HAL
open science

OCCI-Compliant Cloud Configuration Simulation

Mehdi Ahmed-Nacer, Walid Gaaloul, Samir Tata

To cite this version:

Mehdi Ahmed-Nacer, Walid Gaaloul, Samir Tata. OCCI-Compliant Cloud Configuration Simulation. 2017. hal-01523933

HAL Id: hal-01523933

https://hal.science/hal-01523933v1

Preprint submitted on 18 May 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

OCCI-Compliant Cloud Configuration Simulation

Mehdi Ahmed-Nacer*†, Walid Gaaloul† and Samir Tata‡
*University of Sciences and Technology Houari Boumediene, Algiers, Algeria
†SAMOVAR, Telecom SudParis, CNRS. Universite Paris-Saclay. 9, rue Charles Fourier. 91011 Evry Cedex, France
‡IBM Research-Almaden. San Jose, CA, 95120, United States
{firstname.lastname}@telecom-sudparis.eu, m.ahmed.nacer@usthb.dz, stata@us.ibm.com

***Abstract*—In recent years many organizations such as, Amazon, Google, Microsoft, have accelerated the development of their cloud computing ecosystem. This rapid development has created a plethora of cloud resource management interfaces for provisioning, supervising, and managing cloud resources. Thus, there is an obvious need for the standardization of cloud resource management interfaces to cope with the prevalent issues of heterogeneity, integration, and portability issues. To this end, Open Cloud Computing Interface (OCCI) proposed one of the first widely accepted, community-based, open standard for managing any kinds of cloud resources. However, there is a strong need for having a simulation environment to study and analyze the performance of OCCI-compliant applications for two main reasons: (i) in absence of a simulation environment, these applications have to be tested on real cloud infrastructure, which induces a cost issue and (ii) various conditions prevailing in cloud environments are beyond the control of the developers involved in analyzing cloud resource allocation. Therefore, in this paper we propose, (i) an extension to the OCCI metamodel dedicated to the simulation of cloud resources using an open-source cloud simulation framework called CloudSim, (ii) a user-friendly graphical interface to generate the OCCI configuration for simulation.**

***Keywords*-Cloud Computing, Open Cloud Computing Interface, Simulation;**

I. Introduction

The evolution of cloud computing over the past few years is potentially one of the major advances in the history of computing. Many companies, such as Amazon, Google, Microsoft and so on, have accelerated the development of their cloud computing ecosystem and have been providing their services to a larger number of users. However, in order to achieve its full potential, cloud computing needs to tackle many challenges involved in cloud resource management [1]. Indeed, each cloud offering has its own cloud resource management (CRM) interfaces to precise how cloud clients/applications/users interact with the cloud resources. This has led to a plethora of CRM-API's, proposed by Amazon, CloudStack, OpenStack, CloudBees, OpenShift, Cloud Foundry, to name a few.

Thereby there is an obvious need for cloud computing standards to cope with four main issues: (i) heterogeneity of cloud offers, (ii) interoperability between CRM-API, (iii) integration of CRM-API for building multi-cloud systems, and (iv) portability of cloud management applications.

To this end, Open Cloud Computing Interface (OCCI) was introduced [2]. It proposes one of the first widely accepted, community-based, open standards for managing any kinds of cloud resources [3]. But as it is specified in natural language, OCCI is considered to be imprecise, ambiguous, incomplete, and needs a precise definition of its core concepts. This being a relevant issue has been recently addressed by the scientific community and has been supported by many scientific calls for research projects. The OCCIware[1] project is one of such projects. It proposes a new precise metamodel for OCCI, named OCCIware metamodel [4], along with an enhanced tooling environment called OCCIware Studio.

OCCIware Studio is a set of tools developed for designing, managing and analyzing any kind of cloud computing resources. Through OCCIware Studio, users are able to graphically design a cloud system called 'OCCI Configuration'. The users can validate this configuration and deploy it in real cloud computing infrastructure. This configuration is composed of OCCI resources instances and links between them. However, porting an OCCI configuration over a real cloud platform and evaluating it at scale comes with a high cost. That motivated the research idea on integrating a simulation tool into OCCIware Studio.

Furthermore, simulation technology has become increasingly popular, both in the cloud industry and in academia. It allows users to evaluate their algorithms and applications before deploying them on a real cloud environment. Using a real cloud infrastructure limits the experiments to the scale of the infrastructure, and makes the reproduction of results an extremely difficult task. The main reason for this being that, the conditions prevailing in the internet-based environments are beyond the control of the developers involved in analyzing the resource allocation and application of the scheduling algorithms. To avoid these issues, simulation tools open the possibility of experimenting and evaluating the hypothesis in an environment where one can reproduce tests easily prior to the actual software development. This is of special significance in the case of cloud computing, where access to the cloud infrastructure incurs payments in real currency.

Given the importance of the simulation in cloud environment and the lack of OCCI metamodel for supporting

[1] http://www.occiware.org/

a simulation of OCCI resources, we propose in this paper an extension of OCCIware metamodel for supporting a simulation of OCCI based cloud configurations. To achieve this aim, different tracks have to to be studied. These tracks are summarized in Figure 1 and the study is conducted as follows:

1) Study the suitability and the compatibility of various cloud simulators with OCCIware metamodel. This study allows us to select the suitable simulator and extract the entities used by this simulator;
2) Extend OCCIware metamodel for simulation environment. This extension defines the position of the simulator entities in the context of OCCIware;
3) Develop a user-friendly simulation studio using which the users can design and model a simulation configuration;
4) Integrate a simulator tool in OCCIware metamodel and connect it with the simulation studio.

