OCCIware is a project funded by the French FSN (Fonds national pour la Société Numérique), and supported by five clusters: Systematic, Minalogic, PICOM, Images & Réseaux, Solutions Communicantes et Sécurisées.
<table>
<thead>
<tr>
<th>Project Title</th>
<th>OCCIware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliverable Number</td>
<td>3.4.1</td>
</tr>
<tr>
<td>Deliverable Title</td>
<td>Cloud Computing Simulators: State of the Art</td>
</tr>
<tr>
<td>Deliverable Nature</td>
<td>Report</td>
</tr>
<tr>
<td>Level of dissemination</td>
<td>Public</td>
</tr>
<tr>
<td>License</td>
<td>Creative Commons Attribution 3.0 License</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Publication date</td>
<td>September 2015</td>
</tr>
<tr>
<td>Sub-Project</td>
<td>SP2</td>
</tr>
<tr>
<td>Editor(s)</td>
<td>Mehdi Ahmed-Nacer, Sami Bhiri, Samir Tata (TSP)</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Mehdi Ahmed-Nacer, Sami Bhiri, Samir Tata (TSP)</td>
</tr>
<tr>
<td>Reviewer(s)</td>
<td>Philippe Merle (Inria), Nabil Bachir Djarallah (Scalair)</td>
</tr>
</tbody>
</table>

**Abstract**

Cloud Computing is an emerging paradigm in Information Technologies (IT). One of its major assets is the provisioning of resources based on pay-as-you-go model. Cloud resources are situated in a highly dynamic environment. However, each provisioned resource comes with functional properties and may not offer nonfunctional properties like monitoring, reconfiguration, security, accountability, etc. In such dynamic environment, it would be difficult (if not impossible) to precisely predict the behavior of deployed cloud resources and services. Therefore, simulation technology becomes increasingly popular in the cloud industry and academy. It allows user to evaluate their algorithms and applications before running them on a real Cloud. Within the OCCIware project we plan to develop an extensible simulator for cloud services and resources described in OCCI. With this objective in mind, this deliverable presents a short survey of Cloud simulators and evaluates their suitability to be adopted and possibly extended for use in the OCCIware project.

**List of key-words**

OCCI, OCCIware, Simulation, distributed computing, cloud computing, SimGrid, CloudSim.
### Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Changes</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Cloud Computing Simulators: State of the Art</td>
<td>Mehdi Ahmed-Nacer, Sami Bhiri, Samir Tata (TSP)</td>
</tr>
<tr>
<td>0.2</td>
<td>Add simulation resources and extensions part</td>
<td>Mehdi Ahmed-Nacer, Sami Bhiri, Samir Tata (TSP)</td>
</tr>
<tr>
<td>0.3</td>
<td>Figures and Tables updates</td>
<td>Mehdi Ahmed-Nacer, Sami Bhiri, Samir Tata (TSP)</td>
</tr>
<tr>
<td>1.0</td>
<td>Final version</td>
<td>Mehdi Ahmed-Nacer, Sami Bhiri, Samir Tata (TSP)</td>
</tr>
</tbody>
</table>

### Document review

<table>
<thead>
<tr>
<th>Review</th>
<th>Date</th>
<th>Version</th>
<th>Reviewers</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>01/09/2015</td>
<td>0.1</td>
<td>Philippe Merle</td>
<td></td>
</tr>
<tr>
<td>Draft</td>
<td>14/10/2015</td>
<td>0.2</td>
<td>Philippe Merle</td>
<td></td>
</tr>
<tr>
<td>Review</td>
<td>15/10/2015</td>
<td>0.3</td>
<td>Philippe Merle</td>
<td>Nabil Bachir Djarallah</td>
</tr>
<tr>
<td>Final</td>
<td>25/10/2015</td>
<td>1.0</td>
<td>Philippe Merle</td>
<td></td>
</tr>
</tbody>
</table>
# Table of Contents

1. Introduction.......................................................................................................................... 1
2. Cloud Simulator’s Overview ................................................................................................. 2
   2.1. SimGrid ............................................................................................................................ 2
   2.2. GreenCloud .................................................................................................................... 4
   2.3. CloudSim ......................................................................................................................... 6
   2.4. iCanCloud ..................................................................................................................... 6
3. CloudSim .................................................................................................................................. 7
4. Synthesis and objectives ........................................................................................................ 10
5. Conclusion ............................................................................................................................. 11
Bibliography.................................................................................................................................. 17

