Open access • Proceedings Article • DOI:10.1109/CLOUD.2017.96

Resource Allocation in the Cloud: From Simulation to Experimental Validation

Pieter-Jan Maenhaut, Hendrik Moens, Bruno Volckaert, Veerle Ongenae ...+1 more authors

Institutions: Ghent University

Published on: 01 Jun 2017 - International Conference on Cloud Computing

Topics: Cloud computing, Cloud testing, Resource allocation, Resource management and Scalability

Resource Allocation in the Cloud: From Simulation to Experimental Validation

Pieter-Jan Maenhaut*†, Hendrik Moens†, Bruno Volckaert†, Veerle Ongenae* and Filip De Turck†
*Ghent University, Faculty of Engineering and Architecture, Dept. of Information Technology
Valentin Vaerwyckweg 1, 9000 Ghent, Belgium
†Ghent University – imec, IDLab, Dept. of Information Technology, iGent
Technologiepark-Zwijnaarde 15, 9052 Ghent, Belgium
Email: pieterjan.maenhaut@ugent.be

***Abstract*—With cloud computing, the efficient management of resources is of great importance as an increased utilization of the available resources can result in higher scalability and significant energy and cost reductions. Experimental validation of novel resource management strategies is costly and time consuming, and often requires in-depth knowledge of and control over the underlying cloud platform. As a result, many novel strategies are only evaluated by means of simulations, in which the whole cloud computing environment is modelled and simulated.**

Nonetheless, experimental validation should also be considered during the validation, as these types of experiments can often result in new insights or they can be used to fine-tune some specific parameters. In this paper we present a general approach for the experimental validation of cloud resource management strategies, together with the introduction of a cloud testbed adapter which was designed to facilitate the step from simulations towards experimental validation on physical cloud testbeds. We illustrate our solution by means of two case studies, focusing on two different types of testbeds. The adapter mainly acts as a dispatcher towards specific services of the evaluated cloud setup, and allows researchers to easily validate their ideas without having to dive deep into the complex details of the underlying cloud platform.

I. Introduction

Within the context of cloud computing, efficient management of available resources is of great importance as it can not only result in higher scalability, but also in significant energy and cost reductions. In recent years, a lot of research has been done regarding the efficient allocation of cloud resources [1], [2]. This is often done by consolidating the required virtual machines or containers on few physical hosts. Novel resource allocation strategies however are often only evaluated by means of simulations, for example by using CloudSim [3], a mature framework for the modelling and simulation of cloud computing environments.

Apart from these simulations, experimental evaluation using a cloud testbed should also be considered, as these types of experiments can often result in new insights, or they can be used to fine-tune the developed algorithms. Furthermore, simulations are not standardized, and the applicability of the simulation is depending on the design of a good data set which corresponds to real world usage, making validation challenging. Every simulation framework also has its limitations, for example when using CloudSim, custom extensions such as CloudSimSDN [4] are required for validating SDN-based strategies. Unfortunately, experiments are both expensive and time consuming, and require an in-depth knowledge of the underlying cloud infrastructure. Cloud resource allocation strategies also often aim at resource management on the physical hardware level, and with some cloud platforms this level of access is simply impossible. Failing of experiments, for example due to unforeseen hardware constraints or a faulty algorithm, should be avoided when using large testbeds, as access to these testbeds is often limited in time.

In this paper, we present an approach for the experimental validation of novel strategies and the design of a generic cloud testbed adapter. The adapter is designed to facilitate the step towards experimental evaluation, without the need for diving deep into the complex details of the available testbeds. We present the general architecture of the adapter, together with two proof of concepts based on different types of cloud testbeds. The remainder of this paper is structured as follows. In the next section we discuss related work within the field. In Section III we briefly describe the process for developing novel resource management strategies and illustrate the importance of the presented cloud testbed adapter in Section IV. In Section V we illustrate how the adapter can be implemented on top of two different testbeds, followed by a short discussion in Section VI. Finally, in Section VII we state our conclusions and discuss avenues for future research.