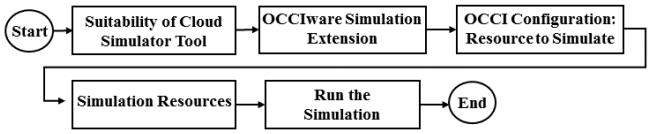

Figure 1: Simulation Flow

Thus, the objective of our research is to provide the users of the OCCIware framework with a possibility to perform simulations in a easy and intuitive manner along with supporting the principle of interoperability. We will measure the easiness and efficiency of our tool based on the time taken to complete the design of the configuration file using our tool and the number of lines-of-code (excluding whitespace). Likewise, the principle of interoperability would be assessed through showing various use cases being supported by our tool.

The remainder of this paper is structured as follows: Section II provides information about the OCCI Standard, OCCIware metamodel along with OCCIware studio. Section III presents the proposed OCCI extension for simulation. Section IV describes the implementation. Section VI evaluates the existing related work. Section VII concludes the paper and discusses the planned future works.

II. Background

In this section, we provide basic information about the OCCI standard and the OCCIware Metamodel. We also introduce the OCCIware studio in this section, as we use this studio to design the simulation extension for the OCCI Metamodel.

A. OCCI Standard

The Open Cloud Computing Interface (OCCI) is a RESTful protocol and provides API's for all kinds of management tasks in cloud environments [5]. The OCCI core model is represented as an UML class diagram where cloud resources and their relationships have been represented. Among various OCCI specifications, in context of this paper, we are mainly interested in two of them i.e. OCCI Core (v1.1) [2] and OCCI Infrastructure (v1.1) [6].

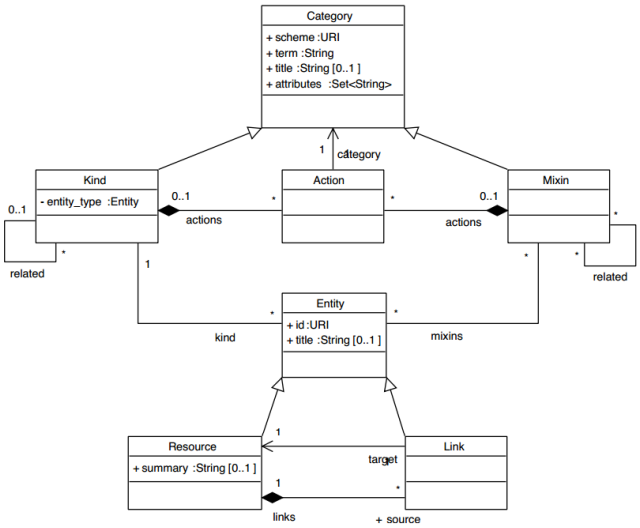

Figure 2: OCCI Core Model

The OCCI Core provides the formal definition of the OCCI core model as shown in the figure 2. This model is an abstraction of real-world resources, including the means to identify, classify, associate and extend those resources. It describes cloud resources as instances of *Resource* or a sub-type thereof. *Resources* could be combined and linked to each other using instances of *Link* or a sub-type of this latter. *Resource* and *Link* are sub-types of *Entity*. Each *Entity* instance is typed using an instance of the class *Kind* and could be extended using one or more instances of the class *Mixin*. *Kind* and *Mixin* are subtypes of *Category* and each Category instance can have one or more instances of the class *Action*. A fundamental advantage of the OCCI Core Model is its extensibility. It is noteworthy that any extension will be discoverable and visible to an OCCI client at run-time. An OCCI client can connect to an OCCI implementation using an extended OCCI Core Model, without knowing anything in advance, and still be able to discover and understand, at run-time, all the instance types supported by that implementation. Extending OCCI core is possible using sub-types of the existing types or using *Mixins*.

OCCI Infrastructure [6] is an extension of the core model to represent the cloud infrastructure layer. This extension provides the definition of new resources that inherit the core basic types, Resource and Link. As shown in Figure 3, new *Resource* subclasses are *Network*, *Compute* and *Storage* while the new *Link* subclasses are *StorageLink* and *NetworkInterface*. The *Compute* resource represents

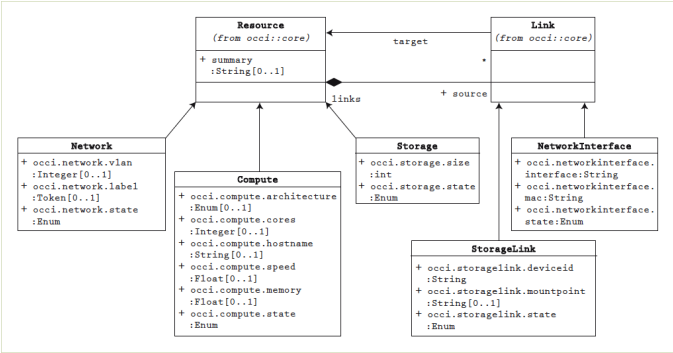

Figure 3: OCCI Infrastructure Model

a generic information processing resource like a virtual machine. However, *Network* resource represents networking entity like a virtual switch. The last defined resource type for this extension is the *Storage* that represents resources that record information to a data storage device.

