Cloud User-Centric Enhancements of the Simulator CloudSim to Improve Cloud Deployment Option Analysis

Florian Fittkau, Sören Frey, and Wilhelm Hasselbring

Software Engineering Group, Christian Albrechts University of Kiel, Germany

Abstract. Cloud environments can be simulated using the toolkit CloudSim. By employing concepts such as physical servers in datacenters, virtual machine allocation policies, or coarse-grained models of deployed software, it focuses on a cloud provider perspective. In contrast, a cloud user who wants to migrate complex systems to the cloud typically strives to find a cloud deployment option that is best suited for its sophisticated system architecture, is interested in determining the best trade-off between costs and performance, or wants to compare runtime reconfiguration plans, for instance. We present significant enhancements of CloudSim that allow to follow this cloud user perspective and enable the frictionless integration of fine-grained application models that, to a great extent, can be derived automatically from software systems. Our quantitative evaluation demonstrates the applicability and accuracy of our approach by comparing its simulation results with actual deployments that utilize the cloud environment Amazon EC2.

1 Introduction

The toolkit CloudSim [2] can simulate cloud environments. It focuses on concepts like CPU scheduling strategies, detailed physical host models, and virtual machine (VM) allocation policies. Hence, it takes the cloud provider perspective. However, for migrating an application into the cloud, a cloud user typically wants to find a cloud deployment option (CDO) that delivers the best trade-off between costs and performance, whereas many details of the underlying platform remain unknown. In the context of deploying software on a cloud platform, a CDO can be seen as a combination of decisions concerning the selection of a cloud provider, the deployment of components to a number of virtual machine instances, the virtual machine instances' configuration, and specific runtime adaptation strategies. The set of combinations of the given choices forms a huge design space that is infeasible to test manually [5]. Thus, simulating CDOs can significantly simplify reasoning about appropriate solutions for cloud users.

We developed the simulation tool CDOSim [3] that can simulate the costs, response times, and SLA violations of a CDO. For these purposes, we utilized and substantially extended the cloud simulator CloudSim by means of elasticity, price models, and remote calls between virtual machine instances. In this paper, we present our enhancements to CloudSim that facilitate a dedicated cloud user view. Our separation of these perspectives follows the definition of the cloud role model by Armbrust et al. [1]: A *cloud provider* offers the *cloud users* the resources in terms of the utility computing paradigm. We report on a case study that uses the public cloud environment Amazon EC2 and demonstrates the accuracy of our CloudSim enhancements.

The remainder of the paper is structured as follows. Section 2 overviews CDOSim. Our CloudSim enhancements are presented in Section 3. Afterwards, Section 4 evaluates the enhancements with the help of a case study, before the related work is described in Section 5. The final Section 6 draws the conclusions and outlines the future work.

2 The Cloud Deployment Option Simulator CDOSim

CDOSim builds on CloudSim [2]. It is a toolkit for the modeling and simulation of cloud environments. With CloudSim, the simulation of distributed environments and corresponding model entities, e.g., virtual machines, scheduling strategies, and data centers, can be conducted using a single computer. Network connections between data centers and data center brokers can also be simulated.

In contrast, our tool CDOSim enables the simulation of different cloud deployment options for software systems that have—often automatically—been reverse-engineered to Knowledge Discovery Metamodel (KDM)[1] code models. KDM is used for representing the architecture of the application under study. CDOSim integrates in our cloud migration framework CloudMIG [4] and is available online as a plug-in for the corresponding tool CloudMIG Xpress.[2] CloudMIG utilizes so called cloud profiles to model, for example, the provided resources, services, and pricing of a cloud environment. In the context of those cloud profiles, CDOSim can simulate the occurring costs, response times, and SLA violations of a CDO. Different VM scheduling strategies of the cloud providers are implicitly measured by our benchmark (for details see Fittkau [3]). CDOSim utilizes Structured Metrics Meta-Model (SMM)[3] models for describing workload profiles. In SMM, the measurement, the measure, and the observation timestamps of each call to the service are described. Furthermore, CDOSim can start or shutdown virtual machine instances based on the average CPU utilization of allocated virtual machine instances resulting from arbitrary workload patterns. Furthermore, the initial VM instance type and the number of instances that shall be run at the beginning of the simulation can be configured. To dynamically start and stop VM instances and to utilize other VM instance types according to varying workload intensities, CloudMIG Xpress provides so called runtime adaptation rules. These rules can be simulated by CDOSim too.

[1] `http://www.omg.org/spec/KDM/`, last accessed 2012-06-29
[2] `http://www.cloudmig.org/`, last accessed 2012-06-29
[3] `http://www.omg.org/spec/SMM/`, last accessed 2012-06-29

3 Cloud User-Centric Enhancements of CloudSim

The next Sections 3.1 to 3.7 describe our enhancements of CloudSim in detail.

