An educational toolkit for teaching cloud computing

Cosimo Anglano
University of Piemonte Orientale
Italy
cosimo.anglano@uniupo.it

Massimo Canonico
University of Piemonte Orientale
Italy
massimo.canonico@uniupo.it

Marco Guazzone
University of Piemonte Orientale
Italy
marco.guazzone@uniupo.it

ABSTRACT

In an educational context, experimenting with a real cloud computing platform is very important to let students understand the core concepts, methodologies and technologies of cloud computing. However, API heterogeneity of cloud providers complicates the experimentation by forcing students to focus on the use of different APIs, and by hindering the jointly use of different platforms. In this paper, we present EasyCloud, a toolkit enabling the easy and effective use of different cloud platforms. In particular, we describe its features, architecture, scalability, and use in our cloud computing courses, as well as the pedagogical insights we learnt over the years.

CCS CONCEPTS

• **Applied computing → Education**.

KEYWORDS

Cloud computing, Educational resources, Multi-cloud, Toolkit

1 INTRODUCTION

Cloud computing [27] has become an essential technology in practically every application domain where computing activities need to be carried out. Teaching cloud computing has consequently become mandatory nowadays not only in Computer Science and Computer Engineering curricula [1, 2] (that focus on how cloud computing technologies may be used to build platforms and applications), but also in other curricula where cloud computing technologies provide the necessary substrate for carrying out various type of computational tasks (e.g., chemistry, biology, bioinformatics, etc.).

In response to this need, suitable training courses [9, 20, 29] have been devised in order to properly take into account both the background of students (i.e., no/basic/intermediate/advanced computer science skills) and the specific training requirements of the course (e.g., focus on core technology/application development/platform usage/etc.) as dictated by the specific discipline under study.

Practical experimentation with a real cloud computing platform is a must in these courses to enable students to master both the core concepts, technologies, applications, and usage of cloud computing. This, in turn, requires the possibility of configuring, deploying, and using such a platform. The high capital and operational expenses needed to acquire and maintain a cloud computing infrastructure pushes educational institutions towards the use of the public computing platforms available today on the market (e.g., *Amazon Web Services* (AWS), *Google Cloud Platform* (GCP), *Microsoft Azure*, and *IBM Cloud*).

Usually, the services provided by these platforms are accessed by using the corresponding interfaces (which are either web-based or APIs). However, these interfaces differ from one cloud platform to another, so choosing one of them unavoidably yields to a form of lock-in [32], where students have to learn the specific interface of the selected platform and use it for their practical activities (so, most likely, they will be able to perform those activities only with the chosen platform), and instructors will not be able to reuse existing exercises developed for a different platform.

Furthermore, besides the lock-in, the above differences makes it very hard to assemble computing infrastructures that draw resources from different cloud platforms, a solution that could be appropriate when the resources accessible from individual providers are not enough for the specific teaching needs (e.g., when the number of students exceeds those that can be accommodated with resources taken from a single provider) or when there is the need to use resources provided by different cloud platforms at the same time (such as using accelerator-optimized resources, including TPUs and GPUs, not available or too expensive on a specific cloud platform).

To address the above issues, it is necessary to provide cloud toolkits that enable:

(1) uniform access to the various cloud platforms regardless of the differences in their access and usage primitives;
(2) simultaneous use of resources drawn from different cloud platforms;
(3) both simplified use of the above resources (for students that have limited computer science skill), and advanced use through a unified interface to interact with the various cloud platforms (for more skilled students);
(4) efficient use of both individual and composite cloud platforms, enabling to practice with large scale system and/or application configurations, so that realistic activities can be carried out.

To provide an answer to the above needs, we have developed, tested, and used in the classroom *EasyCloud* [8], a toolkit providing functionalities to simultaneously interact with different cloud platforms in a simplified but efficient, effective, and platform-agnostic way thanks to its unified API, easy user interface and its internal automatic dynamic management mechanisms.

From the perspective of students, EasyCloud exposes:

- the typical mechanisms to access the services provided by typical cloud platforms (such as, start and stop virtual machines, manage network and storage, etc.), so that students can experiment with the basic concepts pertaining to cloud computing;
- a set of dynamic management functionalities (i.e. a monitoring system that collects statistics about the performance and the health of the infrastructure, and a rule system that uses the above statistics to undertake optimization or corrective actions) that enables them to concretely apply the methodologies for the correct dimensioning and configuration of cloud computing infrastructures to effectively run applications on them.

From the perspective of instructors, the dynamic management functionalities of EasyCloud provide instead suitable mechanisms to ensure the business continuity, or the fulfillment of desired levels of performance and/or availability of the cloud infrastructure assembled and configured for the experimentation needs of the classroom.

Compared to existing alternatives (e.g.,[12, 22, 33, 35], EasyCloud is the only toolkit able to provide both basic management functionality and advanced functionality for monitoring *Virtual Machines* (VMs) and analyzing their performance (such as, preventing failures and automatically scaling in and out the computational resources provided).

EasyCloud is written in Python to ensure portability and fast development, and is released as open source [11] to foster research and provide reproducibility of results.

In this paper, after providing details on the capabilities of EasyCloud (Section 2), we show how we use it in our cloud computing courses as well as the various pedagogical insights we learnt over the years (Section 3), and we present an analysis of the degree of satisfaction of our students in using EasyCloud (Section 4). We then highlight how we designed EasyCloud (Section 5) and present how this design makes it scalable to large classrooms (Section 6). Finally, in Section 7, we conclude the paper and propose some future works.