II. Related Work

In recent years, a lot of research has been done regarding the efficient management of resources in cloud environments, resulting in multiple novel resource allocation strategies [1], [2]. In [5] for example, the authors present a solution for virtual machine consolidation in the cloud to reduce energy consumption. The presented solution is evaluated by means of simulations using workload traces from two real-world publicly available workloads. According to the authors the approach is very effective compared to the state of art, but they however do not provide experimental results using real hardware. In [6] several algorithms for virtual machine packing in the cloud are introduced. Although the authors note that the evaluation is based on an extensive set of experiments, the evaluation results are also solely based on simulations. CoolCloud [7] is a dynamic virtual machine

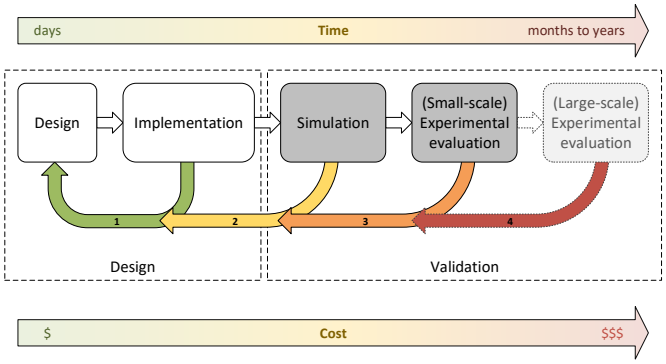

Fig. 1. General workflow for the design, implementation and validation of cloud resource allocation strategies.

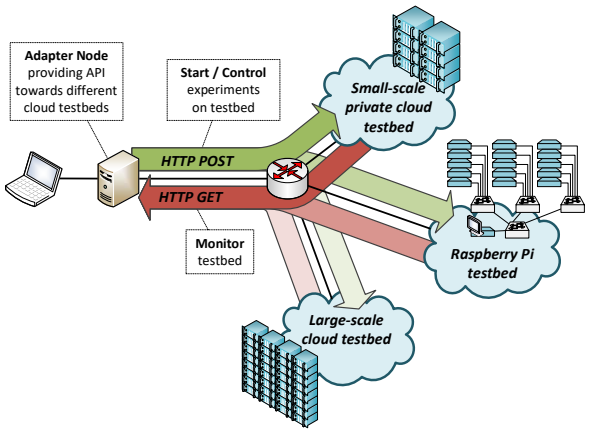

Fig. 2. Architecture of the cloud testbed adapter, providing easy access towards the different types of cloud testbeds.

placement framework for data centres. The framework also focuses on energy savings by consolidating virtual machines on physical servers. The authors evaluate the framework using both simulations and small-scale experiments on a VMWare-based testbed consisting of four physical servers.

As can be seen, most novel strategies are evaluated using only simulations. The adapter presented in this paper can facilitate the step towards experimental validation by providing an easy-to-use interface towards different types of experimental testbeds, from small-scale private clusters to larger research cloud environments. Within our research group, we have also been actively working on the design of new resource allocation strategies. Our research ranges from network-aware resource allocation algorithms in the cloud [8] to the efficient management of storage resources [9], and our evaluations are also often only based on simulations, for example by using the custom simulator tool presented in [10]. The testbed adapter presented in this paper will be used for experimental validation of these strategies on physical hardware.

III. Validation Process

Figure 1 summarizes the general steps for the design, implementation and validation of cloud resource allocation strategies. Initially, a new resource allocation strategy is designed and implemented. This can be an iterative process, as during the implementation some new constraints may be introduced, requiring modifications to the original design (arrow 1 in the figure). Once the implementation is finished, the strategy should be validated by mean of simulations, experimental evaluation on a cloud testbed, or ideally a combination of both. Simulations are often a good start, as these are less costly and less time consuming than experimental evaluations. They can vary from simple unit tests or batch scripts to full simulations of a cloud environment, for example by using CloudSim [3]. During the simulations, new optimizations can be discovered, or unforeseen limitations, again requiring changes to the implementation or design (arrow 2).

