Analyzing Cost-Efficiency of Cloud Computing Applications with SimuLizar*

Sebastian Lehrig Hendrik Eikerling
{sebastian.lehrig|hendrik.eikerling}@informatik.tu-chemnitz.de
Software Engineering Chair
Chemnitz University of Technology, Chemnitz, Germany

ABSTRACT

In cloud computing, software applications are potentially able to use only the computing resources that are minimally needed for performant operation. Because cloud providers provision such resources on a pay-per-use basis, software architects are interested in analyzing the operational costs that accrue for such applications, allowing architects to optimize for cost-efficiency. Current analysis approaches like Palladio focus on traditional performance metrics but lack support for cost-efficiency metrics. Therefore, software architects have to inaccurately estimate operational costs of their applications, potentially leading to economically unusable applications.

To tackle this problem, we integrated cost metrics into SimuLizar, a Palladio extension for analyzing cloud computing applications. Our integration allows architects to attach prices to computing resources used by software applications. In a proof-of-concept evaluation with a simple book shop, we show that our integration allows architects to analyze costs with high accuracy.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed System—*Cloud applications*; D.2.8 [**Software Engineering**]: Metrics—*Scalability, Elasticity, Efficiency*

Keywords

Metrics, Cloud Computing, Cost, Efficiency, SimuLizar

1. INTRODUCTION

The cloud computing paradigm allows software architects to design applications that dynamically adjust the amount of their computing resources: if the workload increases, the capacity of the application can be increased, and vice-versa in case of decreasing workload. The degree of this dynamic adjustment is described by the elasticity quality property [7]. In a perfectly elastic cloud environment, application capacity matches the demand as closely as possible at all times, therefore wasting no resources. Because cloud resources are provided in a pay-per-use fashion [9], elastic applications become more cost-efficient because the monetary overhead of provisioned resources is minimized.

In order to engineer for cost-efficiency, software architects require the ability to accurately analyze the cost accrued for application usage already at design-time. However, current design-time analysis approaches like Palladio [6] focus on traditional performance metrics and lack support for cost< metrics. Therefore, software architects have to inaccurately estimate operational costs of their applications, potentially leading to economically unusable applications.

In related works, initial attempts have been made to determine operational costs of cloud computing applications. Cost calculators (e.g., [1, 2]) require architects to estimate computing resource needs, however, architects have no means to check whether their estimates are suitable, e.g., regarding performance. Approaches that predict future costs based on historic data (e.g., [4]) require an already operating application and are prone to new workload patterns. Design-time analyses (e.g., [5, 8]) cannot assess costs for applications that change their computing resources over time.

Therefore, we integrated cost metrics into SimuLizar [5], a Palladio extension for applications that can vary their computing resource usage over time. The metrics were integrated using the Quality Analysis Lab (QuAL), which enables the measurement, visualization, and recording of metrics in Palladio, see Section 3.3. Our integration allows architects to attach prices to computing resources used by software applications. Afterwards, SimuLizar analyses can accumulate per-resource costs based on regular time intervals, e.g., once per hour. SimuLizar finally outputs cost-efficiency reports for overall operation costs and per-resource operation costs for specific time periods.

The contribution of this paper is a description of our novel cost-efficiency extension to SimuLizar. In a proof-of-concept evaluation, we use a simple book shop example that contains dynamically replicating computing resources. Our evaluation shows that our integration is applicable and provides accurate cost-efficiency analyses.

This paper is structured as follows. Section 2 introduces the online book shop and Sec. 3 introduces the fundamentals for integration. Afterwards, Sec. 4 documents our concrete integration of cost metrics, followed by Sec. 5, evaluating the integrated metrics using the book shop scenario. Finally, Sec. 6 presents related work and Sec. 7 concludes our findings.

2. MOTIVATING SCENARIO: BOOK SHOP

We use a simple online book shop as example and to evaluate the integration of our cost metrics. The shop starts to operate on two inter-connected resource containers hosted in a cloud computing environment (Figure 1). One of the resource container is a normal one, the other is a replicable

*The research leading to these results has received funding from the European Seventh Framework Programme (FP7/2007-2013) under grant no 317704 (CloudScale).

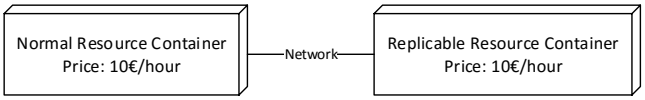

Figure 1: The Book Shop's Resource Environment

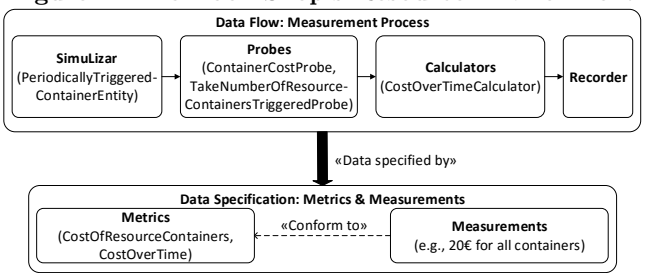

Figure 2: The Quality Analysis Lab

resource container. The replicable container replicates itself as soon as increasing workload causes response times above 3 seconds. Initial and replicated containers are billed according to the denoted price, e.g., each replica causes additional costs of 10€ per hour when it starts.