2 MAIN CHARACTERISTICS

This section gives on overview of the main and most valuable characteristics of EasyCloud in terms of multi-cloud support, VM, storage and IP address management, monitoring and policy services.

2.1 Multi-cloud support

The main characteristic of EasyCloud is multi-cloud support, i.e. its ability to interface, at the same time, with multiple different cloud platforms. At the moment of this writing, EasyCloud supports *OpenStack*, *Amazon Web Services* (AWS), *Google Cloud Platform* (GCP), and *Chameleon Cloud* [23]. As discussed in [8], support to other platforms may be easily added thanks to the modularity of its architecture (see Section 5).

To interface with a given cloud platform, the user has to fill a configuration file specific for that platform, whose template – where most parameters required to access that platform through its specific API are set with suitable default values – is provided by EasyCloud. The bare minimum required to the user is to fill the configuration file with his/her credentials for the specific cloud platform. For example, for the AWS platforms, the only mandatory fields to set in the corresponding configuration file are *ec2_access_key_id* (corresponding to the user account on AWS) and *ec2_secret_access_key* (corresponding to the user access key on AWS). However, other optional fields are provided by configuration files, allowing the user to exploit more advanced features of the specific cloud platform. For example, by setting the *ec2_default_region* parameter in the AWS configuration file, it is possible to fix a specific region where the VMs will be launched; also, by setting parameter *freetier_only* to *true*, only the VMs included in the free tier will be used. [1]

2.2 Virtual machine management

EasyCloud allows to easily start/stop/reboot a VM. For example, if a user wants to start a VM on a given cloud platform, EasyCloud automatically retrieves – for the chosen platform – the list of available images and of the corresponding parameters, namely the *flavors* (i.e., the compute, memory and storage capacity of the VMs), the *security groups* (i.e., a collection of network access rules that are used to limit the types of traffic that have access to VMs) and the *key pairs* (i.e., the public keys to be used for accessing to created VMs). From these lists, the user has just to select the options which are best for his/her purposes, and the number *n* of VM to start, and EasyCloud will automatically start *n* VMs with the characteristics desired.

2.3 IP address management

One of the key aspects of any cloud platform is IP address management. As a matter of fact, any running VM has to be associated with an IP address, otherwise it (and thus the service it provides) would be unreachable. Each cloud platform has a different API for IP address management. EasyCloud

[1]A free tier is a plan that enables to access some of the services provided by a cloud platform without being charged if used within specific usage limits.

hides the differences among them by providing users with just three options: (i) show the list of the available IP addresses, (ii) assign an IP address to a specific VM and, finally, (iii) detach the IP address associated to a specific VM when it is no longer necessary.

2.4 Storage management

Every cloud platform provides permanent storage abstractions named *volumes*, whereby each volume is a detachable block storage device, conceptually similar to an external hard drive that can be created/deleted and attached to/detached from a VM. Each cloud platform provides its own volume API, that typically is quite different from the API of other platforms. EasyCloud hides these differences by providing a storage management interface that is independent from the various cloud platforms.

2.5 Resource monitoring

As for the other services discussed above, each cloud platform provides its own specific resource monitoring services, e.g. AWS provides *CloudWatch*, while OpenStack provides *Ceilometer*. To hide differences in the way monitoring services are accessible through the various cloud APIs, EasyCloud provides a uniform interface to them, allowing users to activate/deactivate monitoring on the various platforms (s)he is using. EasyCloud provides also *Measures Sink* components, which collect the metric data from the monitoring services and dispatch them to *data sinks* (more details on these components will be provided in Section 5). Metric data collected in this way can be used for various purposes, including the computation of usage statistics and the activation of specific resource management policies (see below). Currently, EasyCloud provides connectors for the following measures sinks: *CSV files*, *Apache ActiveMQ* [31], *Apache Cassandra* [18], *Apache Kafka* [24], *MongoDB* [14], *Prometheus* [15], *RabbitMQ* [34], and *Redis* [17].

2.6 Resource management policies

EasyCloud allows users to define dynamic and automatic resource management policies that are triggered by the data collected by the monitoring services. In particular, various policies – that can be used for purposes like energy awareness, resource upscaling/downscaling, and fault tolerance – may be easily created with EasyCloud by using its *Rule Engine* (see Sec. 5.2). For example, EasyCloud allows to define policies like "automatically switch off a VM if its CPU usage is lower than 10% (the threshold)".

3 EASYCLOUD IN THE CLASSROOM

In this section, we explain how we use EasyCloud with our students. In particular, we discuss the exercises (Sec. 3.1) we propose to the students and some of the lessons we learnt (Sec. 3.2) over the last 7 years using our tool.