# Table of Figures

Figure 1: SimGrid components................................................................................................. 2
Figure 2: Architecture of the GreenCloud simulation environment........................................ 5
Figure 3: Layered CloudSim architecture................................................................................ 9

# Table of Tables

Table 1: Simulator synthesis ..................................................................................................... 10
1. Introduction

Cloud computing technology is been developed quickly in recently years. Its computing model enables ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) [15]. With the very attractive features, cloud computing is being widely applied in industry community, businesses, consumers and government organizations, and also obtained huge attention in academy community. In fact, for instance, in a Mckinsey Quarterly survey\(^1\) conducted in 2010 on 332 companies, 75% believe that the use of cloud computing could drive value at their companies. Among these companies, 68% say that they are currently adopting the cloud to set up electronic collaboration and 82% are planning to do it in the 18 coming months. In this new world of business, electronic cooperation, collaboration and/or federation are inevitable.

Along with the research, development and application of cloud computing, it becomes more and more important to study and analyze how cloud computing and the applications which are deployed on clouds will perform. But porting an application over a Cloud platform and evaluating it at scale comes with a high cost. Resources have to be rent for the time needed to test and run the application. Furthermore, if a user wants to evaluate different strategies/algorithms, he/she needs to repeat his/her experiments for a period of time that can be quite long. Researchers working on new models for Cloud applications, elastic resource management functions, provisioning algorithms, etc., are faced to the same issue. They need to pay Cloud provider such for resources they use during development and evaluation processes. If scientists and application developers want to optimize their applications, they will pass through numerous fail runs before finding the good trade-offs between multiple parameters such as instance and storage types. Moreover, when testing different strategies, they are not sure that the underlying platform is the same. Thus they cannot do reproducible experimentations to compare different strategies on such Cloud. The noise coming from the platform itself and the results obtained may be wrong.

Therefore, simulation technology becomes increasingly popular in the cloud industry and academy. It allows users to evaluate their algorithms and applications before running them on real Cloud environments. This solution gives many advantages such as:

- **No capital cost involved**: cloud computing makes a shift from capital expenditure cost to operational cost. Having a cloud simulation tool also involves having no installation cost or maintenance cost related to the application to be simulated as well.

- **Better results**: Using simulators helps to change inputs and other parameters as well very easily, which results in better and efficient output.

- **Evaluation of risks at an early stage**: Because simulation tools involve no cost while running as is in case of being on cloud, so users can identify and solve any risk that is associated with the design or with any parameter.

- **Low cost of learning**: While working with simulators, users need to have only programming abilities and rest all depend on that. If the user is well versed with

programming language, then simulation tools offer facilities with low cost of learning.

In this deliverable, we present an overview of some popular cloud simulators by highlighting the important characteristics of each one of them. This deliverable is organized as follows. Section 2 presents a state of the art survey of some existing cloud computing simulators. Section 3 details more the most adapted simulator to our platform. Section 4 presents our objectives related to cloud simulation in the OCCIware project. Section 5 concludes this deliverable and presents our future work.

2. Cloud Simulator's Overview

To develop and analyze any new cloud environment with the help of simulators, it is required to understand the existing cloud simulators with their pros and cons. Some of the published cloud computing simulators for evaluating cloud computing systems performance are described in this section. We cite SimGrid [1], GreenCloud [10], CloudSim [7] and iCanCloud [5].

2.1. SimGrid

SimGrid [1, 11] is a scientific instrument to study the behavior of large-scale distributed systems such as Grids, Clouds, HPC or P2P systems. It can be used to evaluate heuristics, prototype applications or even assess legacy MPI applications. SimGrid was conceived as a scientific instrument, thus the validity of its analytical models was thoughtfully studied [2], ensuring their realism. The key features of SimGrid are:

- A scalable and extensible simulation engine that implements several validated simulation models, and that makes it possible to simulate arbitrary network topologies, dynamic compute and network resource availabilities, as well as resource failures;
- High-level user interfaces for distributed computing researchers to quickly prototype simulations either in C or in Java;
- APIs for distributed computing developers to develop distributed applications that can seamlessly run in simulation mode "or in real-world mode."

SimGrid is composed of five modules as it can be seen in Figure [1].

