for optimization, e.g. by consolidating VM placement based on changes in usage patterns. Autonomic resource management addresses this problem by automating the allocation of virtual to physical resources. Resource management frameworks continuously optimize, e.g., the mapping of VMs to servers. For this, they can leverage adaptation actions like VM migration.

However, the design and selection of autonomic resource management algorithms for IaaS data centers is a challenging task. In particular, the performance of the algorithms varies [14]. Theoretical guarantees on the performance of resource management algorithms are only valid under impractical assumptions and thus cannot directly be used for the design and selection. The selection of resource management algorithms depends on the Quality of Service (QoS) goals of an IaaS data center operator, and tenant Service Level Agreements (SLAs). The selection of an algorithm thus requires an informed trade-off between conflicting goals [14].

The experimental evaluation of algorithms, e.g. via benchmarking, requires large data center testbeds. This is both time consuming and costly. *Cloud simulators* like CloudSim [5] or GreenCloud [12] offer reproducible conditions for algorithm testing. Once defined, it is possible to use the same workload scenarios to compare different resource management algorithms and configurations.

However, existing IaaS Cloud simulators [5, 12, 18] have specialized APIs, against which resource management algorithms need to be implemented. The re-implementation of algorithms for specific simulators is a challenging task. It requires expert knowledge of the simulator execution semantics, and their correspondence to the managed elements of the runtime management algorithms. Changes made to the algorithm need to be implemented in the runtime and simulation variant of the algorithm. This induces significant effort. Non-expert users, e.g. data center operators, have to rely on the availability of an algorithm implementation for their IaaS simulator of choice.

Another difficulty of simulation-based evaluations is the acquisition of representative and accurate simulation models. The manual construction of simulation models, either in code, or graphical editors, requires significant effort. Furthermore, it requires detailed knowledge of the level of abstraction of the input model used by the simulator.

In this paper, we present an approach to integrate native resource management algorithm implementations into a data center simulation tool. The integration of native resource management algorithm implementations with the data center simulation has two main advantages. First, it removes the need to re-implement the algorithm against a simulation specific interface. This makes it easier to test resource management algorithms using simulation. Second, an evaluation of the actual algorithm implementation increases confidence that it performs as intended.

We build upon our previous work, in which we suggested a generic approach to couple run-time models and simulation models [19]. The implementation of the integration approach leverages instances of the *Adaptation Action* metamodel [20] to define reusable and composable model-to-model transformation rules.

To address the challenge of simulation model acquisition, we present an automated simulation model extraction approach in order to reduce the effort for simulation model acquisition. Our approach reconstructs IaaS workloads, including VM submission and termination requests from historical measurements. Thereby, we enable the reconstruction of complex workloads. This enables the evaluation of runtime management algorithm performance under varying load. We represent extracted VM submissions using our timeline-based modeling language. In a previous short paper [13], we introduced this language and an early evaluation of the language. Algorithm engineers and data center operators can easily modify an extracted model to evaluate alternative scenarios.

To summarize, the contributions of this paper are:

(1) An approach to integrate native resource management algorithm implementations into a data center simulation tool,
(2) A simulation model extraction approach that automatically reconstructs timeline-based, complex workload scenarios,
(3) An evaluation of (1), (2), and the timeline-based modeling language [13].

We evaluated our approach for a diverse set of IaaS workloads. We used a set of scientific computing workloads to investigate the accuracy of extracted simulation models. The workloads were constructed based on expert knowledge on typical workloads submitted at the High Performance Computing Center at Ulm University. We investigated the consistency of resource management decisions between simulation and a real world IaaS testbed deployment at Ulm University. We showed that our integrated approach accurately predicts data center utilization and power consumption metrics. Resource management decisions performed in the simulation are consistent with the behavior in the IaaS testbed. Finally, we illustrated the benefit of simulation-based testing for the selection of power management and autoscaling algorithms. The simulation based evaluation of a power management algorithm showcased significant savings in energy consumption. Simultaneously, the algorithm did not compromise the deployment of VMs. The comparison of autoscalers enabled us to evaluate which algorithm better suited the scalability requirements for the investigated workload.

This paper is organized as follows: Section 2 describes the foundations of our work. In Section 3 we illustrate an example use case of our approach. Section 4 describes how native resource management algorithm implementations can be integrated into our data center simulation tool. Section 5 describes our simulation model extraction approach. The evaluation is presented in Section 6. Section 7 compares our work to related work and Section 8 concludes.

2 FOUNDATIONS

2.1 The CACTOS Project

The CACTOS project [16] developed an approach for the autonomic management of IaaS Cloud data centers. As part of the CACTOS project, two toolkits were developed. The *CACTOS Runtime Toolkit* integrates monitoring and resource management via a variety of algorithms. The *CACTOS Prediction Toolkit* supports the systematic evaluation of alternative data center deployment scenarios. This paper focuses on the prediction toolkit.