3.1 Exercises proposed

In our student's lab, the students are free to use one single PC for person or to group themselves in teams of two people. The workload provided by the exercises we proposes can easy fit both scenarios. Students not able to attend in presence the lessons can find all educational materials (that is, slides, video-lessons, tutorials and exercises) by visiting the *Teaching Cloud Computing* [16] website. This a public website where we add and update all our materials. Besides this public website, the course has a specific/private website (built over the *Moodle* [19] learning platform) with access limited to students of our university where they can interact directly with the lecturers and other students by exploiting the chat and forum resources. The first two lessons are based on the theory of cloud computing (i.e., definition, service models and deployment modes) then a lesson is dedicated to create the SSH key pairs and the credentials for all the cloud platforms used during the course: Google Cloud Platform, Amazon Web Services and OpenStack. Starting from the fourth lesson, the students can easily play with all the three cloud platforms by exploiting the EasyCloud toolkit. In particular, the exercises proposed to the students are:

- **Start a webserver:** in this exercise the students have to create a Virtual Machine (VM) inside one of the cloud platform by using EasyCloud and to install a Linux Apache MySql PHP (LAMP) stack in order to run a webserver and a Content Management System (CMS) (in our lessons we propose both Joomla and WordPress as CMS).
- **Play with volume**: any cloud platform provides a storage service to save data from a running VM and to share the data with other VMs. In particular, in this exercise, we propose to the students to create a volume, attach it to a VM, modify it and then attach the volume modified to another VM. In this way, the data generated by the first VM can be used by the second one.
- **Load balancing:** in this exercise the student has to use EasyCloud to create at least 3 VMs running a webserver (they can use the same VM from the first exercise) and implement a load balancer able to split the workload among the various server. The student can increase/decrease the workload by using a work generator tool (we suggest the *stress app* [26]) and also switch-off/switch-on VMs during the exercise in order to check if the load balancer is working properly.

- **Performance comparison:** thanks to EasyCloud, the students can easily reproduce the same exercises mentioned before on different cloud platforms. For instance, we ask to the students to start several Web Servers on different cloud platforms, and compute how long each platform takes to start all servers. The best platform is the one able to provide all web server up and running in the shortest possible amount of time.

Once the students have confidence with EasyCloud and the cloud platforms, we propose them to create new exercises and provide step-by-step tutorials to carry out the exercises. The best tutorials are published in the "Tutorial" session of the Teaching Cloud Computing website.

3.2 Lessons learnt

In this section, we describe some of the lessons we learnt over the last 7 years of teaching cloud computing by exploiting our toolkit.

- **Interacting with a cloud platform directly through its API and/or its command-line interface may be challenging:** besides the web console provided by any cloud platform, when a student needs to interact with the VM to configure advanced settings or to install a specific software, it is necessary to access it by the SSH protocol. This generated several problems for our students, since most of them had very minimal experience with command-line interfaces, terminal consoles (like *Xterm*) and text editors for the terminal (like *Vim* or *GNU nano*). EasyCloud enabled students to overcome these difficulties by providing them with the possibility of performing the same task using both the simplified text-based interface and its API, and by requesting them to carry out exercises using first the text-based interface, then the API, and finally to compare the results.
- **Cloud unpredictability poses unnecessary challenges:** one of the informal definitions of cloud computing says "Cloud computing simply increases the number of things that can go wrong. And go wrong they do." [21] and from our experience this is true. The same exercise running on the same platform can fail for one student and can succeed for another student. We use public cloud computing and one of the various components of the cloud platform can have a temporary problem that can affect the whole exercise. As a matter of fact, sometimes it is just a matter of time. The cloud console could show that a service is "up and running" but it is actually still booting. For example, concerning the exercise related to the load balancer previously mentioned, we note that the students have to wait at least 2 minutes after the load balancer process is shown as "running". From our experience we note that if they start to use the load balancer immediately after it is shown as "running", the cloud console starts to generate some generic error message difficult to diagnose even if nothing is wrong with their configurations. EasyCloud hides, to a reasonable extent, some sources of unpredictability, such as the need to wait until services complete their startup, thus simplifying the tasks of the instructor and of the students.

4 STUDENT EXPERIENCE EVALUATION

In this section, we present the results of a study we carried out among our students to evaluate their degree of satisfaction in using EasyCloud.

We have been teaching distributed computing systems since 2000 and in the last ten years we have introduced cloud computing as one of the main topics of our courses. Initially, our courses were tailored only to graduate students with either a computer science or a computer engineering background. [2] Subsequently, given the widespread use of cloud computing not only in the computing-related fields but also in many other scientific fields, we then started to teach distributed computing system courses also to non-computer science students like biologists, economists, and chemists, just to name a few. Currently, we teach distributed computing systems to plenty of students with different backgrounds (e.g., computer science, biology, economy and chemistry, just to mention a few) and different levels of education (ranging from undergraduates to PhD students), by covering cloud computing and related topics at different levels of detail [9].

Every course we teach includes a series of hands-on sessions (similar to the one described in Section 3) where students interact with different cloud platforms (either via a textual interface or through an API) by using both EasyCloud and also the native interfaces provided by each cloud platform. As also confirmed by the feedback from our students (see below), for all classes, EasyCloud has been a valid tool to simplify the first approach to cloud computing platforms for any kind of student. For most of the non-computer science students, EasyCloud is just enough to provide them the features necessary to run their applications in to the cloud, while, for computer science students, EasyCloud is good to quickly test applications since it provides an easy and convenient interface to start experiments by exploiting concurrently various cloud platforms.

To evaluate the effectiveness and usefulness of EasyCloud from an educational point of view, at the end of a course we ask our students to provide us a feedback about their

[2]Unless otherwise stated, in the rest of this paper, we use the terms "computer science" and "computer engineering" interchangeably.

experience with both EasyCloud and the native interfaces of cloud platforms they used during the hands-on sessions. Specifically, our survey consists of the following four close-ended questions posed as statements:

- **Q1**: "The EasyCloud textual interface simplified the management of VM instances in a multi-cloud environment with respect to using the native textual interfaces of the various cloud platforms."
- **Q2**: "The EasyCloud textual interface is useful to manage VMs in a multi-cloud environment also outside the context of this course."
- **Q3**: "The EasyCloud API simplified the development of applications in a multi-cloud environment with respect to using the native APIs provided by the various cloud platforms."
- **Q4**: "The EasyCloud API is useful to develop applications in a multi-cloud environment also outside the context of this course."

