MC-Sim: a mobile cloud simulation toolkit based on CloudSim

Manel Gherari*
LAMIS Laboratory,
Department of Computer Science,
University of Tebessa,
Tebessa 12000, Algeria
Email: gherari.manel@yahoo.fr
*Corresponding author

Abdelkrim Amirat
LIM Laboratory,
Department of Computer Science,
University of Souk-Ahras,
Souk-Ahras 41000, Algeria
Email: abdelkrim.amirat@yahoo.com

Mohamed Ridda Laouar
LAMIS Laboratory,
Department of Computer Science,
University of Tebessa,
Tebessa 12000, Algeria
Email: ridda_laouar@yahoo.fr

Mourad Oussalah
LINA Laboratory,
Department of Computer Science,
University of Nantes,
Nantes 44322, France
Email: mourad.oussalah@univ-nantes.fr

Abstract: Mobile Cloud Computing (MCC) has gained a significant attention these past years. MCC consists of migrating mobile applications from the constrained mobile devices to the cloud. This task is highly complicated and demanding, therefore several novel methods, tools, and approaches are introduced to tackle this complexity. At this point, we argue that a simulation of the deployment mechanisms for accessing cloud services and testing mobile cloud applications in the cloud environment is a mandatory phase prior to real deployment in a real environment. A simulation will offer the developer a controllable and cost-free environment to test and evaluate applications’ performance according to different predefined scenarios. As MCC lacks tools of simulation of its aspects, to fill this gap we propose Mobile Cloud Simulation (MC-Sim) toolkits based on CloudSim.

Keywords: CloudSim; mobile cloud computing; cloud simulation; mobile cloud simulation; mobile cloud computing modelling.


Biographical notes: Manel Gherari received her PhD degree in Computer Science from the University of Tebessa, Algeria. She received her master degree in Software Engineering from the University of Souk-Ahras in 2013. Her research field is mainly focused on software architecture design and evolution, mobile computing, and cloud computing.

Abdelkrim Amirat received his PhD degree in Computer Science from the University of Nantes, France, and his Habilitation in 2010. Currently he is a Professor of Computer Science at the University of Souk-Ahras, Algeria. He is the director of Mathematics and Computer Science.
1 Introduction

Cloud Computing (CC) offers cloud services at different levels (infrastructure as service, platform as service, and software as services) to consumers that already have established a Service Level Agreement contract based on certain negotiations with the service providers. The provisioning of cloud services is done following a payment model (pay as you go). Some well-known clouds are Amazon EC2, Google App engine, Aneka, Microsoft Azure.

By migrating to the cloud, IT companies are no longer concerned by tasks such as setting up basic hardware and software infrastructure, so they can focus on innovation and creation of business values for their applications (Armbrust et al., 2010). It should be noted that cloud based applications differ from each other in term of composition, configuration, deployment requirements. Thus testing this kind of applications’ performance, in terms of provisioning policies (scheduling, allocation), is a highly demanding task due to challenges posed by the cloud like the fact the users have heterogenous, dynamic, ever changing QoS requirements.

Using real cloud infrastructure like Amazon EC2 or Microsoft Azure for benchmarking application performance under variable conditions is mainly constrained by the rigidity of real cloud infrastructure (Calheiros et al., 2011).

Motivated by this limitation, a simulated, controllable, and cost free environment is a necessity to any developer before heading to real deployment of its application in real cloud environment. To give researchers a testbed and platform to assist their research and validate their idea, several cloud simulators have been proposed, and many simulation tools dedicated for cloud based applications have been proposed (Calheiros et al., 2011; Buss, 2002; Buyya and Murshed, 2002; Varga, 2001).

Mobile Cloud Computing (MCC) has been introduced in order to enhance the performance of mobile devices that have invaded every personal life these days. Thus, mobile users are seeking PC-like functionalities provided by Smartphone and to answer these issues, MCC outsource the processing and the storage of data from the constrained mobile devices to the elastic cloud infrastructure. As a paradigm still in its infancy, MCC unlike CC lacks tools for modelling and simulation it aspects. To fill this gap we propose Mobile Cloud Simulation toolkits based on CloudSim (Calheiros et al., 2011) (MC-Sim).