3.1 CPU Utilization Model per Core

CloudSim provides a pure random-based CPU utilization model because the CPU utilization is often rather random from a cloud provider perspective. However, from the cloud user perspective we can approximate the CPU utilization because of additional knowledge concerning an application's structure in combination with a recorded workload profile. The CPU utilization is a major predictor indicator for the performance of a VM instance. For this purpose, we implemented a CPU utilization model that follows the conducted work for an application call.

3.2 Starting and Stopping Virtual Machine Instances on Demand

In CloudSim, the virtual machine instances cannot be comfortably started on demand at runtime. They have to be created before the simulation begins or when the simulation is stopped. Hence, there exists no convenient way to simulate automatic elasticity in CloudSim. The CloudSim authors provide a way to stop the simulation and then change the configuration. However, using this approach to enable elasticity would result in stopping the simulation, for example, each minute and testing if the configuration would have to be altered. This activity should be an internal function and as cloud users we should only need to define adaptation rules. We implemented this feature into CloudSim.

Adaptation rules are required for starting and terminating instances on the basis of occurring events or the exceeding of thresholds. An example for an adaptation rule is "start a new VM instance when for 60 seconds the average CPU utilization of allocated nodes stays above 70 %."

CloudSim effectively limits this amount because only a restricted quantity of hosts can be added upfront and each host has a limited capacity as well. We extended CloudSim such that with every virtual machine instance a new host, that fits the needs of the virtual machine instance, is added dynamically at runtime.

3.3 Delayed Cloudlet Creation

CloudSim requires all Cloudlets, which model an application calculation, to be started at the beginning, if we ignore the unapt method of stopping the simulation at a defined timestamp. With this behavior web applications cannot be modeled in a realistic way because all requests would start at the beginning of the simulation and in parallel. Hence, we enhanced CloudSim such that Cloudlets are extended by an attribute *delay*, which corresponds to the time when the Cloudlet should be sent for processing. Hence, we can now handle flexible and realistic usage profiles.

3.4 Delayed Start of Virtual Machines

In CloudSim, a creation of a virtual machine results in instant availability of the VM instance. Our conducted tests showed that there is an average delay of, for example, one minute on our private Eucalyptus cloud which is typically not negligible. Therefore, we implemented an event for the delayed creation of VMs. The former creation method is triggered by this new event handler.

3.5 Configurable Timeout for Cloudlets

In web applications, there is typically a configurable response timeout. After this timeout, an answer is useless because the client or server closed the connection. Most web servers would recognize when a user closes the connection by timeout and would terminate the corresponding task that calculates the answer. This results in savings of CPU time. Hence, we also implemented a timeout for calls to application logic. Every Cloudlet that is executing, paused, or waiting, can get canceled after a configurable timeout.

3.6 Enhanced Debt Model

The debt model in CloudSim is kept coarse-grained and in particular, it's implementation uses just a basic calculation mechanism. Modeling the current debt model of Amazon EC2, for instance, is not possible with this debt model. Hence, we implemented a debt model that follows the pricing model of CloudMIG Xpress and takes a time span for which the debts are calculated. For instance, for modeling the on demand VM instance debt model of Amazon EC2, every begun hour the price for the running VM is added to the debts. Furthermore, the debt model for bandwidth usage is modeled as a step function like done by Amazon EC2. For example, the first gigabyte of traffic is free of charge, above one gigabyte to 10,000 gigabytes, every gigabyte costs 0.12$ at the time of this writing.

3.7 Method Calls and Network Traffic between Virtual Machine Instances

In CloudSim, each Cloudlet runs on one virtual machine instance. It can be moved to other virtual machine instances but a Cloudlet, e.g., representing an object-oriented method, cannot "call" other Cloudlets.

We wanted to simulate the explicit calling of methods between different virtual machine instances and on the same instance. For example, a use case for this is the calling of web services on other virtual machine instances. For this purpose, we had to implement a new Cloudlet scheduler. For example, assume *method1* which should execute on *VM1* and should synchronously call *method2* on *VM2*. *Method1* is represented by *Cloudlet1*. Before *Cloudlet1* is executed, the scheduler searches in the source code of *method1* for methods that are called by *method1*. A call to *method2* is found and the *Index Service* is queried for the location where *method2* is running. The *Index Service* returns *VM2* and for *method2* the new *Cloudlet2* is created on *VM2*. Then, *Cloudlet1* pauses itself, meaning other Cloudlets can process on *VM1*. Assume *method2* conducts no method calls. Therefore, *Cloudlet2* processes and then wakes up *Cloudlet1* on finish. *Cloudlet1* can now process or call other methods.