Each question uses a 5-point *Likert scale* [25], where students must answer by choosing one of the following options: "Strongly disagree", "Disagree", "Neutral" (i.e., neither agree nor disagree), "Agree", and "Strongly agree." The same survey also includes an optional open-ended question where students can put free comments to complement their answers to the above four questions.

In the rest of this section, we present the results of the above survey we conducted for three different distributed computing systems courses we taught in the 2019/2020 academic year, hereafter denoted as *CS basic*, *CS advanced* and *non-CS*, targeted to computer science undergraduates, computer science graduate students, and non-computer science graduate students, respectively. All students who enrolled for the three courses completed the survey. Specifically, 81 students enrolled for *CS basic*, 26 students enrolled for *CS advanced*, and 63 students enrolled for *non-CS*.

Figure 1 shows the results of our survey as four diverging stacked bar charts [30], one for each close-ended question. Each chart has three horizontal bars (one for each course), and for each bar the percentages of students who agree with the associated question are shown to the right of the zero line (in variations of blue), while the percentages who disagree are shown to the left (and colored in variations of red); also, the percentages of students who neither agree nor disagree are split down the middle and are shown in grey color.

According to the answers provided to question "Q1", all students agree that the textual interface of EasyCloud simplified the management of VMs hosted by multiple cloud platforms with respect to the various textual interfaces specific to each cloud platform. Also, the percentage of students that strongly agree decreases with the increase of the computer science skills of our students; this is somewhat expected,

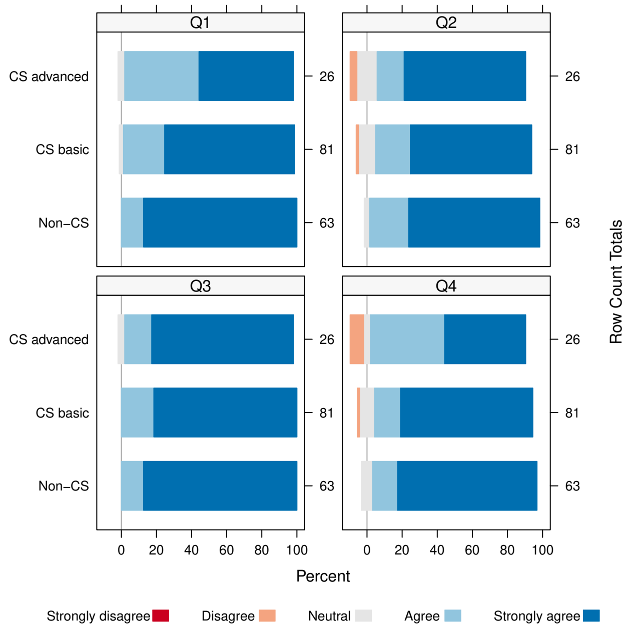


Figure 1: Results of the survey related to the *CS advanced*, *CS basic* and *non-CS* courses offered for the 2019/2020 academic year. There is a separate diverging stacked bar chart for each close-ended question of the survey (i.e., "Q1", "Q2", "Q3", "Q4") and, for each chart, there is a separate horizontal bar for each course (i.e., *CS advanced*, *CS basic* and *non-CS*). Each bar of a diverging stacked bar chart shows the degree of agreement to the right of the zero line, the degree of disagreement to the left, and the neutral opinion (i.e., neither agree nor disagree) in the middle.

considering the greater experience of computer science students in using several commands during their daily practical activity.

Similarly, the answers to question "Q2" show that the majority of students would like to use the textual interface of EasyCloud (rather than the native one provided by each cloud platforms) also in practical activities outside our course. However, we also note that some students (i.e., 15.39% of *CS advanced* and 11.11% of *CS basic*) are indifferent or think that they will not use the textual interface of EasyCloud in their future projects. By analyzing the comments provided in the open-ended question, we found that most of such students suggested that EasyCloud should provide a command-line interface or a RESTful API for using EasyCloud in a shell script. Also, some *non-CS* students stated that they would like to use EasyCloud from a Web browser rather than from the console.

Concerning question "Q3", essentially all students agree that the unified API of EasyCloud simplified the development of applications in a multi-cloud environment rather than using native APIs of cloud platforms; only 3.85% students of *CS advanced* are uncertain about that.

Finally, for question "Q4", the majority of students think that EasyCloud simplifies the development of applications in multi-cloud environments even outside the context of the course they attended. However, we also note some degree of dissatisfaction. Specifically, 6.35% of students of *non-CS* neither agree nor disagree with the statement of question "Q4", 9.87% of students of *CS basic* either disagree or is neutral, and 11.54% of students of *CS advanced* either disagree or is neutral. The analysis of the comments in the open-ended question revealed that most of such students (in particular, those with a computer science background) believe that a lower-level API could be sometimes useful to exploit features specific to a particular cloud platform.

Overall, we conclude that there is a high degree of satisfaction in using EasyCloud (both by means of its textual interface and through its API) among our students, especially to carry out the assignments of our hands-on sessions. Also, students that are less satisfied essentially gave us positive feedback by suggesting adding to EasyCloud new features to improve its usability (e.g., a command-line interface to use inside shell scripts) and its flexibility (e.g., a low-level API to access features specific to a particular cloud platform). In particular, as discussed in Section 5.3, we already extended the EasyCloud API with a lower-level platform-dependent API to enable students to exploit features specific to each cloud platform that are not exposed by the higher-level platform-independent API.