2.1.1 CACTOS Runtime Toolkit. The CACTOS Runtime Toolkit is designed to support different IaaS Cloud platforms. The toolkit integrates with these platforms to offer enhanced resource management capabilities. As part of the project, OpenStack [15] and Flexiant Cloud Orchestrator (FCO) [8] support was developed. CACTOS uses an optimization framework, CactoOpt [1], to derive adaptation action plans. The algorithms offered by CactoOpt [1] aim for different QoS trade-offs. In order to achieve these trade-offs, the algorithms formulate adaptation actions. They span the initial placement of VMs, VM migration, server level power management, and further resource management actions. CactoOpt also offers autoscaling capabilities to horizontally scalable, multi-tier applications. Autoscaling algorithms supported by CACTOS cover *Hist*, *ConPaaS*, *Reg* and *React* [9]. The CACTOS Runtime Toolkit uses a runtime model, which was built specifically to automate the management of heterogeneous IaaS data centers. The CactoOpt algorithms derive their plans from the runtime state that is represented in the runtime model. A set of Cloud middleware components execute the adaptation actions in the data center environment.

2.1.2 CACTOS Prediction Toolkit. The development, selection and parametrization of resource management algorithms are complex tasks. The CACTOS project developed the CACTOS Prediction Toolkit to support what-if analyses for IaaS Cloud data center environments. The CACTOS Prediction Toolkit builds upon the Palladio Component Model (PCM) software performance model [3], and the self-adaptive software systems simulator SimuLizar [2]. The toolkit supports the simulation of applications modeled at different levels of details, ranging from black-box VMs to detailed application architecture models. This paper contributes extensions to the toolkit that enable data center operators and algorithm engineers to systematically investigate the effectiveness and efficiency of resource management algorithms. Our work builds upon [19, 21, 11], which we discuss in the following.

2.2 Achieving Model Consistency between Runtime and Simulation Models

The model consistency approach described in [19] leverages correspondence models and correspondence rules to synchronize runtime and simulation models. The correspondence model holds the

relationship between entities in the runtime and simulation model. Correspondence rules can be subdivided in two categories. *Mapping operations* synchronize the runtime model with changes in the simulation. Example changes are updates of measurements in simulation. The updated simulation measurements need to be propagated to the runtime model. This enables runtime management mechanisms to observe and react to changes in load. *Adaptation enactment rules* enact and synchronize the effect of adaptation decisions made by autonomic resource management mechanisms. This paper applies [19] to support the evaluation of resource management algorithms in the CACTOS Prediction Toolkit.

2.3 Timeline-Based Experiment Scenarios

In order to assess the performance of IaaS data center optimization algorithms, algorithm engineers and data center operators need workload models that are representative for the intended use cases of the algorithms. Our *Experiment Scenario* [13] metamodel enables the specification of complex user interactions with data centers. Instances of the metamodel represent interactions of users with a data center as a timeline of events.

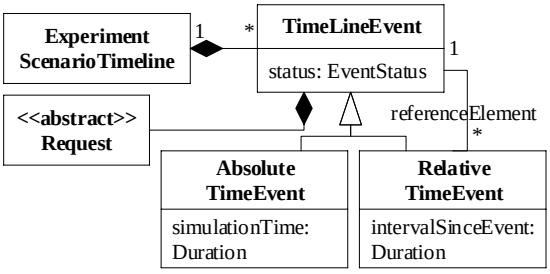

Figure 1: Excerpt from the Experiment Scenario metamodel.

Figure 1 depicts the central classes from the metamodel. The *ExperimentScenarioTimeline* consists of *TimeLineEvents*. Each event maps a *Request* model element to the timeline. There are two types of events. *AbsoluteTimeEvent* specifies an absolute point in time at which a request should be triggered. *RelativeTimeEvent* defines the request time relative to another event. An example request type is *StartApplicationRequest*. *StartApplicationRequest* models a user request to start a new individual VM or distributed application. The request references an application template that is used to assemble the application. Reconfigurations of the data center caused by manual intervention of operators can be expressed with appropriate requests. For an overview of all supported types we refer to [13].

In simulation, a dedicated event scheduler processes the *AbsoluteTimeEvents* ordered by ascending simulation time. The scheduler tracks the execution status of events in their *EventStatus*. Adaptation enactment rules trigger the execution of the requests.

Figure 2 depicts an example excerpt from an Experiment Scenario model based on one of our evaluation scenarios. It contains a start-up request for the VM *instance-1e22*. The scenario prescribes that *instance-1e22* should be started at simulation time 1747s, and terminated 1780s later. The startup request references the VM template, used VM flavor and input parameters. The terminate request references the prior *StartApplicationRequest*, as the VM to be terminated is not yet running in the initial simulation model.