Experimental evaluation using real hardware should also be considered as these experiments often result in new insights (arrow 3). The evaluated setup could for example introduce additional hardware constraints which were not taken into account during the design or the experimental results could be used to more accurately tune the configurable parameters. Experiments on physical hardware however are both costly and time-consuming. This is especially true for the design and fine-tuning of new resource allocation strategies, as these often require multiple incremental iterations of experiments using multiple cloud instances. When the executed experiments fail during the execution, for example due to hardware constraints or a faulty algorithm, these experiments can become very costly. Therefore, experiments on relative small-scale testbeds, such as the evaluation setup described in [7], are initially preferred before doing large-scale experiments.

Academic emulation environments, e.g. the iLab.t Virtual Wall [11], are developed in order to support experimentation in a wide variety of research domains and with increased realism compared to simulations. Although these environments allow for large-scale system validation and offer valuable tool sets for experimentation, they have limited infrastructure resource availability and considerable software and hardware maintenance costs. Typically, such testbeds are used for large and mature validation tests and are less suited for small, repetitive tests with highly frequent updates (arrow 4). Furthermore, such environments are rack-mounted and therefore not suitable for off-premise demonstration purposes.

The adapter presented in the remainder of this paper is designed to facilitate the step towards experimental evaluation. By using the adapter, the developed algorithms can be easily plugged into small-scale testbeds, e.g. a Raspberry Pi cluster or a couple of Linux based VMs, and larger environments such as a private OpenStack environment, without having to dive deep into the complex details of these advanced cloud platforms.

IV. Cloud Testbed Adapter

The cloud testbed adapter is designed as an easy-to-use REST API for validation of cloud resource management

TABLE I
OVERVIEW OF MAIN API METHODS

Method	Path	Description
GET	/info	Overview of configured clusters
GET	/{c}/ping	Check if selected cluster is online
	/{c}/info	General information about cluster
GET	/{c}/{n}/ping	Check if selected node is online
	/{c}/{n}/all	Returns all resource utilization
	/{c}/{n}/usage/cpu	Current CPU utilization
	/{c}/{n}/usage/memory	Current RAM utilization
	/{c}/{n}/usage/disk	Current disk utilization
	/{c}/{n}/usage/network	Current network utilization
POST	/{c}/add	Provision additional node
	/{c}/start	Start distributed task on cluster
	/{c}/stop	Stop task
	/{c}/restart	Restart task
POST	/{c}/{n}/remove	Deprovision selected node
	/{c}/{n}/start	Start task on individual node
	/{c}/{n}/stop	Stop task
	/{c}/{n}/restart	Restart task

strategies, and supports 2 types of operations as illustrated in Figure 2. GET requests are used to retrieve information about the testbed environment, for example the current resource usage or the number of instances currently active. POST requests on the other hand are used to start and/or control tasks on the testbed. The API can be used to either get or send information from/towards the testbed as a whole (e.g. to provision an additional instance or to get the current number of instances), or from/towards an individual node (e.g. to retrieve the current CPU usage of the selected node).

Table I summarizes the available methods of the REST API. In the given path, {c} should be replaced by the cluster identifier, whereas {n} corresponds to a node identifier. Each cluster, and every node inside the cluster should have an unique identifier. All GET requests return a JSON object containing the requested information. Before retrieving information about the cluster or individual nodes, the `ping` method should be used first to determine if the cluster or node is online. This method is very lightweight and only returns the value 1 if the node is accessible. The POST requests are used to start/stop/restart experiments on the selected cluster or node and to (de-)provision additional nodes. For this to work, a simple control script (e.g. a bash or python script) should be deployed onto the individual nodes implementing the different operations. Details about the task are added to the body of the POST request, which in turn are passed as parameters to the control script on the node.

V. PROOF OF CONCEPT

As a proof of concept, we are implementing the cloud testbed adapter on two different testbeds. The first testbed is a Raspberry Pi cluster, consisting of 30 Raspberry Pi 3 nodes grouped into small sub clusters. The second testbed is a medium-scale private OpenStack cloud, running on 10 physical blade servers.