5 ARCHITECTURE

The key design goals of EasyCloud are: *interoperability* (to avoid the vendor lock-in issue), *platform-independence* (to abstract away from the API heterogeneity of the different cloud platforms), *flexibility* (to exploit platform-specific features), *effective resource provisioning* (to monitor VM instances in (near) real-time and to trigger actions in response to specific events), and *ease of use* (to manage VM instances easily and fastly).

Figure 2 presents the system architecture of EasyCloud resulting from the above design goals, where solid boxes denote the components of EasyCloud (and stacked boxes represent components that can be instantiated multiple times), dashed shapes represent components external to EasyCloud (e.g., components of a cloud platform) with whom EasyCloud interacts with, and arrows denote interactions between components (e.g., an arrow from component C_i to component C_j means that C_i uses the functionality of C_j).

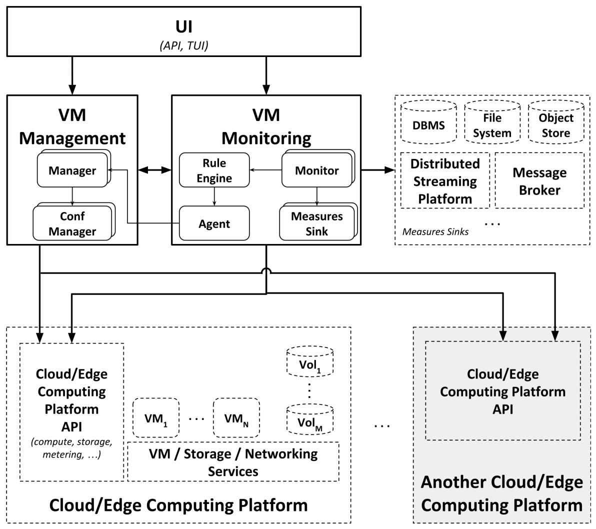

UI
(API, TUI)
VM Management
Manager
Conf Manager
VM Monitoring
Rule Engine
Monitor
Agent
Measures Sink
DBMS
File System
Object Store
Distributed Streaming Platform
Message Broker
Measures Sinks
Cloud/Edge Computing Platform API
(compute, storage, metering, ...)
VM_1
VM_N
Vol_1
Vol_M
VM / Storage / Networking Services
Cloud/Edge Computing Platform
Cloud/Edge Computing Platform API
Another Cloud/Edge Computing Platform

Figure 2: The architecture of EasyCloud.

As shown in the figure, the components of EasyCloud are grouped into three main subsystems, namely *VM Management*, *VM Monitoring* and *UI*. These subsystems provide a unified interface (independent by any cloud platform) that enables users to transparently interact with different cloud platforms simultaneously, without caring about the heterogeneity of the APIs provided by the various cloud platforms. At the same time, these subsystems also provide a low-level interface specific to each cloud platform that gives users full control to use, if needed, platform-specific features (e.g., to access VM instance attributes available only on a specific cloud platform).

The cloud platforms that EasyCloud currently supports are the following: *Amazon AWS*, *Chameleon Cloud* [23], *Google Cloud*, and *OpenStack*. Furthermore, thanks to its highly modular architecture, EasyCloud can be easily extended to support new cloud platforms by just providing the implementation of its interface specific to those new platforms.

The design and implementation details of the architecture of EasyCloud have been presented in detail in [8] (where we introduced a preliminary version of EasyCloud) and in [10] (where we presented an extended and improved version of EasyCloud). Therefore, in this section, we present just an overview of the functionality provided by the subsystems of EasyCloud so as to make this paper self-contained.

5.1 The VM Management subsystem

The *VM Management* subsystem provides platform-independent functionality for the life-cycle management of VM instances and volumes hosted by cloud infrastructures.

This subsystem consists of two main components, namely the *Manager* and the *ConfManager*, each of which may be instantiated multiple times as the number of supported cloud platforms. The *Manager* component interacts with the resource allocation and virtualization services provided by a cloud platform to provide functionality for the management of VM instances and volumes (e.g., for starting and stopping VM instances). The *ConfManager* component allows to get and set configuration parameters of both a given cloud platform (e.g., the user credentials to access Amazon EC2) and specific to EasyCloud (e.g., whether to send metric data – collected from VM instances – to *data sinks*; see Section 5.2 below).

5.2 The VM Monitoring subsystem

The *VM Monitoring* subsystem provides a unified interface for collecting in (near) real-time metric data from VM instances (e.g., CPU load), triggering actions in response to user-defined policies stated in terms of gathered metric data (e.g., create a new VM instance when the CPU load of a VM instance exceeds a given threshold), and for sending collected metric data to one or more *data sinks* (i.e., components external to EasyCloud) for further analysis and processing or simply for reporting and persistence purpose.

This subsystem consists of four main components, namely the *Monitor*, the *Rule Engine*, the *Agent* and the *Measures Sink*, which are run in separate threads and interact with each other by means of synchronized queues. The *Monitor* component collects metric data (e.g., CPU usage) for VM instances hosted by a given cloud platform by querying the telemetry services provided by that platform, and sends the gathered data to the *Measures Sink* and *Rule Engine* components for further storing and processing. The *Rule Engine* component triggers actions according to user-defined rules (stored in a *rule base*) defined in terms of metric data coming from one or more *Monitor* components. The *Agent* component executes actions for specific VM instances, as requested by the *Rule Engine* component, through the *Manager* component. Finally, the *Measures Sink* component takes the metric data received from the *Monitor* component and sends them to a data sink for further processing or persistence (e.g., to ingest data into a database, a message broker or a web dashboard); multiple instances of the *Measure Sink* component can be active at the same time, each of which receiving the same collection of metric data from the *Monitor* component and sending them to a different data sink.