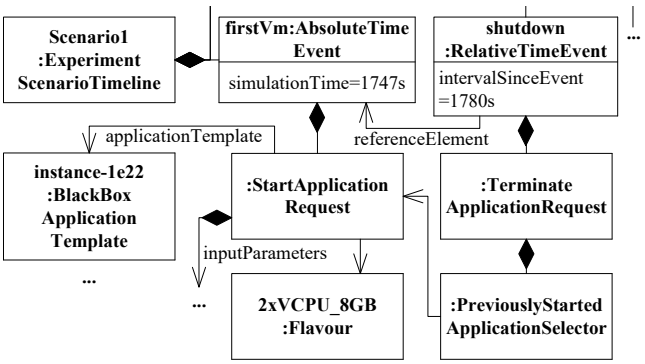

Figure 2: Excerpt from example Experiment Scenario model.

2.4 Extracting Data Center Simulation Models

Svorobej et al. [21] present an initial approach that leverages a runtime model snapshot from the CACTOS Runtime Toolkit as foundation for simulations. The runtime model lacks information on historically executed VMs and their workloads. The runtime model thus can only be used to evaluate how runtime management algorithms would perform under stable load conditions. Kistowski et al. [11] sketches an algorithm for the reconstruction of black-box resource demand functions from a series of load measurements.

This paper contributes a novel model extraction approach that supports the reconstruction of timeline-based workload models from historical measurements. It leverages [21] to gather basic infrastructure information, i.e., on the available servers. Our model extraction approach applies the algorithm from [11] to reconstruct workload models for individual VMs.

3 EXAMPLE USE

A data center operator might use our approach as follows. As a starting point, she might be interested in evaluating how the introduction of automated resource management would affect the performance and efficiency of a manually managed data center. For this, the data center operator can install the monitoring tools provided by the CACTOS Runtime Toolkit to gather monitoring data. Next, the operator applies our simulation model reconstruction methods to the data. The operator then can simulate how the application of an existing runtime resource management algorithm implementation would have affected the performance and efficiency of the data center for this scenario under investigation.

The use of existing algorithm implementations and automated model construction significantly reduces the evaluation effort for the data center operator. It rules out inconsistencies between simulation and runtime implementation variants. It thus increases confidence in the simulation results. Once she has invested the initial effort for the monitoring setup, the operator can continuously reevaluate and compare different algorithms. This enables the operator to adapt the algorithm choice and configuration to changes in the data center setup, workload and performance requirements.

4 INTEGRATING MIDDLEWARE-SPECIFIC RUNTIME MANAGEMENT ALGORITHMS WITH SIMULATION

Runtime management frameworks use *runtime models* to manage resources. Runtime management algorithms leverage information from the runtime models to plan adaptation decisions. Existing Cloud and simulation frameworks lack support for simulating these algorithms in their middleware-specific implementation. This section presents our approach for the integration of middleware-specific runtime management algorithms with an IaaS Cloud simulator. It enables algorithm engineers and data center operators to test and evaluate algorithms, while requiring minimal knowledge of the simulation API.

4.1 Information Gap between Runtime Models and Simulation Models

Software system simulators like SimuLizar [2] or CloudSim [5] naturally abstract from information that is not needed to predict the metrics which are relevant to the use cases of the simulator. The abstraction covers characteristics of the hardware and software stack. This simplifies the simulation, as well as the construction of input models for simulator users.

The level of abstraction chosen when modeling individual entities depends on the pragmatism of the simulation. Many software performance simulators do not model memory [2, 3, 12], as (i) memory accesses are difficult to predict, and (ii) their effect on QoS is considered negligible for CPU-bound applications.

Runtime models are designed to support autonomic resource management. This contrasts the pragmatism of design time performance models like PCM [3]. Design time models focus on modeling system characteristics that impact performance. Runtime models capture all characteristics which are relevant to the management of a system. In IaaS data centers, this can include user management information and detailed VM instantiation parameters. Unlike design time performance models, runtime models may not capture information on user and application behavior on a level that is detailed enough for performance simulations.

A naive approach is to transform the runtime models to performance models of the simulator. This approach, however, requires that all resource management algorithms are reimplemented against the model of the simulator. Runtime management algorithms, which consider properties that are not reflected in the simulation model, can not be simulated. We designed an approach for achieving model consistency between runtime models and simulation models. Our approach supports simulation-based analysis and testing of optimization algorithms without the need to modify or re-implement the algorithms.

4.2 Achieving Model Consistency

We achieve model consistency by implementing the approach discussed in Section 2.2 to achieve model consistency between the CACTOS runtime model and the PCM simulation model. A specialized metamodel maintains the correspondence between runtime and simulation model. In total, the metamodel distinguishes 40 correspondence types.

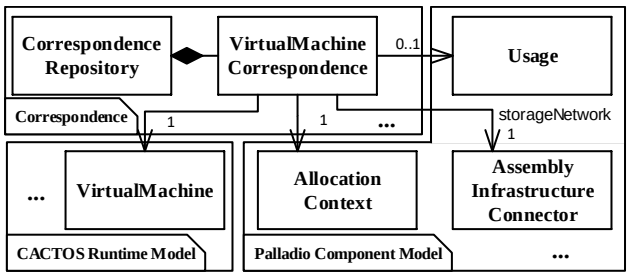

Figure 3: Excerpt from the correspondence metamodel between runtime and simulation model.