Even though OCCI standatd is widely accepted in cloud computing community, it lacks a precise definition of its core concepts. In [4], the authors identify five conceptual drawbacks/limitations on the OCCI Core Model, which are: (i) Informal model, (ii) Imprecise type classification system, (iii) Non-extensible data type system,(iv) Vague and incomplete extension concept, (v) Configuration concept undefined. To tackle these issues, OCCIware project proposed a new and precise metamodel for OCCI, named OCCIware Metamodel [4].

B. OCCIware Metamodel

OCCIware metamodel [4] is a precise metamodel for OCCI. This metamodel rigorously defines the static semantics of the OCCI core concepts, which comprises of a precise type classification system, of an extensible data type system, and of both extension and configuration concepts. This metamodel is based on the eclipse modeling framework (EMF). The structure of OCCIware metamodel is illustrated in Figure 4.

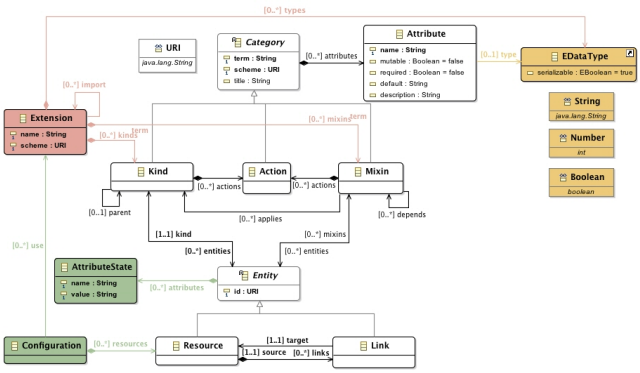

Figure 4: Diagram of OCCIware Metamodel

For our study of OCCI compliant simulation, we have to provide basic definitions of OCCI Extension and OCCI Configuration.

1) OCCI Extension: The OCCI Extension specifications consist of multiple documents each describing a particular extension of the OCCI Core Model. The extension documents describe additions to the OCCI Core Model defined within the OCCI specification suite [7]. Various extensions have been proposed such as infrastructure extension [6], platform extension [8], application extension [8], etc.
2) OCCI Configuration: A configuration is an abstraction of an OCCI-based running system, and is composed of resource and link instances. A configuration must explicitly state which extensions it uses. Modeling a configuration offline could allow designers to think about and analyze their cloud systems without requiring to deploy them actually in the clouds.

The OCCIware Metamodel enhanced tooling paradigm called OCCIware Studio that is able to design, analyze and manage any kind of cloud computing resources. Through OCCIware Studio the users create their extensions and configurations graphically.

C. OCCIware Studio

OCCIware studio is a set of tools designed to ease the development and the deployment of OCCI-based solutions. The studio implementation is based on Eclipse, which provides various frameworks for simplifying the development of OCCI studio's tools

To conduct our study, we use this studio to design the simulation extension and generate OCCI simulation configuration. Figure 5 shows the main feature of OCCIware Studio that interests us in order to design an OCCI simulation. Three of them have already developed: **OCCI Designer** (a graphical tool, based on Sirius, for designing the extension and configurations using diagrams), **OCCI Editor** (a textual editor to edit OCCI models) **OCCI Validator** (a tool to validate OCCI extensions and configurations). The fourth tool **OCCI Simulator** is the purpose of this paper. It simulates an OCCI configuration in order to evaluate performance metrics without deploying it into the clouds and paying for cloud resource usage

III. OCCI SIMULATION

In this section, we first define the OCCI simulation extension. Afterward, we describe the generation of OCCI simulation configuration.

A. Tool selection

To develop and analyze any new cloud environment with the help of the simulators, it is required to understand the existing cloud simulators along with their pros and cons. Among the various existing simulators, we need to study