3. FUNDAMENTALS

The fundamentals required for the integration of cost-efficiency analyses in SimuLizar include SimuLizar (Sec. 3.1), details regarding cost-efficiency in cloud computing (Sec. 3.2), and the Quality Analysis Lab, a framework for extending quality analyses (Sec. 3.3).

3.1 SimuLizar Design-Time Analyses

SimuLizar [5] expands the functionality of Palladio by design-time analyses of systems that are self-adapting. Therefore, SimuLizar is suitable for analyzing cloud computing environments, which are inherently self-adaptive by definition [9]. In order to model such environments, SimuLizar provides tools for modeling self-adaptation rules like the rule of our book shop to replicate containers as soon as response times increase above 3 seconds.

3.2 Cost-Efficiency in Cloud Computing

In our systematic literature review [7], we define cost-efficiency as the relation between resource demand and actually consumed resources (measured in operation cost). A high cost-efficiency indicates that the minimal amount of computing resources is used for satisfying service level objectives like the 3-seconds response time threshold for our book shop. In cloud computing, elasticity allows applications to adapt resource usage to workload over time. Cost-efficiency is therefore an important property software architects have to trade off against other properties like availability.

3.3 Quality Analysis Lab (QuAL)

The Quality Analysis Lab (QuAL) is a framework that enables metric measurements in Palladio [3]. It enables the execution, storage, and visualization of such measurements, as well as the addition of custom metrics by developers.

QuAL focuses on the two aspects depicted in Fig. 2: data specification, i.e., the specification of metrics and measurements, and data flow, i.e., how and where probes take measurements and how they "flow" via calculators through the framework until they are recorded.

The following elements have to be provisioned to integrate custom analyses in QuAL:

Metrics Metrics specify the type of data to be analyzed. Metrics can be basic metrics, i.e., metrics that contain one value, or metric sets, which are composed of other metrics.

Probes Probes take measurements and relay them to calculators. There are two types of probes: Event and Triggered Probes. Event Proves take measurements when events occur, e.g., when a resource is added. Triggered Probes are explicitly triggered, e.g., by an Event Probe or periodically.

Calculators Calculators perform calculations based on the raw measurements delivered by probes.

4. INTEGRATION OF COST-EFFICIENCY ANALYSES IN SIMULIZAR

For providing cost-efficiency analyses, we opt for live analyses at simulation time instead of calculating cost in post processing, in order to enable self adaptation mechanisms during the simulation based on cost. To integrate these analyses into SimuLizar, we provide the following QuAL extensions[1] (cf., Fig. 2):

Cost-Efficiency Metrics We specify custom metrics that capture operation cost for resource containers. The *CostOfResourceContainers* metric captures operation cost of multiple resource containers. It is a rational-scaled basic metric with Euro as default unit. The *CostOverTime* is a metric set that subsumes the *CostOfResourceContainers* and the existing *PointInTime* metric, representing a point in simulation time measured in seconds. Using these metrics, it is possible to capture the total operation cost of multiple resource containers over simulation time.

PeriodicallyTriggeredContainerEntity In cloud computing, resources typically underly a billing period, i.e., a time period for which cloud providers charge resources. In our analysis, we measure the cost of resource containers after a billing period elapses. We accordingly implemented a *PeriodicallyTriggeredContainerEntity* that can be attached to resource containers and fires a trigger after a billing period.

ContainerCostProbe To measure the current cost of a container in a billing period, we devised a custom *ContainerCostProbe*. Each resource container has its own probe.

TakeNumberOfResourceContainersTriggeredProbe A triggered probe to measure the number of active resource containers.

CostOverTimeCalculator This custom calculator receives the measurements from the custom *ContainerCostProbe*, the *TakeCurrentSimulationTimeProbe*, and the *TakeNumberOfResourceContainersTriggeredProbe*. It combines these measurements into a *CostOverTime* metric measurement.

5. PROOF-OF-CONCEPT EVALUATION

We evaluated our integration using the book shop example introduced in Sec. 2. Based on our book shop model, we simulated the *CostOverTime* metric for each resource container over 1/10 of a day (8,640 seconds). We set the periodic trigger for each container (i.e., their billing periods) to 1/10 of an hour (360 seconds). We chose to simulate only a tenth of a day and divided the billing periods of the shop example by 10 in order to keep simulation times and result

[1]Our integration is part of the current SimuLizar release.