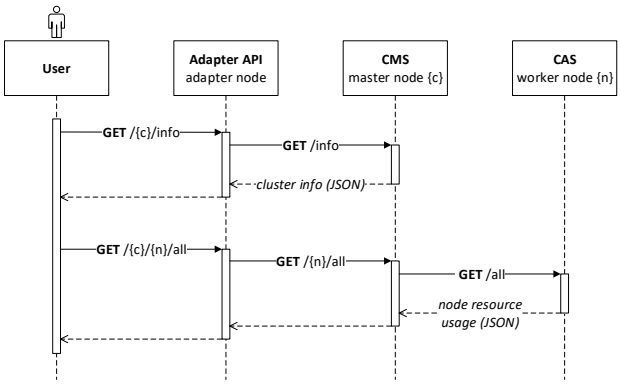

Fig. 3. Interaction between the cloud testbed adapter and the Raspberry Pi cluster for two sample API calls.

A. Raspberry Pi Cluster

The Raspberry Pi cluster testbed consists of multiple Raspberry Pi 3 worker nodes, aggregated in small clusters of around 5 nodes. Every worker node in the cluster is interconnected, and there is one master node managing the whole testbed. This master node can either be a Raspberry Pi or any device such as a laptop or desktop computer running a Linux distribution and the required management services (such as routing, DHCP, DNS, TFTP and/or NFS). We have developed two custom Node.js services for the Raspberry Pi Cluster. The Cluster Management Service (CMS) is deployed on the master node and is used to manage the whole testbed, for example to get the status of individual worker nodes or to (de-)provision additional nodes. The Cluster Agent Service (CAS) is deployed on all worker nodes and communicates directly with the OS of the worker node to retrieve the current resource usage and to control tasks. Both services provide a REST API similar to the main API presented in Table I, and are developed using a combination of Node.js, Python, HTML, JavaScript and Bash scripting. The services can be easily deployed onto any Linux-based system.

Figure 3 illustrates the interaction between the different components for 2 example API calls. Information about the cluster can be retrieved from the CMS, whereas information about an individual node is retrieved from the CAS of this node. For this Proof of Concept, the cloud testbed adapter only dispatches requests towards the correspondent services of the Raspberry Pi cluster.

B. OpenStack Private Cloud

Our second testbed is a medium-scale OpenStack private cloud, deployed on top of 10 physical blade servers. The OpenStack platform consists of interrelated components controlling the different aspects of the cloud, such as OpenStack Compute (Nova) for managing the computational resources and OpenStack Networking (Neutron) managing the internal network. Most components already provide a REST API, which can be used for the implementation of the GET operations listed in Table I. For these operations, the adapter translates incoming

requests to the relevant API's of the different components, aggregates the results and sends them back to the client.

Some operations, e.g. most POST operations used for controlling tasks on the worker nodes, can not be implemented by using only the API of the OpenStack components. In order to support those operations, a custom version of the CAS introduced in the previous case study can be deployed on the instantiated OpenStack nodes. The service can be preconfigured in the main image which is used to deploy new virtual machines.

VI. Discussion

During the design of the adapter, we wanted to keep the interface compact and intuitive, but providing sufficient operations for a broad range of experiments. We illustrate the usage of the adapter using a simple sample strategy. In this example, we focus on the execution of CPU-intensive tasks on a cloud testbed. When we want to allocate resources for a new task, we first retrieve general information about the cluster (e.g. an overview of provisioned nodes) using the **GET** `/{c}/info` method call. Once we have retrieved the list of provisioned nodes for our cluster, we can get the current CPU utilization for each node using **GET** `/{c}/{n}/usage/cpu`. We can now either start the task on one of the existing nodes (**POST** `/{c}/{n}/start`) or provision an additional node and assign the task to this new node (**POST** `/{c}/add` followed by **POST** `/{c}/{n}/start`). During the execution, we can monitor the node using **GET** `/{c}/{n}/usage/cpu`.