Figure 3 provides an example correspondence from this metamodel. Each *VirtualMachine* in the CACTOS runtime model corresponds to multiple entities in the simulation model, the PCM. The *AllocationContext* represents the deployment of a set of software components to a server. It can be used to represent VM allocations in PCM. However, the correspondence with allocated components fails to cover all aspects that are needed to enable meaningful simulations. The CACTOS runtime model contains detailed storage information of VMs. This includes, e.g., the location of their network attached devices. In order to reason the effects of remote storage accesses, the storage characteristics need to be mapped to PCM. Unlike the runtime model, the PCM explicitly models user interactions with the VM components. The correspondence covers this with the reference to the *Usage* model.

A model-to-model transformation from the CACTOS runtime model to PCM establishes the initial correspondence between runtime and simulation model. The mapping operations update the correspondence model during simulation. We implemented the operations as modular model-to-model transformations for the model-based simulated runtime management of SimuLizar [2, 20]. The implementation uses the Adaptation Action metamodel [20] to specify the mapping operations in a reusable manner.

4.3 Implementation of Runtime Management Integration

We integrated all runtime management algorithm types supported by the Cloud middleware of the CACTOS Runtime Toolkit with the simulation environment of the CACTOS Prediction Toolkit. This enabled us to support all algorithms of the same type. No additional effort was required to integrate specific algorithms with the simulator.

A *Placement Connector* bridges the gap between the simulated runtime management, and the placement algorithms of CactoOpt. When a new VM startup request is issued, the simulated runtime management calls the native optimization implementation via the placement connector. The model consistency mechanism discussed in Section 4.2 realizes the simulated runtime management.

We integrated data center optimization algorithms via an *Optimization Connector*. This connector couples the resource management optimization algorithms and heuristics of CactoOpt with the simulation. The Optimization Connector functions similar to the Placement Connector. Instead of a single supported decision, i.e.,

placement, the consistency mechanism translates a wide range of server and VM reconfiguration decisions to simulation. It uses the adaptation enactment rules for this purpose.

The *Autoscaling Connector* communicates with the native autoscaling middleware of CACTOS. The middleware manages autoscaling on a per-application basis. It uses the runtime model passed by the simulation to decide on horizontal scaling. Instead of measurements from a real world data center, the simulation runtime management exposes simulated measurements to the autoscalers. Mapping operations of the rule-based synchronization engine update the measurement representations in the runtime model, as Section 4.2 described.

5 AUTOMATED SIMULATION MODEL CONSTRUCTION

A major challenge in the application of simulations is the acquisition of simulation models. It is more attractive to reason on the performance of resource management algorithm using simulation if the models can be obtained with little effort. We implemented an approach that enables the evaluation of resource management algorithm performance using automatically constructed performance models. Our approach uses the historical information collected by the CACTOS Runtime Toolkit to construct performance models.

5.1 Black-Box Performance Models

Black-box VM workload models describe VM workloads in terms of their resource usage over time. The main benefit of black-box VM models is that they do not require insight into the applications deployed in a VM. As they only depend upon metrics and topology information available to the data center operator, they can be constructed for any VM based on its past observed behavior. We construct the black-box models from past load measurements, which were recorded in the historical database of the CACTOS Runtime Toolkit. We normalize the load levels of VMs using the processing speed of their original host to account for VM migrations.

5.2 Timeline-Based Experiment Scenarios

Data center operators and algorithm engineers can use instances of the Experiment Scenario model to evaluate complex user interaction with the simulated data centers. Section 2.3 introduced the Experiment Scenario metamodel, and provided an overview of supported user interactions, e.g., VM submissions. The manual modeling of Experiment Scenarios requires expert knowledge of potential VM submission patterns, typical VM configurations, and workloads. We realized an automated approach for the reconstruction of Experiment Scenarios from historical measurements. It enables data center operators and algorithm engineers to evaluate resource management algorithms based on past workloads and user interactions.

Our approach uses recorded VM submission and reconfiguration events to reconstruct an Experiment Scenario timeline. We link each VM submission to a black-box performance model, which is automatically constructed. In order to construct an Experiment Scenario model, the user specifies a period of time, and a subset of servers she is interested in. We translate this to a set of queries on a historical measurement database. The data from these queries is funneled into the reconstruction of user interactions. We enrich the Experiment Scenario with VM instantiation parameters. This increases the accuracy of placement decisions in simulation. Placement algorithms [1] consider these parameters to determine if VMs can be deployed on the same server without causing resource contention.

5.3 Power Models

Power consumption decisively determines the operational cost of data centers. System-level power models [17] enable power consumption predictions of individual servers based on metrics like CPU utilization. Power models need to be trained for specific servers in order to make accurate power consumption predictions. We realized an approach that uses the historical data collected in the historical measurement database as the source of training data for statistical power model learning. For a given time frame, we query the collected measurements. We use this data as input to power model training. A non-linear regression technique trains a given power model type on the selected training data set.