5.3 The UI subsystem

The *UI* subsystem provides user interfaces to access the functionality provided by the *VM Management* and *VM Monitoring* subsystems. Specifically, at the lowest level, it exposes a unified, object-oriented API in Python that allows developers to use EasyCloud as a framework to interact with different cloud platforms in a programmatic and platform-independent way, thus avoiding to worry about the heterogeneity of the APIs of the various cloud platforms. Furthermore, this subsystem also provides a low-level (platform-dependent) API to fully exploit platform-specific features (i.e., to access platform-specific information about a given VM instance that is not provided by the higher-level, platform-independent API), so that flexibility is achieved. Moreover, it allows students to be exposed to the variety of platform-specific APIs, and to learn what those APIs convey, and how to profitably use them to achieve their goals. Finally, this subsystem provides a user-friendly text-based user interface (built atop of the platform-independent API) through which users can use EasyCloud interactively, via a textual interface.

6 EXPERIMENTAL EVALUATION

Even though, in an educational context, performance is not the first concern, it is very important that an educational toolkit is able to sustain realistic workloads so as educators can design hands-on sessions based on real-world scenarios with a large number of instances.

For this reason, EasyCloud has been designed with efficiency in mind, and encompasses several optimizations and mechanisms aimed at reducing as much as possible the overhead of its various operations.

To assess the efficiency of EasyCloud, we performed an experimental evaluation to assess its ability to scale with respect to the number of managed instances. In this section, we show the results obtained from such experimental campaign.

Each experiment consists of a series of runs where, in each one of them, we measure the *CPU time* (both in user space and in kernel space) taken by EasyCloud to retrieve the list of all M VM instances hosted on a given cloud infrastructure, and to start just N of them, with $N \leq M$. It is worth noting that we consider the CPU time instead of the elapsed real time so as to assure that our results are not affected by external and non-controllable factors, like the scheduling policy of the operating system running on the same computer where EasyCloud components are run, unpredictable network load conditions, and API rate limiting policies specific to a given cloud provider (e.g., [3]).

We use as performance index the *average CPU time* that we computed by averaging the CPU time obtained in the various runs of an experiment, until the relative error of its 95% confidence interval is of at most 4% [13]. Therefore, the number of runs of a single experiment is computed online, and may change from one experiment to another.

The experiments were carried out on a physical testbed consisting of one Fujitsu Server PRIMERGY RX300 S7 (equipped with two 2.4 GHz Intel Xeon E5-2665 processors with 8 cores and 96 GiB of RAM, and running the Linux kernel version

Table 1: Fitting of linear models to the experimental results. Each cell (i, s) of the table contains the intercept i and the slope s of the regression line fitted to the results obtained with EasyCloud on a particular cloud infrastructure (row).

Cloud infrastructure	(intercept, slope)
AWS EC2	$(0.067, 0.006)$
Chameleon Cloud	$(0.192, 0.012)$

5.6.13) located in Italy and used to run EasyCloud, and a public cloud infrastructure hosting the set of instances that we managed with EasyCloud. We considered two public cloud infrastructures, that is Chameleon Cloud (site "KVM@TACC", located in Texas, USA) and AWS EC2 (region "us-east-1", located in Virginia, USA), where we hosted $M = 45$ Linux instances.

In Figure 3, we present the results of the experiments both for AWS EC2 (see Figure 3a) and for Chameleon Cloud (see Figure 3b). In these figures, each filled circle denotes the average CPU time required by EasyCloud to perform the above operations for a given number of instances N (with N ranging from 5 to 45), and the associated error bar represents its 95% confidence interval. These figures show that EasyCloud is able to scale well with respect to the number of instances to manage as the average CPU time just grows linearly as a function of the number of managed instances.

To better understand this linear trend, we fitted a linear model to the results obtained in the various experiments for each cloud infrastructure. The parameters of the resulting linear models are reported in Table 1, where the pair (i, s) in each cell represents the intercept i and the slope s of the regression line fitted to the experimental results obtained with EasyCloud on the cloud infrastructure of the corresponding row. For instance, we expect that, on average, EasyCloud will take approximately 0.667 seconds to manage 100 instances hosted on the AWS EC2 infrastructure, and nearly 1.392 seconds to manage the same number of instances hosted on the Chameleon Cloud infrastructure.

7 CONCLUSIONS

In this paper, we presented EasyCloud, a modular toolkit that provides a unified interface to various cloud platforms for managing VMs, monitoring their performance, and storing for further analysis the collected metrics, as well as an intuitive and interactive user interface.

With EasyCloud, students and educators can experiment with both the basic and advanced concepts related to the cloud computing in a multi-cloud environment and without caring of the API and user-interface heterogeneity of the various cloud providers. Our experimental evaluation shows that EasyCloud can be efficiently used in real-world scenarios as it is able to scale well with respect to the number of VM instances.