Although in this paper we implemented the adapter on top of two specific types of cloud testbeds, the developed services can be easily modified to support other types of testbeds. The services developed for the Raspberry Pi cluster for example can be executed on any Linux-based operating system with Node.js, python and bash installed. Furthermore, the API of the adapter node can be further extended, for example to monitor the current GPU utilization of the nodes, or to include other types of hardware resources. The adapter can also be used to manage multiple testbeds at once, as long as an implementation is provided for every distinct testbed type, which can be very useful for the validation of strategies focusing on resource management in heterogeneous cloud environments.

The source code for the adapter will be made available to the general public through GitHub[1][2], together with the implementation for both testbeds described in this paper. We encourage fellow researchers within the field to try out and customize or extend the code for their own research projects.

VII. Conclusions and Future Work

With cloud computing, efficient resource management is of great importance as it can result in higher scalability and significant energy and cost reductions. Although simulations are a great tool for the development of new resource allocation strategies, experimental validation using physical hardware should also be considered as these types of experiments will often result in new insights. Experimental validation however is both costly and time-consuming, especially during the initial design phase.

In this paper, we presented a general workflow for the design, implementation and validation of cloud resource allocation strategies, together with the introduction of the cloud testbed adapter. The adapter was designed to facilitate the step towards experimental validation, by providing a convenient and compact REST API towards the user. As a proof of concept, we implemented the adapter on top of two different types of cloud testbeds. The implementation can be easily extended or customized, for example to include other types of hardware resources or to support other testbed types. In the near future we will use the adapter for the experimental validation of the cloud resource allocation strategies developed within our research group.

[1] https://github.com/IBCNServices/RPiaaS

[2] https://github.com/IBCNServices/cloud-adapter

References

[1] Z. Lu, S. Takashige, Y. Sugita, T. Morimura, and Y. Kudo, "An analysis and comparison of cloud data center energy-efficient resource management technology," *International Journal of Services Computing (IJSC)*, vol. 2, no. 4, pp. 32 – 51, 2014.

[2] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management*, vol. 23, no. 3, pp. 567 – 619, 2015.

[3] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23– 50, 2011.

[4] J. Son, A. V. Dastjerdi, R. N. Calheiros, X. Ji, Y. Yoon, and R. Buyya, "Cloudsimsdn: Modeling and simulation of software-defined cloud data centers," in *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2015, pp. 475–484.

[5] N. T. Hieu, M. D. Francesco, and A. Yl-Jski, "Virtual machine consolidation with usage prediction for energy-efficient cloud data centers," in *2015 IEEE 8th International Conference on Cloud Computing*, June 2015, pp. 750 – 757.

[6] S. Rampersaud and D. Grosu, "Sharing-aware online algorithms for virtual machine packing in cloud environments," in *2015 IEEE 8th International Conference on Cloud Computing*, June 2015, pp. 718 – 725.

[7] Z. Zhang, C.-C. Hsu, and M. Chang, "Coolcloud: A practical dynamic virtual machine placement framework for energy aware data centers," in *2015 IEEE 8th International Conference on Cloud Computing*, June 2015, pp. 758 – 765.

[8] H. Moens, B. Hanssens, B. Dhoedt, and F. D. Turck, "Hierarchical network-aware placement of service oriented applications in clouds," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–8.

[9] P.-J. Maenhaut, H. Moens, B. Volckaert, V. Ongenae, and F. D. Turck, "Design of a hierarchical software-defined storage system for data-intensive multi-tenant cloud applications," in *2015 11th International Conference on Network and Service Management (CNSM)*, November 2015, pp. 22–28.

[10] P.-J. Maenhaut, H. Moens, B. Volckaert, V. Ongenae, and F. D. Turck, "A simulation tool for evaluating the constraint-based allocation of storage resources for multi-tenant cloud applications," in *2016 IEEE/IFIP Network Operations and Management Symposium (NOMS 2016)*, Istanbul, Turkey, April 2016, pp. 1017–1018.

[11] iMinds iLab.t Virtual Wall. [Online]. Available: http://ilabt.iminds.be/iminds-virtualwall-overview