MC-Sim is an extension of CloudSim that adds more packages dedicated to simulate mobile entities such as mobile devices, mobile applications, and other sub-entities needed during the simulation. The main difference from CloudSim, is that MC-SIM is designed in a manner that separates the modelling phase from the simulation phase. First the mobile cloud application architecture is described via MC-ADL (Gherari et al., 2014a). Mobile cloud application architecture ensures the production of applications of quality and has a major impact on its cost (Dhaya and Zayaraz, 2016; Huang et al., 2016). Based on the latter, simulation entities are dynamically generated. The next step consists of executing the simulation by either defining new allocation/scheduling policies, or using the already predefined ones. MC-Sim ensures a high performance of tests that are open to scalability since they are predefined in well-specified order to be executed dynamically.

The remainder of this paper is organised as follows: Related work presenting cloud simulators will be given in Section 2. Section 3 will introduce MC-SIM. A case study is given in Section 4. Finally, we conclude this paper in Section 5.

2 Related work

In this section, we will summarise some cloud computing simulators.
2.1 CloudSim

CloudSim (Calheiros et al., 2011) is a new generalised and extensible simulation framework that is characterised by modelling and simulating of large scale infrastructure including data centres (DCs) that host several Virtual machines (Vms). Vms are allocated according to different (Space/Time) share allocation policies, with the possibility to customise policies according to the user’s requirements.

The first layer SimJava is a discrete event simulation engine that implements core functionalities, like queuing and processing of events, and creation of systems components (Service, host, DCs, broker and Vms). Next layer is Gridsim that regroups libraries supporting modelling multiple grid infrastructure, including networks, association traffic, resources data sets, workload traces, and information services. As an extension to Gridsim, CloudSim allows modelling and simulation of virtualised cloud based DCs’ environment, such as dedicated management interfaces for Vms, memory, storage, and bandwidth. As a third layer, CloudSim is dedicated for the instantiation and execution of core entities (Vms, hosts, and DCs). The final layer is the user core that consists of the configuration of related functionalities hosts such as the number of tasks and the user requirements.

Several works have been proposed to extend and improve CloudSim, by focusing on reducing and eliminating failures that occur during job submission (Belalem et al., 2010). Another extension of CloudSim as shown in Figure 2 is Cloud Analyst (Wickremasinghe et al., 2010; Wickremasinghe, 2009); this toolkit is designed to study the behaviour of large scale internet applications in cloud environment while offering the possibilities to perform different simulation experiments, by simply changing parameters.

Garg and Buyya (2011) proposed NetworkCloudSim extension of CloudSim which supports modelling real time DCs and generalised applications like e-commerce. The architecture of NetworkCloudSim is presented in Figure 3.

2.2 SPECI

Motivated by the need to predict the performance and the behaviour of future DCs, Sriram (2009) proposed SPECI (Simulation Program for Elastic Cloud Infrastructure). SPECI is simulation toolkit based in SimKit (Buss, 2002) that allows exploration of aspects of scaling as well as performance proprieties of future DCs, by offering two main packages: one representing DCs, layout and topologies, and the other encompassing the components for experiments execution and measuring.
2.3 iCanCloud

iCanCloud (Núñez et al., 2012) is a simulator for cloud infrastructure that promises different features such as flexibility, scalability, performance and usability. iCanCloud is designed targeting the following objectives:

- Aims to conduct large scale experiments
- Provides flexible and fully customised global hypervisor for integrating any cloud brokering policy
- Recreates instance type provided by cloud infrastructure
- Offers GUI interface for configuration and launching simulation.

Distinguished from other simulation toolkits, iCanCloud offers a GUI interface, and also tackles the deficiency of performing a paralleled simulation across multiple machines.

As presented in Figure 4, the first layer of the iCanCloud architecture is hardware layer that contains models, which are in charge of modelling the hardware part (e.g., disc drive, CPU, etc.). In order to guarantee a certain degree of compatibility, API is used to create an application that enters the simulation with iCanCloud. Next layer is Vm repository that contains a collection of Vms that have been defined by users. Cloud Hypervisor consists of models responsible for managing all jobs instance of Vms, and cost policies. Cloud System Module encompasses the definition of the entire cloud system by defining hypervisor and Vms.