5.4 Limitations

The reconstruction of black-box VM and timeline-based scenario models is well suited to evaluate the performance of resource management algorithms for workloads observed in the past. It is of limited use to explore scenarios that involve user-facing applications with varying workloads. For this, other means of performance model acquisition are more suited. The extraction of server power models from historical measurements requires that the server has run workloads which cover the utilization ranges investigated in simulation.

6 EVALUATION

In order to evaluate our approach, we compared simulation results and measurements for a set of experiments conducted in a data center testbed.

6.1 Scientific Computing

We evaluated the applicability of our approach using a case study from the scientific computing domain.

6.1.1 Scenario Description. We conducted the case study in a commodity hardware testbed. The testbed was operated using the OpenStack Cloud middleware in combination with the CACTOS Runtime Toolkit. The servers ran KVM hypervisors. The CACTOS Runtime Toolkit contributed autonomic resource management. In each of the evaluated configurations, VM placement and migration algorithms were in use. We used IPMI to collect power consumption measurements of the servers over time.

In order to evaluate the accuracy of our integration and model extraction methods, we proceeded as follows. First, we ran an experiment in an IaaS data center testbed. Second, we obtained a simulation model by applying our model extraction method. We applied our Experiment Scenario extraction to obtain models of the user interactions, and VM black-box workload models from the experiment run. Next, we conducted a simulation using the resulting input models. We used the same algorithm implementations and

configurations for the simulated run as in the testbed experiment run. Finally, we compared measured and predicted results.

The following outlines the scenarios for which we conducted the experimental evaluation. Every scenario encompassed a set of scientific computing workloads. Specifically, we executed a set of Molpro [23] workloads. Molpro is a framework for quantum chemistry calculations. Molpro follows run-to-completion semantics, as is common for scientific computing applications. A compute job submission system translated each scientific compute job submission request to a VM submission request on the IaaS testbed. Per scenario, a load driver submitted the jobs over time based on a predefined submission schedule. We constructed the submission schedule to resemble a typical daily cycle of user job submissions in the High Performance Computing Center at Ulm University. The job submissions consist of a mix of long and short running jobs. The short jobs reach execution times of up to two hours. Long running jobs may span eight to ten hours.

Scenario 1. The first scenario covered 26 VM submissions to a testbed setup which consisted of eight servers. Six of these eight servers had a power meter, from which we could collect measurements. The experiment lasted just short of one and a half hours. We used consolidation algorithms for both VM placement and migration. They consolidated the VMs based on their RAM requirements.

Scenario 2. The second scenario consisted of 15 compute job submissions. It covered a run time of approximately eight and a half hours. We allocated the same eight servers as Scenario 1. We configured the Runtime Toolkit to use load balancing algorithms for both VM migrations and placement. The algorithms aim to evenly distribute the VMs on all servers based on their RAM requirements.

Scenario 3. Scenario 3 encompassed 19 compute job submissions. It covered the same basic experiment as Scenario 1, but with an extended run time of eight hours and 46 minutes. The 26 VM submissions from Scenario 1 were reduced to 19. The VMs were hosted on the same set of eight servers. We used RAM based consolidation algorithms for VM placement and migrations.

Scenario 4. The fourth scenario consisted of 37 Molpro job submissions. It covered a run time of roughly 26 hours. It used six servers from the IaaS testbed. We could collect power measurements from four of these six servers. Scenario 4 used the same migration and placement algorithms as Scenario 1 and 2.

6.1.2 Results. For Scenario 1, the algorithms placed the VMs on the same servers as in the measured experiment. In order to quantify the prediction accuracy over the duration of the experiment, we compared the predicted and measured accumulated energy consumption of all servers with power metersusing the error formula $|\frac{E_{\text{Meas}}-E_{\text{Sim}}}{E_{\text{Meas}}}|$, where E is the aggregate energy consumption.

In Scenario 1 we employed a linear power model to predict the energy consumption of the servers. The linear model was trained using historical measurements from each server. We used more complex power models for the other scenarios, e.g., with exponential components. Table 1 lists the measured and predicted total energy consumption over each run. In Scenario 2, the energy consumption prediction reached a prediction error of 0.39%. Scenario 3 had the highest prediction error at 7.08%.

Table 1: Total measured and predicted energy consumption for the four evaluated scenarios, with prediction error. Duration in minutes. Energy consumption in W h, error in %.

Scenario	Duration	Measured	Predicted	Error
1	75 min	1 783 W h	1 661 W h	6.85%
2	514 min	5 443 W h	5 464 W h	0.39%
3	526 min	5 238 W h	5 609 W h	7.08%
4	1561 min	13 558 W h	12 826 W h	5.40%

We could trace back the source of the prediction error for Scenario 3 to a lack of historical measurements from one of the VMs. While the VM was running in the experiment, its measurements were not recorded due to the failure of VM internal monitoring. Thus, our tooling was unable to reconstruct a behavior model of the VM. Down the line, this led to the placement of a highly active VM on one of the servers with a power meter. This increased the predicted energy consumption.