Figure 1: SimGrid components.
1. **MSG** module provides an API for the easy prototyping of distributed applications by letting users focus solely on algorithmic issues. Simulations are constructed in terms of concurrent processes, which can be created, suspended, resumed and terminated dynamically, and can synchronize by exchanging data. Moreover it proved perfectly usable in other contexts, such as desktop grids [2]. An important difference between MSG and SMPI is that all simulated processes are in the same address space. This simplifies development of the simulation by allowing convenient communications via global data structures. In other words, while MSG can accurately simulate the interactions taking place in a distributed application, including communication and synchronization delays, the simulated application can be implemented with the convenience of a single address space.

2. **SMPI** is a simulator for MPI applications. Its first version provided only on-line simulation, i.e., the application is executed but part of the execution takes place within a simulation component. SMPI simulations account for network contention in a fast and scalable manner. SMPI also implements an original and validated piece-wise linear model for data transfer times between cluster nodes. Finally SMPI simulations of large-scale applications on large-scale platforms can be executed on a single node thanks to techniques to reduce the simulation’s compute time and memory footprint.

3. **SimDag** is designed for the investigation of scheduling heuristics for applications as task graphs. SimDag allows to prototype and simulate scheduling heuristics for applications structured as task graphs of (possibly parallel) tasks. With this API one can create tasks, add dependencies between tasks, retrieve information about the platform, schedule tasks for execution on particular resources, and compute the DAG execution time.

4. **SURF** considers the platform as a set of resources, with each of the simulated concurrent tasks utilizes some subset of these resources. SURF computes the fraction of power that each resource delivers to each task that uses it by solving a constrained maximization problem: allocate as much power as possible to all tasks in a way that maximizes the minimum power allocation over all tasks, also called Max-Min fairness. SURF provides a fast implementation of Max-Min fairness. Via Max-Min fairness, SURF enables fast and realistic simulation of resource sharing (e.g., TCP flows over multi-link LAN and WAN paths, processes on a CPU). Furthermore, it enables trace-based simulation of dynamic resource availability. Finally, simulated platform configurations are described in XML using a syntax that provides an unified abstract basis for all other components of the SIMGRID project (MSG and SMPI).

5. **XBT** is a toolbox module used throughout the software, which is written in ANSI C for performance. It implements classical data containers, logging and exception mechanisms, and support for configuration and portability.

The GridSim toolkit supports modeling and simulation of a wide range of heterogeneous resources, such as single or multiprocessors, shared and distributed memory machines such as PCs, workstations, SMPs, and clusters with different capabilities and configurations. It can be used for modeling and simulation of application scheduling on various classes of parallel and distributed computing systems such as clusters, grids and P2P networks.

GridSim provides facilities for the modeling and simulation of resources and network connectivity with different capabilities, configurations, and domains. It supports primitives for application composition, information services for resource discovery, and interfaces for assigning application tasks to resources and managing their execution. These features can be used to
simulate resource brokers or Grid schedulers for evaluating performance of scheduling algorithms or heuristic.

SimGrid is extensible and much functionality can be added easily. For instance, Data Grid [1] is an extension of GridSim. With the addition of this extension, GridSim has the ability to handle core Data Grid functionalities, such as replication of data to several sites, query for location of the replicated data, access to the replicated data and make complex queries about data attributes.

Overview of other GridSim extensions functionalities [9]:

- incorporates failures of Grid resources during runtime.
- incorporates a functionality that reads workload traces taken from supercomputers for simulating a realistic grid environment.
- incorporates an auction model into GridSim.
- incorporates a network extension into GridSim. Now, resources and other entities can be linked in a network topology.
- incorporates a background network traffic functionality based on a probabilistic distribution. This is useful for simulating over a public network where the network is congested.

One of the main issues of SimGrid is a scalability problem [4] that impedes the simulation of large-scale systems. Currently, the size of the simulated networks is mainly limited by system memory. In fact, his memory limit is reached at roughly 4000 hosts on a machine with 8 GB of memory. Although scaling beyond this limit is possible through virtual memory, these results in extensive swapping which cripples the simulator’s performance. The large memory requirements of SimGrid are due to the memory-intensive manner of storing the routing information that is used in the simulated network. Another but less important problem is due to the structure of the platform files that describe the network topology in SimGrid. These files are larger than necessary, resulting in significant startup costs of the simulation and decreased manageability of those files.