2.4 GreenCloud

Based on NS2 (https://www.isi.edu/nsnam/ns/), GreenCloud (Kliazovich et al., 2012; Liu et al., 2009) is packet level cloud simulator that mainly focuses on evaluating the energy cost of DCs, operation. The GreenCloud architecture is presented in Figure 5.
2.5 Open Cirrus

Established by Yahoo and HP, Open Cirrus (Avetisyan et al., 2010; Campbell et al., 2009) provides a testbed heterogeneous distributed DC for systems, applications and services. Open Cirrus aims to foster system level research in cloud computing, encouraging new cloud application research by offering a platform for real world application, and providing a collection of experimental data in order to support researchers with their high quality experiments.

Other than cloud computing, network computing also has its share of simulation tools that are interested in network details, network protocol, and path discovery like: DaSSF (Liu and Nicol, 2001), OMNeT++ (Varga, 2001; Varga and Horning, 2008), OPNet (Chen, 2004; Chang, 1999) and NS2 (https://www.isi.edu/nsnam/ns/).

3 Proposed approach

Mobile cloud computing has gained the interest of researchers this past few years, as mobile devices have invaded our lives by being a necessity and no longer a luxury. So in order to provide PC-like functionalities to mobile users, merging the cloud and the mobile computing was the important step to brighter future in mobile applications’ industry.

We argued in previous work (Gherari et al., 2014a) that mobile cloud applications need an architecture representation as any other system. In this paper and following the development cycle of mobile cloud applications we believe that a simulation of an behaviour of the application in cloud environment is indispensable before heading to real deployment. At this phase, the developer can test repeatedly the application in a simulated, controllable and cost-free environment, and ensure a pane/bug free application smoothly deployed in real mobile cloud environment.

After studying several simulations tools, we have come to choose CloudSim to extend as the more adequate and highly corresponding toolkit to our vision. We mainly implemented our work in Java language and the simulation in our case is based on mobile cloud architecture model described by MC-ADL (Gherari et al., 2014a) Eclipse plug-in.

CloudSim does not offers classes to simulate mobile entities since it is not developed for mobile cloud applications simulations. Addressing this limit, we have added new simulating packages to CloudSim dedicated to mobile cloud application simulation. These latter regroup classes that are defined in the same logic of CloudSim entities by abstracting the mobile cloud elements and their behaviours. Figure 6 shows the MC-Sim layered architecture based on CloudSim. MC-App Cloudlet encapsulates all the relations between the application task and the provided cloud service. Mobile Device encapsulates the information concerning the mobile device starting from its operating systems to its intrinsic and external contextual information.

It should be noted that we are interested in taking advantage of contextual information provided by the cloud and the mobile environments. This kind of information is presented in the architecture model and will be mapped later in the simulation process. Concerning the cloud, its contextual information encompasses type cloud model, type of cloud service, cloud provider, availability, QoS, payment model, etc.

Mobile contextual information is divided into two classes:

- **Intrinsic**: regroups the permanent information as the characteristic of the mobile device (e.g. OS, memory, storage capacity, processing power).
- **External**: encompasses environment information like location, time, neighbour devices, and network connection.
Based on the architecture model (see Figure 7) representing the Mobile Cloud Application (MC-App) a simulator class is a Java program that will exploit the information in the MC-App model, more specifically its xml representation, to create extended entities corresponding the mobile and cloud elements that answer the characteristics specification defined by the developer during the design phase. Some entities like cloud broker, allocation policies, and scheduling are created dynamically since they are predefined entities, but they can be customised later according to the developer’s requirements.

As presented in the activity diagram (see Figure 8), after generating the object representing the xml version of the architecture model, the latter will be an input to the second activity that takes it as parameter to create the two objects representing cloud entities and mobile entities. The developer can change the specification of the simulated entities according to requirements.

To foster the concept of dynamic architecture evolution, we manipulate the execution of the simulation dynamically by running several simulation scenarios that add or delete some cloud entities, which will provoke an evolution at the architecture level.

During the simulation phase an important question must be asked is: Which mobile application task needs to be offloaded to the cloud and which one needs to be executed locally? Many researchers (Flores and Srirama, 2013; Kumar and Lu, 2010), have tried to answer this question. Thus, in our proposed framework we have not implemented an offloading decision algorithm, assuming that every task of the mobile cloud application will be delegated/offloaded to the cloud according to predefined or customised allocation policies in CloudSim.
4 Case study

To evaluate the operating mechanism of contextual information, which we propose via Smart Cloud Gate (Gherari et al., 2014b), we present in this section a case study to investigate the behaviour of MC-App according to defined changes in the cloud context.