6.2 Power Management

In order to validate that our algorithm integration approach also supports the analysis of power management algorithms, we applied an existing algorithm to the Scenario 3 workload. We configured VM migration and placement to consolidation algorithms. This enabled the power management algorithm to turn off free servers. Our simulations were able to show significant power savings, without negatively affecting the deployment of new VMs.

6.3 Autoscaling

The selection of the right autoscaling algorithm for an application is a challenging task. This section explores how we can employ our simulation-based method to compare different autoscaling policies for an enterprise web application.

6.3.1 Compared Autoscalers. We evaluated which of two autoscalers performed the best for the evaluated enterprise application. We compared the two autoscalers *React* and *Reg*.

React [6] is a rule-based autoscaler. Its algorithm increases the number of active instances of a scalable application tier if the measured user workload surpasses a specified threshold capacity. If at least two instances are under-utilized, React shuts down and decomissions one instance.

Reg [10] is an autoscaler, which scales the number of active instances based on a regression model. If the measured load falls below a specified threshold, the autoscaler reduces the number of active instances. It uses a regression model to determine the number of active instances, which should remain active. For user workload levels higher than a threshold capacity, Reg initiates the startup of additional instances.

6.3.2 Case Study System. DataPlay[1] is a horizontally scalable multi-tier enterprise web application. It is a gamified social platform for data exploration. DataPlay follows a three-tier architecture style, where the business tier can be horizontally scaled.

[1] https://github.com/cactos/DataPlay, last retrieved 24.10.2017.

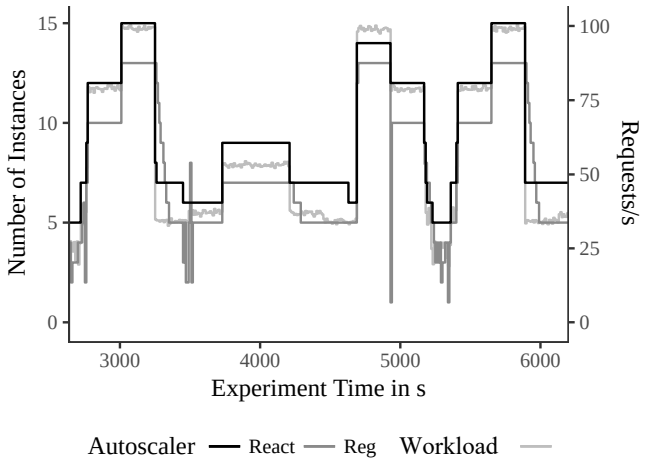

Figure 4: Experimental dynamics of the two simulated autoscaling policies. Excerpt from the full experiment, which spanned a time frame of over 11 hours. The black line shows the workload intensity as requests per s. The grey lines represent the number of active instances over time, which the autoscalers allocate.

We obtained the input model for our simulation by enriching a runtime model snapshot of the data center. The snapshot contained a description of its server infrastructure, and an application model of DataPlay. We instantiated the application at the beginning of the simulated experiment using our Experiment Scenario model.

We used a synthetic workload that covered a wide range of workload intensities, and workload variations. The workload covered a time frame of over eleven and a half hours of simulation time. Over this period, a seasonal pattern repeated sixteen times. The workload reached approximately 100 requests per second at its peak. It contained short periods with request rates just above, or at zero. The seasonal pattern was folded with uniform noise in the interval $[-3, 2]$ requests per second.

6.3.3 Results. We look at an excerpt of the whole experiment results to discuss our findings. Figure 4 illustrates the experimental dynamics of the autoscalers and the workload. The *workload* line shows the rate at which requests arrived at the DataPlay application. The grey lines represent the total number of VMs over time, which the autoscaling algorithms suggested to keep allocated. The behavior of both algorithms differed significantly. Reg frequently triggered scale-out and scale-in decisions. Particularly, Reg overeagerly performed scale-ins once the workload started to decrease. This led to staggered scaling, e.g., at around 3500 experiment time. React over-provisioned VMs. Compared to Reg, it allocated more or an equal number of VMs most of the time. React recommended to operate 9.12 VMs on average, while Reg only proposed 7.47. Over the course of the experiment, Reg issued 2544 scale-in or scale-out actions, while React only adapted 647 times. The higher frequency of Reg led to a larger overhead for (de-)commissioning VMs.

The poor performance of Reg is in line with the experimental comparison of autoscalers by Ilyushkin et al. [9]. In their experiments, Reg also under-provisioned VMs and quickly varied the number of active VMs. The authors only could improve the performance of Reg, once they implemented a set of improvements to the original algorithm and its implementations. React manages to match demand in most periods, or overprovisions.