2.2. GreenCloud

GreenCloud [5] is an extension to the network simulator NS2 [6] developed for the study of cloud computing environments. The GreenCloud offers users a detailed fine-grained modeling of the energy consumed by the elements of the data center, such as servers, switches, and links. Moreover, GreenCloud offers a thorough investigation of workload distributions. Furthermore, a specific focus is devoted to the packet-level simulations of communications in the data center infrastructure, which provides a finest-grain. GreenCloud aims at:

- developing high-end computing systems such as Clusters, Data Centers, and Clouds that allocate resources to applications hosting Internet services to meet users' quality of service requirements,

- minimizing consumption of electric power by improving power management, dynamically managing and configuring power-aware ability of system devices,

- providing a detailed simulators, analyzing energy efficiency and measure cloud performance.
Figure 2 presents the structure of the GreenCloud extension mapped onto the three-tier data center architecture.

Figure 2: Architecture of the GreenCloud simulation environment

- **Servers** ($S$) are the staple of a data center. They are responsible of task execution. In GreenCloud, the server components implement single core nodes that have a preset on a processing power limit, associated size of memory/storage resources, and different task scheduling.

- **Switches and Links** form the interconnection fabric that delivers workload to any of the computing servers for execution in a timely manner. The interconnection of switches and servers requires different cabling solutions depending on the supported bandwidth, physical and quality characteristics of the link. The quality of signal transmission in a given cable determines a tradeoff between transmission rate and the link distance, which are the factors defining the cost and energy consumption of the transceivers.

- **Workloads** are the objects designed for universal modeling of various cloud user services, such as social networking, instant messaging, and content delivery. The execution of each workload object requires a successful completion of its two main components: computational and communicational.

There are basically four main entities involved [3]:

1. Consumers/Brokers: Cloud consumers or their brokers submit service requests from anywhere in the world to the Cloud. It is important to notice that there can be a difference between Cloud consumers and users of deployed services. For instance, a consumer can
be a company deploying a Web application, which presents varying workload according to the number of "users" accessing it.

2. Green Resource Allocator: Acts as the interface between the Cloud infrastructure and consumers. It requires the interaction of the following components to support energy-efficient resource management,

3. Vms: Multiple VMs can be dynamically started and stopped on a single physical machine to meet accepted requests, hence providing maximum flexibility to configure various partitions of resources on the same physical machine to different specific requirements of service requests. Multiple VMs can also concurrently run applications based on different operating system environments on a single physical machine. In addition, by dynamically migrating VMs across physical machines, workloads can be consolidated and unused resources can be put on a low-power state, turned off or configured to operate at low-performance levels (e.g., using DVFS) in order to save energy.

4. Physical Machines: The underlying physical computing servers provide hardware infrastructure for creating virtualized resources to meet service demands.

The drawback of the GreenCloud Simulator is that it confines its scalability to only small data centers [4] 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 less used compared to other cloud simulators, there is no extension of GreenCloud to our best knowledge.

2.3. CloudSim

CloudSim [7] 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. It was developed in the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the Computer Science and Software Engineering Department of the University of Melbourne.

The CloudSim toolkit supports system and behavior modeling of cloud system components such as data centers, virtual machines (VMs) and resource provisioning policies. It implements generic application provisioning techniques that can be extended with ease and limited efforts. CloudSim helps the researchers and developers to focus on specific system design issues without getting concerned about the low level details related to cloud-based infrastructures and services. In the next section, we detail more about this simulator.

2.4. iCanCloud

iCanCloud [5] is a software simulation framework for large storage networks. iCanCloud can forecast the accommodation between price paid and performance of a particular application in a specific hardware in order to inform the users about the costs involved. It focuses on policies, which charge users in a pay-as-you-go manner. iCanCloud has a full graphical user interface from which experiments can be modeled and run, but existing systems can only be modeled manually. It also allows parallel execution of one experiment over several
machines. This simulation framework has been developed on the top of OMNeT++ and INET frameworks.

The most remarkable features of the iCanCloud simulation platform include the following:

- Both existing and non-existing cloud computing architectures can be modeled and simulated,
- A flexible cloud hypervisor module provides an easy method for integrating and testing both new and existent cloud brokering policies. Customizable VMs can be used to quickly simulate uni-core/multi-core systems,
- iCanCloud provides a wide range of configurations for storage systems, which include models for local storage systems, remote storage systems, like NFS, and parallel storage systems, like parallel file systems and RAID systems,
- iCanCloud provides a user-friendly GUI to ease the generation and customization of large distributed models. This GUI is especially useful for: managing a repository of pre-configured VMs, managing a repository of pre-configured Cloud systems, managing a repository of pre-configured experiments, launching experiments from the GUI, and generating graphical reports,
- iCanCloud provides a POSIX-based API and an adapted MPI library for modeling and simulating applications. Also, several methods for modeling applications can be used in iCanCloud: using traces of real applications; using a state graph; and programming new applications directly in the simulation platform,

iCanCloud is in charge of achieving four main tasks: (1) managing the VMs of the entire cloud system, (2) managing the list of jobs submitted by users, (3) scheduling those jobs to be executed in the corresponding VM instances, and (4) defining cost policies for each VM instance type.