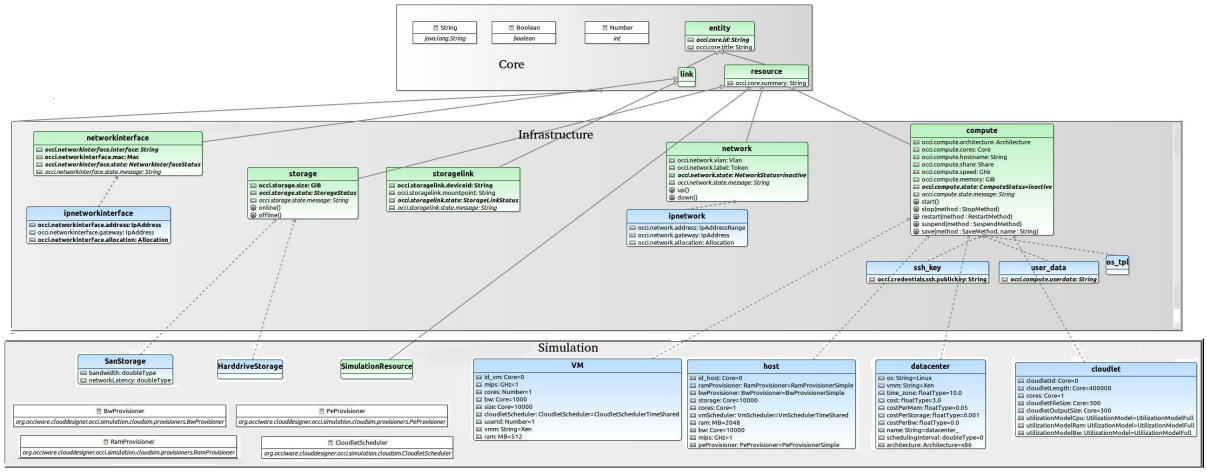

Figure 6: Overview of the Defined Extension

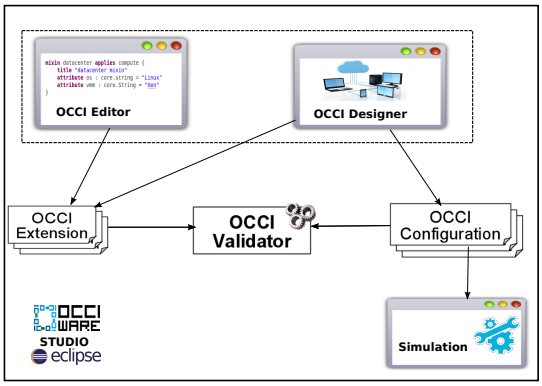

Figure 5: OCCIware Simulation Studio

and choose the most appropriate simulator for OCCIware Metamodel. Such a study will allow us to define the different entities for the simulation extension.

In the literature, many cloud simulators exist such as SimGrid [9], CloudSim [10], GreenCloud [11], iCanCloud [12], GridSim [13] and many others. However because of the varieties of challenging issues of cloud computing, one particular existing cloud simulator isolates clearly the multi-layer service abstractions (SaaS, PaaS, and IaaS) differentiation. This simulator is the CloudSim [10].

We believe that CloudSim [10] is the most appropriate simulator to achieve our goal. CloudSim is a generalized and extensible simulation framework that allows seamless modeling, simulation, and experimentation of emerging cloud computing infrastructures and application services. It is open source and has been developed in java programming language which makes it compatible with OCCIware metamodel. By using CloudSim, researchers and developers can test the performance of a newly developed application service in a controlled environment. Moreover, CloudSim allows a user to model and simulate all the cloud infrastructure resources. It also requires much less effort and time to implement cloud based application. The developers can model and test the performance of their application services in heterogeneous cloud environments with little programming and deployment effort.

Since OCCI simulation extension uses CloudSim entities, we define in the following the different entities of CloudSim.

B. CloudSim entities

The main components of CloudSim are: *Datacenters*, *Hosts*, *Virtual Machines* (VM) and *Cloudlets*. *Datacenter* class models the core infrastructure level services (hardware, software) offered by resource providers in cloud computing environment. The *Datacenters* encapsulates a set of *Hosts* and their resource configurations (memory, cores, capacity, and storage). Furthermore, every *Datacenter* component instantiates a generalized resource provisioning component that implements a set of policies for allocating bandwidth, memory, and storage devices. Each *Host* component can instantiate multiple *VMs* and allocate cores based on predefined processor sharing policies. Each *VM* has an owner, which can submit *Cloudlets* to the *VM* to be executed.

According to the resources needed by CloudSim, some of them must be defined as a new *Kind* in OCCI metamodel, while others are defined as a *Mixins* applied for the existing *Kind* from OCCI core and infrastructure model.

Figure 6[2] presents an overview of the different OCCI resources and the connection of these types with OCCI core and infrastructure entities. In the following we present, the relationships between the main OCCI resources types and the simulator entities.

[2]Please note that the figure of the simulation extension illustrated in Fig 6 is not an UML digram. The figure was generated through the OCCI studio which is designed for the generation of OCCI extensions. The Kinds are in green color, while the Mixins are in blue.

The *Datacenter* resource type represents the resource providers in CloudSim. It is the main hardware infrastructure that provides services for servicing user requests. Datacenter has a set of hosts (physical machine), VM and Object Constraint Language (OCL) different management components and policies. A Datacenter has a set of interconnected hosts that are managed by a set of management policies. The Datacenter is a *Mixin* applied for COMPUTE resources, base type defined in OCCI infrastructure.

Host executes actions related to management of virtual machines (e.g., creation and destruction). A host has a defined policy for provisioning memory and bandwidth, as well as an allocation policy for processing elements (Pe's) to virtual machines. A host is associated to a datacenter. It can host virtual machines. The Host is a *Mixin* applied for COMPUTE resources, base type defined in OCCI infrastructure..

VM runs inside a Host, sharing *hostList* with other VMs. It processes *cloudlets*. This processing happens according to a policy, defined by the *CloudletScheduler*. Each VM has an owner, which can submit cloudlets to the VM to be executed. The VM is a *Mixin* applied for COMPUTE resources, base type defined in OCCI infrastructure.

Cloudlet models the cloud-based application services or the CloudSim task running in Vms. Each cloudlet is defined by the number of CPU operations it requires. The Cloudlet is a *Mixin* applied for COMPUTE resources, base type defined in OCCI infrastructure. *HardDriveStorage* represents the storage system and the behaviour of a typical hard drive storage. The *HardDriveStorage* is a *Mixin* applied for STORAGE resources, base type defined in OCCI infrastructure.

SanStorage represents a storage area network composed of a set of hard disks connected in a LAN. The *SanStorage* is a *Mixin* applied for STORAGE resources, base type defined in OCCI infrastructure.