In conclusion, we determined that React provides better operational stability at the cost of light overprovisioning. Thus, we consider React to be better suited as a autoscaler for DataPlay for the investigated workload patterns. We did not record performance metrics, such as response time, and average, minimum and maximum CPU utilization as part of our comparison. In future work, we plan to compare the autoscaling policies based upon these further metrics, and the metrics outlined in Ilyushkin et al. [9].

7 RELATED WORK

The simulation-based evaluation of Cloud resource management has been a topic of great interest in recent years. Sakellari and Loukas [18] provide an overview of Cloud simulators. Two popular IaaS Cloud simulators discussed by the survey are CloudSim [5] and GreenCloud [12]. Both support the evaluation of resource management algorithms. However, they require a reimplementation of the algorithms for the simulator specific APIs.

Vondra and Šedivý [22] present a Cloud simulator that has been built for the simulation-based evaluation of autoscaling algorithms. Like CloudSim and GreenCloud, their simulator requires a reimplementation of each algorithm for the simulator interface.

CDOSim [7] extends CloudSim [5] to support the evaluation of enterprise Cloud application migration scenarios. CDOSim offers an approach to extract white-box application models using static code analysis, and dynamic instruction counting. This requires source-code level access to, and extensive profiling of the evaluated Cloud application. In the context of our work, it could be applied to extract white-box application models.

Calheiros et al. [4] propose a profiling based approach to construct coarse grey-box workload models of applications. Their approach requires dedicated profiling infrastructure. It profiles the Cloud application with varying workload intensities and compute resources. This complements our black-box model extraction approach. Unlike our approach it can, however, not be applied to model arbitrary VM workloads.

Ilyushkin et al. [9] experimentally evaluate a set of seven state of the art autoscaling algorithms. The authors evaluate the algorithms for scientific computing workloads. The comparison required a complex IaaS testbed setup and extensive experiments. Our work aims to reduce the effort for testing using simulations. Indeed, we were able to evaluate two of the algorithms from [9], of which we had the implementations.

8 CONCLUSION

This paper presents an approach for rapid testing of resource management algorithms for IaaS Cloud data centers. Our approach enables algorithm engineers and data center operators to evaluate IaaS Cloud resource management algorithms using simulations. Our simulation-based approach supports the simulation of algorithms which are natively implemented for Cloud middleware. The CACTOS Prediction Toolkit implements our approach for the CACTOS Runtime Toolkit, and its supported adaptation actions. These

actions include the initial placement of VMs, VM migration, power management and autoscaling. We show that our integration approach enables reasoning on the performance of diverse types of resource management algorithms.

We evaluated our approach for a diverse set of real-world workloads. We evaluated a set of resource management algorithms for scientific computing application workloads. The results from simulation have a high accuracy for energy consumption and utilization measurements. VM placement and migration decisions are consistent between the measured and simulated experiments. Our simulation enabled us to explore the effect of active power management algorithms on total consumption, and the reliability of VM placements. Thereby, the simulation-based evaluation helps avoid scenarios where power management interferes with the ability of a data center to serve all VM submission requests. We applied our approach to compare two autoscalers for an enterprise web application. Our observations on the autoscaler dynamics from simulation are consistent with published experimental evaluations [9].

Our approach enables algorithm engineers and data center operators to rapidly evaluate the performance of IaaS resource management algorithms. It requires no additional effort or in-depth knowledge of simulation models and APIs. The users of our approach can simply evaluate their existing resource management algorithm implementations. We automate the construction of simulation models. For this, we leverage existing runtime models and historical measurements.

We plan to expand our approach in two directions. First, we aim to automate the construction of detailed application models of scientific computing applications. This will reduce the effort for the construction of accurate simulation models, which consider the phases of scientific computing applications. Second, we plan to conduct case studies which investigate the prediction accuracy of application workloads with large heterogeneity between workloads, and used servers. We intend to expand the quantitative comparison of simulation and measurements of autoscaling policies to the metrics listed in Section 6.3.3.

ACKNOWLEDGMENTS

This work is funded by the European Union's Seventh Framework Programme under grant agreement 610711 (CACTOS), the Swedish Research Council (VR) project Cloud Control and the Swedish Government's strategic research project eSSENCE.

REFERENCES

[1] Ahmed Ali-Eldin, Per-Olov Östberg, Jakub Krzywda, Christopher Hauser, Jörg Domaschka, and Henning Groenda. 2017. Predictive Cloud Application Model: Project Deliverable D3.2. Tech. rep.

[2] Matthias Becker, Markus Luckey, and Steffen Becker. 2013. Performance Analysis of Self-Adaptive Systems for Requirements Validation at Design-Time. In *Proc. of the 9th ACM SigSoft Intl Conf on Quality of Software Architectures (QoSA'13)*. ACM, (June 2013).

[3] Steffen Becker, Heiko Koziolek, and Ralf Reussner. 2009. The Palladio component model for model-driven performance prediction. *Journal of Systems and Software*, 82, 1, 3–22.