In the future, we plan to extend EasyCloud as follows to support more advanced teaching activities. Besides extensions suggested by our students, namely a RESTful API to access EasyCloud functionalities and to support a richer set of rule types (e.g., with the *Intellect* [36] package), we plan (a) to support new cloud/edge computing platforms (e.g., *IBM Cloud*, *Microsoft Azure*, *EdgeX* and *MicroK8s*) and new measures sinks (e.g., *Amazon S3*), (b) to add new features like resource reservation and intelligent fault-detection system techniques [4, 28] and, finally, (c) to integrate cloud/edge infrastructure management [5, 6] and experimentation [7] systems.

ACKNOWLEDGMENTS

This research has a financial support of the Università del Piemonte Orientale and of the INdAM – GNCS Project 2020. Results presented in this paper were in part obtained using the Chameleon testbed supported by the National Science Foundation.

[1] ACM/IEEE-CS. 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* ACM. https://doi.org/10.1145/2534860

[2] ACM/IEEE-CS. 2016. *Computer Engineering Curricula 2016: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering.* Technical Report CE2016. ACM.

[3] Amazon. 2021. Request throttling for the Amazon EC2 API. https://docs.aws.amazon.com/AWSEC2/latest/APIReference/throttling.html

[4] C. Anglano and M. Botta. 2002. NOW G-net: Learning classification programs on networks of workstations. *IEEE Transactions on Evolutionary Computation* 6, 5 (2002), 463–480.

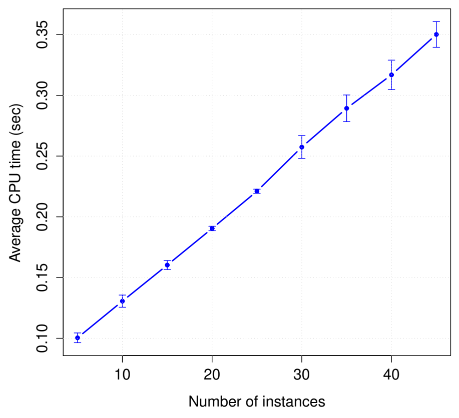
[5] Cosimo Anglano, Massimo Canonico, and Marco Guazzone. 2017. FCMS: A fuzzy controller for CPU and memory consolidation under SLA constraints. *Concurrency Computat.: Pract. Exper.* 29, 5 (2017), e3968.

[6] Cosimo Anglano, Massimo Canonico, and Marco Guazzone. 2018. Profit-aware Resource Management for Edge Computing Systems. In *Proc. of the 1st International Workshop on Edge Systems, Analytics and Networking (EdgeSys).* 25–30.

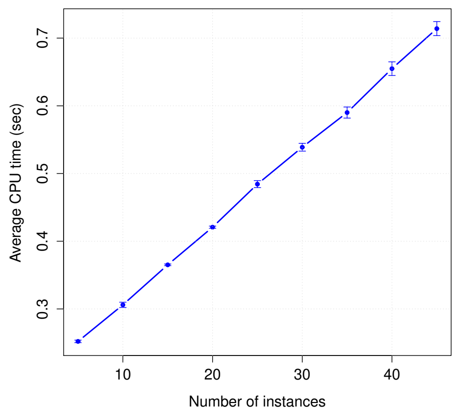
[7] Cosimo Anglano, Massimo Canonico, and Marco Guazzone. 2018. Prometheus: A flexible toolkit for the experimentation with virtualized infrastructures. *Concurrency Computat.: Pract. Exper.* 30, 11 (2018), e4400.

[8] Cosimo Anglano, Massimo Canonico, and Marco Guazzone. 2020. EasyCloud: a Rule based Toolkit for Multi-platform Cloud/Edge Service Management. In *Proc. of the 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)* (Paris, France). IEEE, 188–195. https://doi.org/10.1109/FMEC49853.2020.9144821

[9] Cosimo Anglano, Massimo Canonico, and Marco Guazzone. 2020. Teaching Cloud Computing: Motivations, Challenges and Tools. In *Proc. of the 2020 IEEE International Parallel and Distrituted Processing Symposium Workshops (IPDPSW)* (New Orleans, LA, USA). IEEE,



(a) AWS EC2.



(b) Chameleon Cloud.

Figure 3: Performance of EasyCloud for increasing numbers of managed instances and on different cloud infrastructures. Each filled circle represents the average CPU time (in seconds) taken by EasyCloud to manage a given number of instances hosted on a particular cloud platforms, and the associated error bar denotes its 95% confidence interval.

300–306. https://doi.org/10.1109/IPDPSW50202.2020.00062

[10] Cosimo Anglano, Massimo Canonico, and Marco Guazzone. 2021. EasyCloud: Multi-clouds made easy. In *Proc. of the 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. 526–531. https://doi.org/10.1109/COMPSAC51774.2021.00078

[11] Cosimo Anglano, Massimo Canonico, and Marco Guazzone. 2021. EasyCloud repository. Available: https://gitlab.di.unipmn.it/DCS/easycloud. Accessed: May 28, 2021.

[12] Apache Software Foundation. 2021. The Java Multi-Cloud Toolkit. Available: https://jclouds.apache.org/. Accessed: May 28, 2021.

[13] Jerry Banks, John S. Carson, Barry L. Nelson, and David M. Nicol. 2010. *Discrete-Event System Simulation* (5th ed.). Prentice Hall.

[14] Shannon Bradshaw, Eoin Brazil, and Kristina Chodorow. 2019. *MongoDB: The Definitive Guide* (3 ed.). O'Reilly.

[15] Brian Brazil. 2018. *Prometheus: Up & Running*. O'Reilly.

[16] Massimo Canonico and Marco Guazzone. 2021. Teaching Cloud Computing website. Available: https://tcc.uniupo.it. Accessed: August 2, 2021.