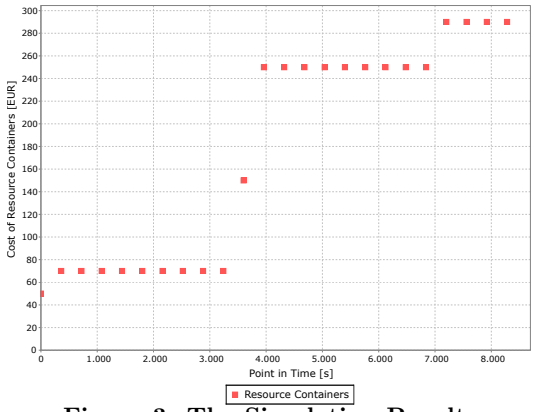

Figure 3: The Simulation Results

sizes adequate. Moreover, we specified a workload where arrival rates linearly increase over time such that we force the application to replicate containers.

Figure 3 shows the results of our simulation. The x-axis denotes simulation time and the y-axis the total operation cost for all resource containers in Euros. We see that the accrued operation costs are provided at the start of each billing period. We also see the points in time when the book shop application allocated further containers to cope with increased workload, along with increased costs. For example, around the 4,000 seconds mark, 8 additional containers were allocated, resulting in additional costs of 80€. These results correspond to our expected increase in operation costs, thus, successfully concluding our proof-of-concept evaluation.

6. RELATED WORK

The industry has adopted simple **cost calculators** (e.g., by Amazon [1] and ProfitBricks [2]). These calculators compare typical configurations (e.g., number of virtual machines) of different cloud computing environments in terms of operation costs. When the set of concrete configurations is known, such comparisons provide accurate cost estimates and can easily be conducted. However, software architects have to determine potential configurations without knowing whether these configurations provide suitable application capacity. Thus, such calculators involve the risk that software architects under- or over-estimate application needs and therefore misconfigure such cost calculators. Our cost-efficiency analysis instead depends on the concrete software architecture model, thus, enabling cost-efficiency analyses tied to the actual resource demands of the application.

Another category of related work are approaches that **predict future costs based on historic data**, e.g., the cloud analytics tools of RightScale [4]. As long as future workload and application behavior follow the same patterns as in previous situations, such approaches provide accurate cost predictions. However, such approaches need an already operating application and are prone to unexpected workloads (e.g., sudden spikes) and changes in application behavior (e.g., due to reimplementations and data flow that was never triggered before). In our approach, we can systematically elaborate varying workload situations and assess different variants of application behavior at design-time.

In the category of **design-time analyses**, several approaches build-up on Palladio [6], a design-time analysis approach focused on performance. SimuLizar [5] extends Palladio by models and analyses for self-adaptations, allowing software architects to assess performance when software applications adapt their use of computing resource over time. Martens et al. [8] extend Palladio by cost analyses but assuming that computing resource do not change over time. Our work builds up on these approaches by integrating Martens et al.'s cost analyses into SimuLizar. Therefore, our approach can accurately assess costs accrued for using varying computing resource over time.

7. CONCLUSIONS

In this paper, we integrate cost metrics into SimuLizar. As a result, SimuLizar now can produce cost-efficiency reports, e.g., reporting on the total operation costs accrued for using computing resources in a given time frame.

Software architects can use our integration for trade-off analyses, e.g., to balance traditional performance metrics like throughput against operation costs. Such analyses are now possible for dynamically changing workloads.

In future work, we plan to extend our current cost-efficiency analysis by further metrics that automatically trade costs off against other quality properties like elasticity, performance, and availability. We plan to use Palladio's Experiment Automation for such advanced analyses [3]. In addition, the overhead of performing the cost calculations is not addressed at the moment, which we plan to do in the future. The current cost model regards only resources as cost factors. We aim to expand this notion of cost by other factors, e.g., network bandwidth, which could be billed by a cloud provider.

8. REFERENCES

[1] Amazon monthly cloud calculator. `http://calculator.s3.amazonaws.com/index.html`. Accessed: 10/20/2015.

[2] ProfitBricks virtual data center calculator. `https://www.profitbricks.com/pricing#section=pricing-cloud-calculator`. Accessed: 10/20/2015.

[3] Quality Analysis Lab (QuAL): Software design description and developer guide version 0.2. `http://wiki.cloudscale-project.eu`. Accessed: 10/20/2015.

[4] RightScale cloud analytics. `http://www.rightscale.com/products-and-services/products/cloud-analytics`. Accessed: 10/20/2015.

[5] M. Becker, S. Becker, and J. Meyer. SimuLizar: Design-time modeling and performance analysis of self-adaptive systems. In *In SE'13 Proceedings*, volume 213 of *LNI*, pages 71–84. GI, 2013.

[6] S. Becker, H. Koziolek, and R. Reussner. The palladio component model for model-driven performance prediction. *J. Syst. Softw.*, 82(1):3–22, Jan. 2009.

[7] S. Lehrig, H. Eikerling, and S. Becker. Scalability, elasticity, and efficiency in cloud computing: A systematic literature review of definitions and metrics. In *QoSA '15 Proceedings*, pages 83–92. ACM, 2015.

[8] A. Martens, H. Koziolek, S. Becker, and R. Reussner. Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms. In *WOSP/SIPEW '10 Proceedings*, pages 105–116, New York, NY, USA, 2010. ACM.

[9] P. Mell and T. Grance. The NIST definition of cloud computing. *NIST Special Publication*, 145(6):7, 2011.