[4] Rodrigo N. Calheiros, Marco A.S. Netto, César A.F. De Rose, and Rajkumar Buyya. 2013. Emusim: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications. *Software: Practice and Experience*, 43, 5, 595–612.

[5] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. 2011. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.*, 41, 1, (Jan. 2011), 23–50.

[6] Trieu C. Chieu, Ajay Mohindra, Alexei A. Karve, and Alla Segal. 2009. Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment. In *Proc of the IEEE Intl Conf on e-Business Engineering* (ICEBE). IEEE CS, 281–286.

[7] F. Fittkau, S. Frey, and W. Hasselbring. 2012. CDOSim: Simulating cloud deployment options for software migration support. In *2012 IEEE 6th International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA)*. (Sept. 2012), 37–46.

[8] [n. d.] Flexiant Cloud Orchestrator. Last retrieved 2017-10-26. Flexiant Ltd. https://www.flexiant.com/flexiant-cloud-orchestrator/.

[9] A. Ilyushkin, A. Ali-Eldin, N. Herbst, A. V. Papadopoulos, B. Ghit, D. Epema, and A. Iosup. 2017. An Experimental Performance Evaluation of Autoscaling Policies for Complex Workflows. In *Proc. of the 8th ACM/SPEC Intl Conf on Performance Engineering* (ICPE '17). ACM, L'Aquila, Italy, 75–86.

[10] Waheed Iqbal, Matthew N. Dailey, David Carrera, and Paul Janecek. 2011. Adaptive resource provisioning for read intensive multi-tier applications in the cloud. *Future Gener. Comput. Syst.*, 27, 6, (June 2011), 871–879.

[11] Jóakim Von Kistowski, Nikolas Herbst, Samuel Kounev, Henning Groenda, Christian Stier, and Sebastian Lehrig. 2017. Modeling and extracting load intensity profiles. *ACM Trans. Auton. Adapt. Syst.*, 11, 4, Article 23, (Jan. 2017), 23:1–23:28.

[12] D. Kliazovich, P. Bouvry, Y. Audzevich, and S.U. Khan. 2010. Greencloud: a packet-level simulator of energy-aware cloud computing data centers. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. (Dec. 2010), 1–5.

[13] Sebastian Krach, Christian Stier, and Athanasios Tsitsipas. 2016. Modeling IaaS Usage Patterns for the Analysis of Cloud Optimization Policies. *Softwaretechnik-Trends*, 36, 4.

[14] Sunilkumar S. Manvi and Gopal Krishna Shyam. 2014. Resource management for infrastructure as a service (iaas) in cloud computing: a survey. *Journal of Network and Computer Applications*, 41, Supplement C, 424–440.

[15] [n. d.] OpenStack. Last retrieved 2017-10-26. The OpenStack Foundation. https://www.openstack.org/.

[16] P-O Östberg et al. 2014. The CACTOS Vision of Context-Aware Cloud Topology Optimization and Simulation. In *Proc. of the Sixth IEEE Intl Conf on Cloud Computing Technology and Science (CloudCom)*. IEEE CS, Singapore, 26–31.

[17] Suzanne Rivoire, Parthasarathy Ranganathan, and Christos Kozyrakis. 2008. A Comparison of High-level Full-system Power Models. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems* (HotPower'08). USENIX Association, San Diego, California, 3–3.

[18] Georgia Sakellari and George Loukas. 2013. A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing. *Simulation Modelling Practice and Theory*, 39, 92–103. Special Issue Energy Efficiency in Grids and Clouds.

[19] Christian Stier and Henning Groenda. 2016. Ensuring Model Continuity when Simulating Self-adaptive Software Systems. In *Proc. of the Modeling and Simulation of Complexity in Intelligent, Adaptive and Autonomous Systems 2016* (MSCIAAS '16) Article 2. Society for Computer Simulation International, Pasadena, California, 2:1–2:8.

[20] Christian Stier and Anne Koziolek. 2016. Considering Transient Effects of Self-Adaptations in Model-Driven Performance Analyses. In *Proceedings of the 12th International ACM SIGSOFT Conference on the Quality of Software Architectures* (QoSA'16). ACM, Venice, Italy.

[21] Sergej Svorobej, James Byrne, Paul Liston, Peter J. Byrne, Christian Stier, Henning Groenda, Zafeirios C. Papazachos, and Dimitrios S. Nikolopoulos. 2015. Towards automated data-driven model creation for cloud computing simulation. In *Proceedings of the 8th International Conference on Simulation Tools and Techniques, Athens, Greece, August 24-26, 2015*, 248–255.

[22] T. Vondra and J. Šedivý. 2017. Cloud autoscaling simulation based on queueing network model. *Simulation Modelling Practice and Theory*, 70, Supplement C, 83–100.

[23] Hans-Joachim Werner, Peter J. Knowles, Gerald Knizia, Frederick R. Manby, and Martin Schütz. 2012. Molpro: a general-purpose quantum chemistry program package. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2, 2, 242–253.