4 Case Study

We conducted a case study to show that our CloudSim enhancements perform in a valid, realistic way. We utilized our developed tool CDOSim, which includes our CloudSim enhancements, to reproduce a real run we conducted on Amazon EC2. Further evaluations of CDOSim, which also show its scalability, can be found in Fittkau [3]. We utilize iBatis JPetStore 5.0[4] in the case study.

4.1 Methodology

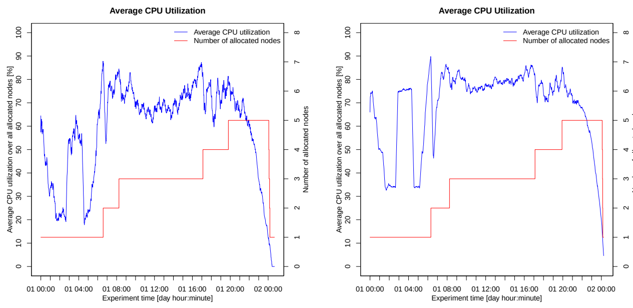
We compare the measured values with simulated values per minute. The values we compare are CPU utilization, instance count, costs, and response times. As a metric, we utilize the relative error for each of those aspects in percent values. For calculating the relative error at timestamp t, the simulated value is subtracted from the measured value and then divided by the measured value. The relative error (RE) for the whole run is calculated by summing up the relative error for each timestamp and then dividing the value by the number of timestamps. For details we refer to Fittkau [3]. All percent values will be truncated after the second decimal place. We feature four different REs. RE_{CPU} stands for the relative error of the CPU utilization. RE_{IC} is the relative error of the VM instance count. RE_{Costs} is the relative error of the costs output. RE_{RT} marks the relative error of the response times. Due to space limitations, we only provide the plots for the CPU utilization. For comparing whole runs, we introduce the overall relative error ($OverallRE$) which is the arithmetical mean of the four former described relative errors. The $OverallRE$ should remain below 30 % to have results that are sufficiently accurate [8].

4.2 C1: Case Study using Single Core Instances

Goal Our goal for this case study is to show that our CloudSim enhancements are valid by simulating a conducted run, that used single core instances.

Experimental Setting The workload intensity function that is used in the case study origins from a service provider for digital photos. It represents a day-night-cycle workload that's pattern can be considered typical for regional websites. It starts with a few requests at night and increases in the morning. Then, it peaks at about 3,500 requests per minute in hour 10 and slowly decreases to about 3,000 requests per minute at noon. Afterwards, there is a second peak in hour 20 with about 5,800 requests per minute which decreases until midnight.

[4] `http://sf.net/projects/ibatisjpetstore/`, last accessed 2012-06-29



(a) Measured CPU utilization (b) Simulated CPU utilization

Fig. 1. Average CPU utilization of allocated nodes

The simulation takes place on the basis of a workload from a run that was conducted on Amazon EC2 incorporating *m1.small* instances. The adaptation strategy is 90 % CPU utilization for starting a new instance and 10 % CPU utilization for terminating a running instance. The run starts with one instance, which will not be terminated.

Results Fig. 1 shows the average CPU utilization of the allocated nodes. In Fig. 1(a), the measured CPU utilization on Amazon EC2 and in Fig. 1(b) the simulated CPU utilization by CDOSim are presented. Over time, the instance count in the simulation and the conducted run is approximately equal. The CPU utilization is also roughly equal except from the beginning to hour 6. In this time period, the simulated CPU utilization differs by an offset of about 10 %.

The relative error for the CPU utilization is $RE_{CPU} = 30.64$ %. The average difference per minute is 12.04 % CPU utilization. The relative error of the instance count is $RE_{IC} = 1.32$ %. The overall difference of the instance minutes amounts to 28 instance minutes. The incurred costs account for 6.745$ for the Amazon EC2 run. The simulation costs result in 7.125$, which is $RE_{Costs} = 5.63$ %. The relative error for the response times is $RE_{RT} = 37.57$ %. The average difference per minute is 120.29 milliseconds. The overall relative error for this scenario amounts to $OverallRE = 18.79$ %.

Discussion of the Results The relative error for the CPU utilization is 30.64 % which we attribute mainly to the differences from hour 1 to hour 6 which most probably resulted from the performance variations of *m1.small* instances on Amazon EC2 [3]. The relative error of 1.32 % for the instance count shows that the number of instances that were utilized in the conducted run can be sufficiently well reproduced. The relative error of 5.63 % for the costs is also low and shows that the corresponding reproduction is sufficiently accurate. The relative error for the response times is 37.57 %. We attribute this rather high value to the high response times that were simulated in hour 20 [3]. The overall relative error of 18.79 % is below our 30 % threshold and thus, we conclude that the simulation sufficiently well reproduces the conducted run in total.