Furthermore, we also define new *Kinds* derived from OCCI core model as follows:

- *Contains* to connect the resources between them. For example, *Datacenter* contains a list of hosts. Each host contains a list of VM. Each VM contains a list of CloudLet, etc. *Contains* link extends LINK resource of OCCI core model.
- *SimulationResource* this kind is a generic cloud resource. It extends RESOURCE of OCCI core model.

In addition to that, we define some *Types*. These types are associated to the attributes of CloudSim entities:

- *CloudletScheduler*: represents the policy of scheduling performed by a virtual machine (VM).
- *VmScheduler*: represents the policy used by a *VM* to share processing power among *VMs* running in a host.
- *RamProvisioner*: represents the provisioning policy of memory to virtual machines inside a host.
- *BwProvisioner*: represents the provisioning policy of bandwidth to virtual machines inside a host.
- *PeProvisioner*: defines native types of processing elements provisioners

.

The OCCI simulation extension is implemented as an Eclipse modeling project, which is an instance of OCCIware metamodel. Based on this extension, the users generate an OCCI simulation configuration.

C. OCCI simulation configuration

Through OCCI Designer the user generates an OCCI simulation configuration. This configuration uses the described extension and contains two kinds of resources:

1) *Resource to simulate*: are the resources defined in OCCI metamodel such as *compute* and *link* instances
2) *Simulation resources*: represent the resources used by CloudSim simulator such as datacenter, host, VMs and cloudlets

During the generation of OCCI simulation configuration, the user have to map from *resources to simulate* to *simulation resources*. This step specifies which *resources to simulate* is associated to which *simulation resources*. In other words, it is a traduction from OCCI defined entities to CloudSim entities.

The OCCI simulation designer allows to generate an OCCI simulation configuration in two different ways:

1- From an existing configuration: the users generate an OCCI simulation configuration from an existing OCCI configuration. In this case, the user intervention is needed. Indeed, OCCI configuration does not contain enough information that allows to map from *resource to simulate* $\rightarrow$ *simulation resources*. In addition, it is difficult to understand the intention of the user about which resource is to be simulated (e.g. a compute resource can be a datacenter, host, VMs, etc). Thus, the user intervention is required to specify which *resource to simulate* corresponding to which *simulation resource*.

2- Configuration specific for simulation: this configuration contains *Entities* that can be *Resources* or *Links* defined in OCCI Core Model (Fig 2). From the simulation designer, the user inserts the *Kind* instances defined for the infrastructure sub-types of *Resource* and *Link*, e.g. a Compute, Storage, etc (Fig 3). At this stage, the configuration is a set of infrastructure entities. To understand what the user wants to simulate, the user intervention is not required. Indeed, the user needs to associate these entities to the defined *Mixins*, e.g. datacenter, host, VMs, etc (Fig 6). For instance, the user can associate a Compute resource to it a Datacenter mixin. Therefore this Compute is considered as a datacenter simulation resource.

To help the users to design an OCCI simulation configuration, we developed a user-friendly simulation GUI on top of Eclipse: OCCI simulation Designer. In the next section, we detail the implementation.

IV. A Tool for OCCI Simulation Configuration

In order to extend OCCI Studio for supporting a simulation environment, we define a new modeler. This modeler provides a graphical interface developed on top of Eclipse Sirius[3] and a textual editor developed with Xtext. The modeler also provides a powerful constraint checking, helping developers to generate correct configuration. The *OCCI Simulation Designer*, generated from OCCI Simulation extension, allows the creation of simulation resources layout and runs the simulation. Figure 7 presents this simulation designer.

- Frame (a) in figure 7 displays the Eclipse Model Explorer used to navigate through the simulation project containing the OCCI simulation configuration.
- Frame (b) in figure 7 shows the OCCI Editor that provides a textual representation of simulation configuration.
- Frame (c) in figure 7 shows the OCCI Simulation Designer that provides a graphical representation of simulation configuration.
- Frame (d) of figure 7 shows the palette of the OCCI Designer, with palette elements to import an existing configuration, create OCCI resources and map them to Cloudsim resources, establish relations between resources, and assign new attributes to resources.
- Frame (e) of figure 7 contains the Eclipse properties editor for viewing and modifying attributes of a selected modeling element.

From OCCI simulation designer, the user has the ability to import any OCCI configuration. In this case, the OCCI configuration contains some resources, but not sufficient to run the simulation. As discussed in the previous section, the user intervention is required. To this end, a set of Cloudsim entities is proposed to annotate the resources. Moreover, the user can add another simulation resource and attributes from the modeler. Please note that when the user annotates an OCCI resource with CloudSim entity, all the attributes of this entity are imported from the simulation extension and included in the annotated resource.

Some verifications are performed to guarantee a correct configuration. These verifications ensure taht (1) all resources needed by CloudSim are in the configuration, (2) a resource is tagged by only one tag, (3) a correct match between the tag and resources (e.g. a storage cannot be tagged by a Datacenter or a Host).

Once the simulation configuration is created, the user has to execute the simulation by selecting the "Run Simulation" from the right-click menu. This will start the simulation. Once the simulation is completed, a new window appear in which we display the simulation result. The results will list out the data collected from the simulation presented in table I.