Data centers represent a set of Virtual Machines, each one configured with pre-defined features such as CPU, storage, memory, and network. Finally, users are represented as entities that submit a set of jobs to be executed on specific VM instances.

iCanCloud is an extensible simulator. New components can be added to the repository of iCanCloud to increase the functionality of the simulation platform. For instance, in [5] the authors specified that one of the main directions would be to incorporate a power consumption estimation module, enabling the research on Green IT techniques involving cloud computing.

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 2) environments.

3. CloudSim

As described previously, CloudSim was developed in the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the Computer Science and Software Engineering Department of the University of Melbourne. It supports seamless modeling, simulation and experimentation
on designing cloud infrastructures. It is used to model data centers, service brokers, scheduling and allocation policies of large-scaled cloud computing platforms. CloudSim uses Java language and is built on top of SimGrid framework [1]. It provides features for modeling and creation of virtual machines or engines in a data center.

The main components of CloudSim are: Datacenters, Hosts, Virtual Machines (VM) and Cloudlets. The data center entity manages a number of host entities. The hosts are assigned to one or more VMs based on a VM allocation policy that should be defined by the Cloud service provider. Here, the VM policy stands for the operations control policies related to VM life cycle such as: provisioning of a host to a VM, VM creation, VM destruction, and VM migration. Similarly, one or more application services can be provisioned within a single VM instance, referred to as application provisioning in the context of Cloud computing. In the context of CloudSim, an entity is an instance of a component. A CloudSim component can be a class (abstract or complete) or set of classes that represent one CloudSim model (data center, host).

A data center can manage several hosts that in turn manage VMs during their life cycles. Host is a CloudSim component that represents a physical computing server in a Cloud: it is assigned a pre-configured processing capability (expressed in millions of instructions per second—MIPS), memory, storage, and a provisioning policy for allocating processing cores to VMs. The Host component implements interfaces that support modeling and simulation of both single-core and multi-core nodes.

VM allocation (provisioning) is the process of creating VM instances on hosts that match the critical characteristics (storage, memory), configurations (software environment), and requirements (availability zone) of the SaaS provider. CloudSim supports the development of custom application service models that can be deployed within a VM instance and its users are required to extend the core Cloudlet object for implementing their application services. Furthermore, CloudSim does not enforce any limitation on the service models or provisioning techniques that developers want to implement and perform tests with. Once an application service is defined and modeled, it is assigned to one or more pre-instantiated VMs through a service-specific allocation policy.

For each Host component, the allocation of processing cores to VMs is done based on a host allocation policy. This policy takes into account several hardware characteristics, such as number of CPU cores, CPU share, and amount of memory (physical and secondary), that are allocated to a given VM instance. Hence, CloudSim supports simulation scenarios that assign specific CPU cores to specific VMs (a space-shared policy), dynamically distribute the capacity of a core among VMs (time-shared policy), or assign cores to VMs on demand. Each host component also instantiates a VM scheduler component, which can either implement the space-shared or the time-shared policy for allocating cores to VMs. Cloud system/application developers and researchers, can further extend the VM scheduler component for experimenting with custom allocation policies.

Figure 3 shows the multi-layered design of the CloudSim software framework and its architectural components. The CloudSim simulation layer provides support for modeling and simulation of virtualized Cloud-based data center environments including dedicated management interfaces for VMs, memory, storage, and bandwidth. The fundamental issues, such as
provisioning of hosts to VMs, managing application execution, and monitoring dynamic system state, are handled by this layer.

A Cloud provider, who wants to study the efficiency of different policies in allocating its hosts to VMs (VM provisioning), would need to implement his strategies at this layer. Such implementation can be done by programmatically extending the core VM provisioning functionality. There is a clear distinction at this layer related to provisioning of hosts to VMs. A Cloud host can be concurrently allocated to a set of VMs that execute applications based on SaaS provider’s defined QoS levels. This layer also exposes the functionalities that a Cloud application developer can extend to perform complex workload profiling and application performance study. The top-most layer in the CloudSim stack is the User Code that exposes basic entities for hosts (number of machines, their specification, and so on), applications (number of tasks and their requirements), VMs, number of users and their application types, and broker scheduling policies. By extending the basic entities given at this layer, a Cloud application developer can perform the following activities: (i) generate a mix of workload request distributions, application configurations; (ii) model Cloud availability scenarios and perform robust tests based on the custom configurations; and (iii) implement custom application provisioning techniques for clouds and their federation.