Threats to Validity The performance of the instances can differ with the location where the VM instances are spawned in a public cloud. The performance can also be influenced by the workload intensity which might have changed during the run on the executing host. We cannot control these factors and thus, they stay as a threat to validity.

5 Related Work

GroudSim is a tool for simulating clouds environments. It was developed by Ostermann et al. [10] and supports the simulation of clouds and grids. The equivalent to Cloudlets in CloudSim are GroudJobs in GroudSim. Failures of different components can be defined in GroudSim. They are then generated in a defined interval for a specific registered resource. In contrast to CloudSim, GroudSim is not under active development.

Another cloud simulator is MDCSim [7]. It is especially designed for in-depth analysis of multi-tier data centers and can estimate the throughput, response times, and power consumption. In contrast to CloudSim, its simulation is configured into three layers, namely a communication layer, kernel layer, and user-level layer for modeling the different aspects of a cloud.

GreenCloud [6], which is an extension to the network simulator Ns2, enables the simulation of energy-aware cloud computing data centers. It is designed for the simulation of detailed energy consumption of data center components like servers, switches, and links, and packet-level communication patterns. On the contrary, CDOSim focuses on the cloud user perspective which often has no knowledge about the internal components of a data center.

Nuñez et al. [9] developed the simulation platform iCanCloud for modeling and simulating cloud computing architectures. It bases on the SIMCAN simulation framework and can predict the trade-off between costs and performance of a particular application in a specific cloud environment and configuration. Existing software systems can only be modeled manually with iCanCloud, whereas CDOSim utilizes KDM models that can often be extracted automatically.

6 Conclusion and Future Work

A wide range of different cloud deployment options (CDOs) has to be assessed by a cloud user during a cloud migration. Basic CDOs are the selection of a cloud provider, suitable VM instance types, and runtime adaptation strategies, for instance. Due to the infeasibility of manually testing all CDOs, the best ratio between high performance and low costs can be found by utilizing simulators like CloudSim. CloudSim is a very useful toolkit for simulating cloud environments. It follows the cloud provider perspective but it lacks support for the cloud user view, which restrains the possibilities to simulate CDOs.

Therefore, this paper presented our enhancements to CloudSim that establish a cloud user perspective for simulating CDOs with our developed tool named CDOSim. We presented a case study that utilizes the public cloud provider Amazon EC2. It showed that the simulation results that were produced by incorporating our CloudSim enhancements are reasonably near to the conducted run concerning accruing costs and performance on Amazon EC2.

Most future work lies in further adaptations to CloudSim. For enabling efficient automatic CDO optimization support that requires plenty of simulations, CloudSim should be extended to support parallel simulations.

Bibliography

[1] Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the Clouds: A Berkeley View of Cloud Computing. Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley (Feb 2009)

[2] Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience 41, 23–50 (Jan 2011)

[3] Fittkau, F.: Simulating Cloud Deployment Options for Software Migration Support. Master's thesis, Software Engineering Group, University of Kiel, Kiel, Germany (March 2012)

[4] Frey, S., Hasselbring, W., Schnoor, B.: Automatic Conformance Checking for Migrating Software Systems to Cloud Infrastructures and Platforms. Journal of Software Maintenance and Evolution: Research and Practice (2012), doi: 10.1002/smr.582

[5] Grundy, J., Kaefer, G., Keong, J., Liu, A.: Guest Editors' Introduction: Software Engineering for the Cloud. IEEE Software 29, 26–29 (2012)

[6] Kliazovich, D., Bouvry, P., Khan, S.: GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. The Journal of Supercomputing pp. 1–21 (2010), doi: 10.1007/s11227-010-0504-1

[7] Lim, S.H., Sharma, B., Nam, G., Kim, E.K., Das, C.: MDCSim: A multi-tier data center simulation, platform. In: IEEE International Conference on Cluster Computing and Workshops 2009. pp. 1–9 (Aug 2009)

[8] Menasce, D.A., Almeida, V.A.F.: Capacity Planning for Web Services: Metrics, Models, and Methods. Prentice Hall International (Sep 2001)

[9] Nuñez, A., Vázquez-Poletti, J., Caminero, A., Carretero, J., Llorente, I.: Design of a new cloud computing simulation platform. In: Murgante, B., Gervasi, O., Iglesias, A., Taniar, D., Apduhan, B. (eds.) Computational Science and Its Applications - ICCSA 2011, Lecture Notes in Computer Science, vol. 6784, pp. 582–593. Springer Berlin / Heidelberg (2011)

[10] Ostermann, S., Plankensteiner, K., Prodan, R., Fahringer, T.: GroudSim: An Event-based Simulation Framework for Computational Grids and Clouds. In: CoreGRID/ERCIM Workshop on Grids, Clouds and P2P Computing. Springer (Aug 2010)