4.1 Shopping Assistant: smart mobile cloud application

Shopping Assistant is an application that allows the user to view the stores and visit them virtually using the mobile device without actually entering the store. This application offers the possibility to have a list of all the stores in the area, paths to follow to visit them, and keep favourite items to check later.

The architecture of Shopping Assistant defined by MC-ADL in Figure 9 shows that the application is deployed in a smartphone Samsung galaxy Edge 7 device, the internal memory capacity of the mobile device is 32GB, RAM of 4GB (intrinsic contextual information of the mobile device). The application offers several features, that are executed in the cloud using Vms allocated in data centre 1. Note that the service used is provided in a public cloud platform.

MC-ADL aims to provide a comprehensive perspective on the application and its cloud and mobile environment. We note that in the proposed architecture, Cloud’s details are defined at Data Centre Characteristics context, but details on the application and Cloud will be given during the simulation, as some concepts are not taken into consideration when designing the application to build generic reusable entities. Therefore, separation of the architecture from the implementation details is needed.

The implementation of Shopping Assistant is done as a Java prototype to test its behaviour with CloudSim. The proposed code can be deployed in the Android environment easily. Functionalities of Shopping Assistant are described as Java classes presented in Table 1.

4.2 Analysing Shopping Assistant behaviour in cloud environment

One of the biggest downfalls of many mobile applications is their real or perceived response time. The importance of optimising the experience is to give the user the data they want as fast as possible or somehow distract them to retain their attention longer. Mobile users these days expect results in seconds. The expected response time of an application should not exceed one second, thus the response time of a Cloud based application not only depends on the application, but also on error rates and latency of the Cloud service.

One of the main objectives of the intertwining of Mobile Computing with Cloud Computing paradigm was to respond to performance issues for mobile applications. Therefore, with each proposed solution several questions arise. For example, the delegation of mobile functionality to the Cloud will improve application performance in terms of energy, but will infect the response time as explained above.
Table 1  Shopping assistant’s Java classes functionalities

<table>
<thead>
<tr>
<th>Classes</th>
<th>Functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-Account</td>
<td>To collect, organise user’s information (Id, name, email address, etc.), and save it in CSV file</td>
</tr>
<tr>
<td>Check-In</td>
<td>To identify users and access data, it also saves users preferences to use it later in recommendation class</td>
</tr>
<tr>
<td>Stores</td>
<td>Represents stores and save their information (i.d. name, address, location)</td>
</tr>
<tr>
<td>Article</td>
<td>Define the articles available in the stores: URL, ID, Image, Description, Price</td>
</tr>
<tr>
<td>Wish-List</td>
<td>A class that organises the articles selected by the user and the information stores that offer it</td>
</tr>
<tr>
<td>Up-Date-Data</td>
<td>A class that provides the basic update given based on rental change and the addition of new stores</td>
</tr>
<tr>
<td>Main-Class</td>
<td>The main class that implements the store search, the location and path display, and also the display of items available</td>
</tr>
</tbody>
</table>

The purpose of this case study is to prove that our hypothesis:

“Cloud must be intelligently exploited to improve the performance of mobile applications without having a negative effect and open the door to other concerns.”

We will study the performance of the application using a predefined cloud service and study its behaviour by stimulating its context awareness by introducing a change in the context Cloud. The second case aims to study how the application will take advantage of this change to optimise its response time.

Table 2  MC-AppCloudLet functionalities

<table>
<thead>
<tr>
<th>MC-AppCloudLet</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Save stores’ data</td>
</tr>
<tr>
<td>C2</td>
<td>Save the user data</td>
</tr>
<tr>
<td>C3</td>
<td>Save data items</td>
</tr>
<tr>
<td>C4</td>
<td>Classify items according to price</td>
</tr>
<tr>
<td>C5</td>
<td>Classify according stores closest to farthest from the user in its current position.</td>
</tr>
<tr>
<td>C6</td>
<td>Update databases</td>
</tr>
<tr>
<td>C7</td>
<td>Search for the nearest shops</td>
</tr>
<tr>
<td>C8</td>
<td>Establish the preference list</td>
</tr>
<tr>
<td>C9</td>
<td>Post the nearest shops and the way forward to visit</td>
</tr>
</tbody>
</table>

Application as presented in its architecture (see Figure 9) shows its initial state where it uses a single service Cloud “Data Center 1” to which it delegates all of its tasks. Representation of the internal structure of the Data Centre as described by CloudSim is given in Figure 10.