[3]https://eclipse.org/sirius/

Metric	Description
MemoryUsage	% memory used
DiskUsage	% disk used
AvgCpu	average CPU utilization
TimeExec	total simulation time
CostPe	cost of processing used
CostMem	cost of memory used
CostStorage	cost of storage used
CostBw	cost of bandwidth used

Table I: List of Metrics

Basically, CloudSim lacks the ability to bind the specified resources defined by the user. It assumes that the physical infrastructure is abstracted from Cloud users/brokers. For instance, it is not possible to create two VMs in two different datacenters if only one datacenter can host them. However, in our metamodel we need to give to the users the ability to choose where the cloud resources should be placed on. To this end, we need to develop a custom broker and override several methods. Thereby, we extend these two main classes:

1) *DatacenterBroker*: modifying the way VM provisioning requests are submitted to data centers and the way cloudlets are submitted and assigned to VMs.
2) *VmAllocationPolicy*: we extend this abstract class to implement your own algorithms for deciding which host a new VM should be placed on.

Once the simulation configuration file is created, the users can easily launch the simulation by a simple right click. The simulation measures the system performance and the cost of some resource consumed. The list of metrics measurement are illustrated in Table I.

V. Experiments

In the section, our objective is to experimentally prove the efficiency (in terms of easiness, coverage) of our simulation tool. In fact, the proposed extension can be easily integrated into any existing cloud computing API based on OCCI model such as OpenStack, Rackspace and Amazon S3 [14]. A user-friendly simulation GUI based on the proposed extension is provided to design easily the simulation configuration. Through this GUI, the users do not need to understand the concepts and principles of cloud computing such as, data centers and their federation, load balancing and task scheduling. The users just have to slide a CloudSim entities from the panel and generate their simulation configuration. This model contributes to saving time by decreasing the programming effort.

Otherwise, the users have to generate a manifest file that describes the simulation configuration manually. This manifest is the only input required by CloudSim simulator. Once the manifest file is generated, the users need to write around 200 line of Java code for a simple configuration. They need to go through a very complex process: (1) Initialize the Cloudsim packages; (2) Create Datacenters and

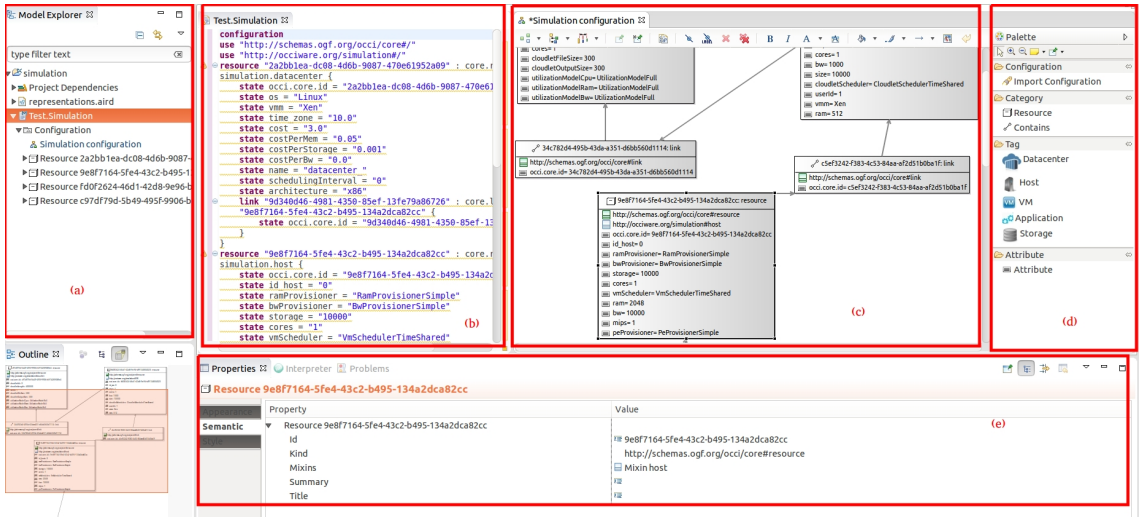

Figure 7: A Screenshot of Simulation Designer

set the needed Datacenter Characteristics; (3) Create Broker; (4) Create virtual machine and set the configuration of each virtual machine, such as mips, size, ram and processing element number; add virtual machine to the virtual machine list; submit the virtual machine list to the broker; (5) Create Cloudlet (6) similarly, set the properties of each Cloudlet and add it into the cloudlet list; (7) submit the cloudlet list to the broker; (8) Implement a tasks scheduling function in the DatacenterBroker; (9) Bind the cloudlets to VMs by calling the function defined in DatacenterBroker to apply your own task scheduling algorithm; and (10) Start the simulation. While, by using our proposed GUI, the users have just to slide CloudSim entities from a panel and the manifest is generated automatically without the need of any code.

In order to quantitatively evaluate our approach, we use 15 use cases described in the three different standard specifications : 4 use cases from TOSCA, 4 use cases from OCCI and 2 use cases for CIMI[4].

We use our simulation designer to construct the configurations of these use cases. The number of entities that can be represented, the coverage percentage and the number of code lines that can be reduced are shown in Table II. The results show that most of the configuration are fully covered by simulation designer, apart of CIMI, which has a coverage of 60%. Basically, we found that all use cases from OCCI and TOSCA are fully generated for simulation by taking into account all the described features. However, in CIMI, only the user profil and cloud federation features are not supported by the simulation designer. Therefore, we can conclude, by taking into consideration these experiments, that our simulator ensures good standard coverage interoperability by handling representative uses cases coming from three of the most used Cloud standards.