[17] Josiah Carlson. 2013. *Redis in Action*. Manning.

[18] Jeff Carpenter and Eben Hewitt. 2020. *Cassandra: The Definitive Guide* (3 ed.). O'Reilly.

[19] Jason Cole and Helen Foster. 2007. *Using Moodle: Teaching with the popular open source course management system*. O'Reilly Media, Inc.

[20] Google. 2021. Codelabs. Available: https://codelabs.developers.google.com/. Accessed: May 28, 2021.

[21] InternetNews. 2021. Why 'Cloud Computing' Is for the Birds. Available: https://www.internetnews.com/blog/why-cloud-computing-is-for-the-birds/. Accessed: August 2, 2021.

[22] Yaser Jararweh, Zakarea Alshara, Moath Jarrah, Mazen Kharbutli, and Mohammad N. Alsaleh. 2013. TeachCloud: a cloud computing educational toolkit. *International Journal of Cloud Computing* 2, 2–3 (2013), 237–257. https://doi.org/10.1504/IJCC.2013.055269 PMID: 55269.

[23] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S. Gunawi, Cody Hammock, Joe Mambretti, Alexander Barnes, François Halbah, Alex Rocha, and Joe Stubbs. 2020. Lessons Learned from the Chameleon Testbed. In *Proc. of the 2020 USENIX Annual Technical Conference (USENIX ATC 20)*. USENIX Association, 219–233. https://www.usenix.org/conference/atc20/presentation/keahey

[24] Jay Kreps, Neha Narkhede, and Jun Rao. 2011. Kafka: A Distributed Messaging System for Log Processing. In *Proc. of the 6th International Workshop on Networking Meets Databases (NetDB)*.

[25] Rensis Likert. 1932. A Technique for the Measurement of Attitudes. *Archives of Psychology* 22, 140 (1932), 1–55.

[26] Linux System. 2021. The stress tool. Available: https://https://linux.die.net/man/1/stress. Accessed: May 28, 2021.

[27] Peter M. Mell and Timothy Grance. 2011. *The NIST Definition of Cloud Computing*. Technical Report SP 800-145. NIST, Gaithersburg, MD, USA.

[28] Stefania Montani and Cosimo Anglano. 2006. Case-Based Reasoning for Autonomous Service Failure Diagnosis and Remediation in Software Systems. In *Advances in Case-Based Reasoning*, Thomas R. Roth-Berghofer, Mehmet H. Göker, and H. Altay Güvenir (Eds.). Springer Berlin Heidelberg, 489–503.

[29] qwiklabs. 2021. Hands-On Cloud Training. Available: https://www.qwiklabs.com. Accessed: May 28, 2021.

[30] Naomi B. Robbins and Richard M. Heiberger. 2011. Plotting Likert and other rating scales. In *Proc. of the 2011 Joint Statistical Meeting*, Vol. 1.

[31] Bruce Snyder, Dejan Bosanac, and Rob Davies. 2011. *ActiveMQ in Action*. Manning.

[32] Ian Stoica and Scott Shenker. 2021. From Cloud Computing to Sky Computing. In *Proc. of the 18th Workshop on Hot Topics in Operating Systems (HotOS)*.

[33] C. Tranoris. 2011. Adopting the DSM paradigm: Defining federation scenarios through resource brokers for experimentally driven research. In *Proc. of 12th IFIP/IEEE International Symposium on Integrated Network Management (IM) and Workshops*. 1140–1147. https://doi.org/10.1109/INM.2011.5990574

[34] Alvaro Videla and Jason J.W. Williams. 2012. *RabbitMQ in Action: Distributed Messaging for Everyone.* Manning.

[35] Gregor Von Laszewski, Fugang Wang, Hyungro Lee, Heng Chen, and Geoffrey C Fox. 2014. Accessing multiple clouds with cloudmesh. In *Proc. of the 2014 ACM international workshop on Software-defined ecosystems (BigSystem).* 21–28.

[36] Michael Joseph Walsh. 2021. Intellect: A Domain-specific language and Rules Engine for Python. Available: https://github.com/nemonik/Intellect. Accessed: May 28, 2021.

A APPENDIX

In this section, we report the steps required to repeat the experiments presented in Section 6. Actually, this section provides just an overview of the above steps. More detailed instructions are available in the following public repository that we specifically created to store the artifacts needed to reproduce our experiments: https://gitlab.di.unipmn.it/sguazt/easycloud_artifacts-ccr2021. Please, visit that URL and start following the instructions contained in the `README.md` file.

A.1 EasyCloud installation

The first step to reproduce the experiments consists in installing the EasyCloud toolkit on your computer. To do so, follow the guide available at the following URL: https://gitlab.di.unipmn.it/sguazt/easycloud_artifacts-ccr2021/-/blob/main/docs/install.md.

After EasyCloud has been installed, you can move to next sections so as run the experiments either on AWS (Section A.2) or on the Chameleon testbed (Section A.3).

A.2 AWS experiments

To reproduce the experiments we run on AWS and to plot the obtained results, follow the guide available at the URL: https://gitlab.di.unipmn.it/sguazt/easycloud_artifacts-ccr2021/-/blob/main/docs/aws.md.

A.3 Chameleon experiments

To reproduce the experiments we run on Chameleon and to plot the obtained results, follow the guide available at the URL: https://gitlab.di.unipmn.it/sguazt/easycloud_artifacts-ccr2021/-/blob/main/docs/chameleon.md.