The main features of CloudSim are:

- backing for modeling and simulation of large scale Cloud computing data centers, virtualized server hosts, energy-aware computational assets, data center network topologies and message-passing applications, federated clouds,
• backing for dynamic insertion of simulation components, end and begin of simulation,
• backing for user-defined policies for allocation of hosts and resources to virtual machines,
• supplies modeling and simulation of large scale Cloud Computing surroundings, including Data centers on a single physical computing node,
• an autonomous platform for modeling Clouds, Service Brokers, provisioning, and allotment approaches,
• supplies simulation of network connections among the simulated arrangement components,
• facility for simulation of league cloud surroundings that inter-networks assets from both private and public domains, a feature critical for research studies related to Cloud-Bursts and automatic application scaling.

CloudSim is extensible and many features can be easier added and enabling the modeling of new types of applications, not supported by CloudSim. Currently the following features are included:

• Web session modeling;
• Utilities for generating CSV files for statistical analysis;
• Modeling of disk operations;
• Utilities for running multiple experiments in parallel;
• Utilities for modeling network latencies;
• Map Reduce simulation.

4. Synthesis and objectives

Table 1 presents a comparison of the different simulators that we introduced previously on the supported resource services. As described, all simulators lack on providing simulation and modeling of cloud platform and application. Furthermore, there is no approach using standardization for modeling different cloud resources.

<table>
<thead>
<tr>
<th></th>
<th>SimGrid</th>
<th>GreenSim</th>
<th>iCanCloud</th>
<th>CloudSim</th>
</tr>
</thead>
<tbody>
<tr>
<td>IaaS</td>
<td>Some resources</td>
<td>Some resources</td>
<td>Some resources</td>
<td>Yes</td>
</tr>
<tr>
<td>PaaS</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SaaS</td>
<td>No</td>
<td>No</td>
<td>Only (HPC)</td>
<td>No</td>
</tr>
<tr>
<td>Standard</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 1. Simulator synthesis.
The goal of our study is to extend the OCCIware framework [8] for supporting a simulation environment. Until now, OCCIware framework does not integrate a simulator on its model. To achieve this, we need in the first time to generate a representation of the workload, i.e. the set of inputs, to be applied to the studied system. This representation may be a trace (or scenario) depicting a real workload or a synthetic workload generated by heuristics or stochastic functions. The second step is related to the simulation. It is performed by applying a workload on a model of the system and producing results. Finally, the third step consists in analyzing simulation results in order to acquire a better understanding of the considered system.

Within the OCCIware project, we mainly use OCCI to model the first step. We provide an interpreter for OCCI designed to bridge the graphical editor to the simulation for implementing the second step. Finally, a declarative language for describing rules can be plugged on demand during the simulation to ensure that no constraints are violated.

However, to realize our simulation, we need to choose the most appropriate simulator for our framework among the previous described simulators. None of the current distributed system simulators offer the environment that can be directly used for modeling cloud environments. Nevertheless, CloudSim, which is generalized, and extensible simulation framework, allows seamless modeling, simulation, and experimentation of emerging cloud computing infrastructures and application services. By using CloudSim, researchers and developers can test the performance of a newly developed application service in a controlled and easy to set-up environment. The vast features of CloudSim would speed up the development of new application provisioning algorithms for Cloud computing.

We believe that CloudSim is the most appropriate simulator to achieve our goal. It allows us to model and simulate all the cloud infrastructure resources, it requires very less effort and time to implement Cloud based application provisioning test environment and developers can model and test the performance of their application services in heterogeneous Cloud environments (Amazon EC2, Microsoft Azure) with little programming and deployment effort.

5. Conclusion
In this deliverable, we have discussed the various simulators that have taken place in cloud computing. First we presented an overview of various cloud computing simulator tools including their advantages and limits. Then in the second part of the deliverable, we presented in a more detailed manner CloudSim, since we believe that it is the most suitable simulator for our goal. Finally, we gave a synthesis of our study following by our objectives and future work.
Bibliography