[4]available at http://www-inf.it-sudparis.eu/SIMBAD/tools/linked-cr/usecases

Standard	Nbr of Entities	Covered	Nbr of reduced Lines
TOSCA	8	100%	1600
OCCI	4	100%	1400
CIMI	3	60%	800
Total	**15**	**90%**	**3800**

Table II: Use case experiments

Moreover, our experiments show that our approach reduces drastically the user efforts, in terms of code lines, that have been avoided thanks to our simulation designer. Concretely, without our our simulation designer the users may have to write 3800 code lines in total for the 15 use cases in order to simulate them in CloudSim.

VI. Related Work

Simulations tools are essential for carrying out research experiments in cloud computing. In the past decade, many simulation techniques to investigate behavior of cloud computing systems have been developed. In this section, we describe some of them. Since the simulator needs to be included in OCCIware Metamodel, the chosen simulator tool must be (i) compatible with this metamodel in programming languages, (ii) extensible to open new functionalities and (iii) isolate the multi-layer service abstractions (SaaS, PaaS, and IaaS) differentiation.

SimGrid [9] is a generic framework for simulation of distributed applications in Grid platforms. It is mostly written in C and has dependencies with Python and Perl. SimGrid is opensource and can be extensible. From early 2009 up until today, SimGrid has been extended to P2P, HPC and Cloud infrastructures and applications [15].However, SimGrid is limited in memory and suffers on scalability

[16] that impedes the simulation of large-scale systems. In addition, SimGrid is restricted to a single scheduling entity and time-shared systems while there are many space-shared resources needing to be supported in simulation.

GreenCloud [11] is a packet-level simulator that uses the Network Simulator 2 (NS2) libraries for energy-aware data centers. GreenCloud simulator confines its scalability to only small data centers [17] due to very large simulation time and high memory requirements. In addition, GreenCloud is written in C++ and OTcl, two different languages must be used to implement one single experiment. Since GreenCloud is used quite less as compared to other cloud simulators, there is no extension of GreenCloud to our best knowledge.

iCanCloud [12] is a software simulation framework for large storage networks. It can predict the trade-off between costs and performance of a particular application in a specific hardware in order to inform the users about the costs involved. iCanCloud has been developed in C++ on the top of OMNeT++ and INET frameworks. iCanCloud is open source and extensible simulator. New components can be added to the repository of iCanCloud to increase the functionality of the simulation platform. The disadvantages of iCanCloud are firstly, only Cost per Performance (C/P) modeling of cloud computing environments is simulated or validated and secondly, it models and simulates only EC2 (Elastic Compute Cloud) environments.

GridSim [13] provides facilities for the modeling and simulation of resources and network connectivity with different capabilities, configurations, and domains. It is implemented in Java by leveraging SimJavas [18] basic discrete event simulation infrastructure. GridSim is opensource and extensible. Many extensions of GridSim were developed in[19], [20], [21]. Even if GridSim is a powerful simulator, it does not explicitly define any specific application model.

Although the aforementioned toolkits are capable of modeling and simulating cloud application management behaviors, none of them are able to clearly isolate the multi-layer service abstractions (SaaS, PaaS, and IaaS) differentiation required by Cloud computing environments.

CloudSim [10] is an open source simulation application which enables seamless modeling, simulation, and experimentation of cloud computing and application services due to the problem that existing distributed system simulators were not applicable to the cloud computing environment. The CloudSim is implemented at the next level by programmatically extending the core functionalities exposed by the GridSim layer [22]. It provides novel support for modeling and simulation of virtualized cloud-based data center environments such as dedicated management interfaces for VMs, memory, storage, and bandwidth. CloudSim is extensible and many features can be easily added enabling the modeling of new types of applications, not supported by CloudSim.

We have chosen CloudSim as a simulation tool for supporting simulation of OCCI based cloud configuration. This decision was due to its ability to support many more functionalities than other simulation tools, as well as the flexibility in which it was designed for. CloudSim is open source, completely written in Java, and having many extensions being developed and published having it as the bases such as, CEPSim [23], WorkflowSim [24], DynamicCloudSim [25] and CloudAnalyst [26]. Moreover, its layered architectural model makes it easy to understand and differentiate clearly the cloud computing layers service abstractions.

The Table III provides a summary of the critical evaluation provided in this section.

Requirements / Simulator	Extensible	Programming Language	Resources		
			IaaS	SaaS	PaaS
SimGrid [9]	Yes	C	Yes	No	No
GridSim [13]	Yes	Java	Yes	No	No
CloudSim [10]	Yes	Java	Yes	No	Yes
GreenCloud [11]	No	C++, OTcl	Yes	No	No
iCanCloud [12]	Yes	C++	Yes	No	HPC

Table III: Related Work Evaluation Synthesis

VII. Conclusion

OCCIware projet proposes a new precise metamodel for Open Cloud Computing Interface (OCCI) standards. However, this metamodel lacks a simulation environment. Given the importance of cloud simulation, this paper proposes an OCCI extension and a methodology for simulation of OCCI based cloud configuration. The extension defines the representation of CloudSim entities in OCCI context, while the methodology precises how the resources extracted from OCCI configuration are expressed in CloudSim.

The conducted study provides a user-friendly GUI for easily designing and modeling a simulation configuration. This tool is open source, generic, extensible and developed on top of Eclipse IDE.