Figure 9  Shopping assistant architecture

![Shopping assistant architecture](image)

Figure 10  Data centre structure

![Data centre structure](image)
The Data Centre used consists of a single host running three virtual machines to which are assigned three MC-AppCloudLet. The characteristics of the data centre (see listing 1) are saved in Cloud Information Service (CIS).

Listing 1  Data centre characteristics

```java
// S. Create a datacenterCharacteristics object that stores the
// properties of a data center: architecture, OS, list of:
// Machine, allocation policy: time or space-claimed, time zone
// and its price (US$) time unit.
String str = "three hosts architecture time zone = 10.8;
// the time zone this resource located
double cost = 5;
// the cost of using processing in this resource in dollars
double costStorage = 10;
// the cost of using storage in this resource in dollars
double cost = 0.8;
// the cost of using memory in this resource in dollars
```

CloudBroker assigns each MC-AppCloudLet the available virtual machines, after checking some constraints assignment. Each MC-AppClouLet is responsible for performing a function of Shopping Assistant as explained in Table 1.

In the contract associated with the application there is a term indicating the distributed execution on multiple services (data centres) if they exist. So running scenario ‘1’ follows the dynamic addition of a new DataCenter ‘2’ (see Figure 11), which causes the activation of the class “StartArchitecturalEvolution” in the package “Smart Cloud Gate”.

The “Smart Cloud Gate” package contains Java classes that establish a relationship between the simulation level and the architectural level. The order of execution and the role of each class are shown and explained as following.

Figure 11  Simulation scenario

Figure 12  Shopping assistant new version architecture
The Contract-Reader class parses the XML file representing the contract to extract contextual information. The result of this function is a set of parameters that will be sent to the Evol-Param class to set the parameter model that comes as a second model in execution. The parameter model is a smaller version of the contract model that includes the necessary information to the architectural evolution. Finally, the Architecture-Evolution class executes the evolution policy that corresponds to the addition of a service. In this case, the service is provided by the same supplier in the same platform (see Figure 12).

4.3 Results discussion

A term in the MC-App Contract indicates the use of services available to share the execution of application components. The objective to define this term is to optimise response time to improve the performance of Shopping Assistant. We have verified that the scenario ‘1’ promotes the allocation of VMs by following “TimeSharedAllocationPolicies” where the time allocated to each MC-App Cloudlet is based on number of instruction (each instruction is estimated to take only one millisecond to be executes), but the order of execution is organised sequentially. Adding a new Data Centre will set new allocation parameter that will allow the execution of some Cloudlet simultaneously to save time.

The initial state 0 of the application corresponds to the sequential execution of Cloudlet taking $673\text{ ms}$ as total execution time (i.e. response time). As for the state 1 in which the MC-Cloudlets (C1-C7, …) are running simultaneously, has caused a remarkable decrease of the response time, to $265\text{ ms}$.

We see from the results in Figure 13 that the addition of a new service has improved the application performance significantly. The intelligent use of cloud based contract model was beneficial. Without the contract model, which models a kind of consciousness in the cloud, the application will continue to use the same service, ignoring the opportunity to exploit the novelty that can occur in the cloud.

Figure 13 Analytic study of response time MC-App CloudLets

5 Conclusion

We have presented in this paper a brief state of the art of cloud computing simulation tools, highlighting the lack of simulators for mobile cloud computing. We argue that the simulation offers means to test mobile applications according to different scenarios that can predict the application evolution during its life cycle (Cherait and Bounour, 2016). Motivated by this limit we have proposed Mobile Cloud Simulation toolkit based on CloudSim (MC-Sim). During the design of MC-Sim, we have settled the following objectives:

- Exploit the generic, independent mobile cloud architecture model described via MC-ADL (Gherari et al., 2014a), to dynamically create simulation entities.
- Foster a highly efficient simulation process by defining several simulation scenarios that run dynamically in certain order to fulfill any test requirements.
- Simulating mobile cloud environment changes in terms of adding, modifying, or deleting cloud resources or mobile resources to foster a certain degree of adaptability of mobile cloud application.

In this first prototype of MC-SIM, the mobile cloud application task is directly assigned to be executed by a cloud service; we plan as future work to implement a decision algorithm to decide when and how to offload/delegate certain mobile cloud application tasks to the cloud.

References