As future work, we will continuously improve the simulation extension to support other abilities such as network topologies, elasticity, sharing resources policies and include a behavioral study of cloud resources.

Acknowledgements

This work is partially supported by OCCIware, a research project funded by French FSN (Fonds national pour la Societe Numerique) program.

Availability

Open source code for the OCCIware Studio and the Simulator is available at https://github.com/occiware/ecore/tree/master/clouddesigner/

OCCIware Studio binary bundles for Linux, Windows and MacOS are available at http://www.obeo.fr/download/occiware/.

REFERENCES

[1] J. Martin-Flatin, "Challenges in cloud management," *IEEE Cloud Computing*, no. 1, pp. 66–70, 2014.

[2] R. Nyrn, A. Edmonds, A. Papaspyrou, and T. Metsch, "Open Cloud Computing Interface – Core," GFD-P-R.183, April 2011. [Online]. Available: {http://ogf.org/documents/GFD.183.pdf}

[3] A. Edmonds, T. Metsch, A. Papaspyrou, and A. Richardson, "Toward an open cloud standard," *Internet Computing, IEEE*, vol. 16, no. 4, pp. 15–25, July 2012.

[4] P. Merle, O. Barais, J. Parpaillon, N. Plouzeau, and S. Tata, "A precise metamodel for open cloud computing interface," in *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, June 2015, pp. 852–859.

[5] T. Metsch, A. Edmonds *et al.*, "Open cloud computing interface-restful http rendering," in *Open Grid Forum-OCCI Working group technical report*, 2011.

[6] T. Metsch and A. Edmonds, "Open Cloud Computing Interface – Infrastructure," GFD-P-R.184, April 2011. [Online]. Available: {http://ogf.org/documents/GFD.184.pdf}

[7] ——, "Open Cloud Computing Interface – HTTP Rendering," GFD-P-R.185, April 2011. [Online]. Available: {http://ogf.org/documents/GFD.185.pdf}

[8] S. Yangui and S. Tata, "An occi compliant model for paas resources description and provisioning," *The Computer Journal*, p. bxu132, 2014.

[9] H. Casanova, A. Legrand, and M. Quinson, "Simgrid: A generic framework for large-scale distributed experiments," in *Computer Modeling and Simulation, 2008. UKSIM 2008. Tenth International Conference on*. IEEE, 2008, pp. 126–131.

[10] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," in *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on*. IEEE, 2009, pp. 1–11.

[11] D. Kliazovich, P. Bouvry, and S. U. Khan, "Greencloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, 2012.

[12] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente, "icancloud: A flexible and scalable cloud infrastructure simulator," *Journal of Grid Computing*, vol. 10, no. 1, pp. 185–209, 2012.

[13] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and computation: practice and experience*, vol. 14, no. 13-15, pp. 1175–1220, 2002.

[14] H. Brabra, A. Mtibaa, L. Sliman, W. Gaaloul, B. Benatallah, and F. Gargouri, "Detecting cloud (anti)patterns: OCCI perspective," in *Service-Oriented Computing - 14th International Conference, ICSOC 2016, Banff, AB, Canada, October 10-13, 2016, Proceedings*, 2016, pp. 202–218. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46295-0_13

[15] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Simgrid: a sustained effort for the versatile simulation of large scale distributed systems," *arXiv preprint arXiv:1309.1630*, 2013.

[16] S. De Munck, K. Vanmechelen, and J. Broeckhove, "Improving the scalability of simgrid using dynamic routing," in *Computational Science–ICCS 2009*. Springer, 2009, pp. 406–415.

[17] R. Malhotra and P. Jain, "Study and comparison of various cloud simulators available in the cloud computing," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 9, pp. 347–350, 2013.

[18] F. Howell and R. McNab, "Simjava: A discrete event simulation library for java," *Simulation Series*, vol. 30, pp. 51–56, 1998.

[19] R. Albodour, A. E. James, and N. Yaacob, "An extension of gridsim for quality of service." in *CSCWD*, 2010, pp. 361–366.

[20] A. Caminero, A. Sulistio, B. Caminero, C. Carrión, and R. Buyya, "Extending gridsim with an architecture for failure detection," in *Parallel and Distributed Systems, 2007 International Conference on*, vol. 2. IEEE, 2007, pp. 1–8.

[21] J. L. Albın, J. A. Lorenzo, J. C. Cabaleiro, T. F. Pena, and F. F. Rivera, "Simulation of parallel applications in gridsim," in *Proceedings of the Iberian Grid Infrastructure Conference*, 2007, pp. 208–219.

[22] R. N. Calheiros, R. Ranjan, C. A. De Rose, and R. Buyya, "Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services," *arXiv preprint arXiv:0903.2525*, 2009.

[23] W. A. Higashino, M. A. Capretz, and L. F. Bittencourt, "Cepsim: Modelling and simulation of complex event processing systems in cloud environments," *Future Generation Computer Systems*, 2015.

[24] W. Chen and E. Deelman, "Workflowsim: A toolkit for simulating scientific workflows in distributed environments," in *E-Science (e-Science), 2012 IEEE 8th International Conference on*. IEEE, 2012, pp. 1–8.

[25] M. Bux and U. Leser, "Dynamiccloudsim: Simulating heterogeneity in computational clouds," *Future Generation Computer Systems*, vol. 46, pp. 85–99, 2015.

[26] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*. IEEE, 2010, pp. 446–452.