Computer Networks 108 (2016) 78–96

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Review article

# Building a cloud on earth: A study of cloud computing data center simulators☆

Mohamed Abu Sharkh[a,*], Ali Kanso[b], Abdallah Shami[a], Peter Öhlén[b]

[a] Department of Electrical and Computer Engineering, Western University, London, Canada
[b] Ericsson Research, Montreal, Canada

ARTICLE INFO

*Article history:*
Received 4 August 2015
Revised 30 May 2016
Accepted 27 June 2016
Available online 22 July 2016

*Keywords:*
Cloud computing
Cloud simulators
Scalability
Data centers
Virtualization
Network and systems monitoring and measurements

ABSTRACT

As cloud computing technologies finalize their transformation into the standard technologies for businesses of all sizes, they face more scrutiny than ever. Clients are expecting the benefits of turning infrastructure, platform and network into services payable per use without tolerating any service hiccups caused by performance bottlenecks or overprovision. This puts cloud providers under pressure to deliver data center management solutions and deployment plans in minimal time and with failure allowance close to none. Any comprehensive solution evaluation could gain much from the use of cloud simulators. Cloud simulators have the advantage of practicality over both mathematical proofs and real testbeds. They support any amount of heterogeneous use cases demanded by the cloud provider. Despite being a relatively new concept, multiple cloud simulators were developed. However, they are still in the phase of adapting to the scenarios, objectives and characteristics of the cloud. This paper examines a selected set of the current cloud simulators in terms of vision, features, and architecture. Strong points and limitations are discussed. Moreover, this paper presents a framework for cloud simulator design that can serve as an elaborate design checklist. A discussion of the open research challenges concludes the paper.



## 1. Introduction

As the cloud computing client base grows, the cloud service providers face the challenge of adapting to the needs of this growth in both the technical and business dimensions. Cloud providers should maintain this gradual enhancement without losing focus on delivering the level of service their clients demand. The dynamic and unpredictable nature of cloud computing adds extra layers of complexity to the providers' tasks. This challenge is magnified by the rise of Big Data concepts that are pushing a new wave of solutions and use case scenarios. Newly developed cloud solutions should be able to handle the scale client data have reached. It is expected of the available infrastructure and software stack to serve the 2.5 Quintillion Bytes (2.3 trillion Gigabytes) that are created every day [1]. IBM estimates that there will be 18.9 billion network connections by 2016. This covers all types of connections especially the ones originated at smartphones carried by any one of the 6 billion expected carriers all over the world.

In the article on Big data facts, myths and possibilities [2], O'Neal presents an example of this change. "With machine to machine (M2M), though, this paradigm changes. Systems generate vast amounts of data independent of human business processes – collecting, analyzing and then deploying this information for use represents a net-new activity for most organizations, in both IT and business operations."

When you add to this mix the number of technical/performance related parameters involved in forming a cloud solution, the task looks increasingly challenging. Providers need efficient tools to support solution design decisions related to deployment models, resource allocation, scheduling, and performance adjustments. Evaluating solutions directly and from the beginning of the solution development process using the real infrastructure is not always practical due to cost factors. The idea of benchmarking using a subset of the infrastructure (like a set of servers with the same configuration and the same topology of the data center for example) would not guarantee a wholesome vision of scalability issues. Scalability is a key player in this scenario and it (along with the cost) constitutes a challenge when using real testbeds. Solution evaluation using analytical methods is rendered infeasible due to the increasing complexity as the scale of the problem grows. Simulation arises here as a tool that – while is not enough alone to handle the whole cloud solution evaluation process – can rather

☆ This work is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC-STPGP 447230) and Ericsson Research.

* Corresponding author.
*E-mail addresses:* mabusha@uwo.ca (M. Abu Sharkh), ali.kanso@ericsson.com (A. Kanso), ashami2@uwo.ca (A. Shami), peter.ohlen@ericsson.com (P. Öhlén).

http://dx.doi.org/10.1016/j.comnet.2016.06.037
1389-1286/

play major roles before a solution is deployed on real hardware. These roles can be considerably less expensive and with less risk. Cloud simulation involves modeling a real or a proposed cloud system using computer software. It is notably useful when changes to the actual system are either difficult to implement, involve high costs, or are impractical.

Several attempts have been made to develop a competitive cloud simulator. Each cloud simulator differs in vision, the focal points and the resulting features. In this work, we examine the major cloud simulators available to researchers and industry engineers and compare them in terms of the main simulated components, application model, network model, and architecture. Previous attempts to survey cloud simulators can be seen in the literature [3–5] and [86]. In this work, we strive to offer an updated and more comprehensive view of this topic. We present the limitations found in each simulator using an approach that depicts what it is and what it is not. We also aim at illustrating the ground on which the current simulators stand. This helps us to construct a framework for the cloud simulator design process which would ideally cover the industry and research community needs.

Moreover, the paper offers a deep analysis of the open research challenges related to this topic. Challenges covered include realistic user application patterns, cloud deployment and pricing, reliability and high availability, challenges originated outside the data center and Big data considerations.

The coming sections are divided as follows:

Section 2 details the specific roles expected of a cloud simulator. Section 3 is a discussion of the design decisions included in the process of developing a cloud simulator in terms of visions for hardware, applications and network sides. Next, Section 4 traverses a chosen set of common cloud simulators, giving significant attention to a few simulators that contain interesting implementation concepts. This simulator comparison is then consolidated in Section 5. We then proceed to discuss pros and cons of building a new simulator as opposed to extending a current one. Finally, an illustration of some of the research challenges and future work is presented and the paper is concluded.

## 2. The activities of a cloud simulator

To better understand expectations of the varying simulator users, a clear perspective of the cloud simulator role needs to be defined. This can be done by depicting the potential cloud service planning activities a simulator can serve in a cloud environment. Each activity can be matched with a use case where using a cloud simulator can achieve the required impact. These activities can be summarized in the following:

1. **Define**: Develop greater understanding of process details. Service deployment options, green data center policies and high availability policies are examples of aspects that are affected by several conditions that work simultaneously. Understanding their interaction on the lowest level is a must if efficient resource management is to be achieved.
2. **Pinpoint**: Identify problem areas or bottlenecks in a process that affect execution speed or increase the solution cost.
3. **Maneuver**: Test different “What if?” scenarios to better predict how a real life problem evolves under specific conditions.
4. **Analyze**: Evaluate the effect of system or process changes such as demand, supply, resources specifications, and constraints
5. **Decide**: Compare the impact of alternative policies to determine their points of strength. A simulator is an accurate way to quantify the advantages and disadvantages of newly developed policies.
6. **Scale**: Run real life scenarios on different scales as needed and repeat them as many times as the verification and validation process requires.

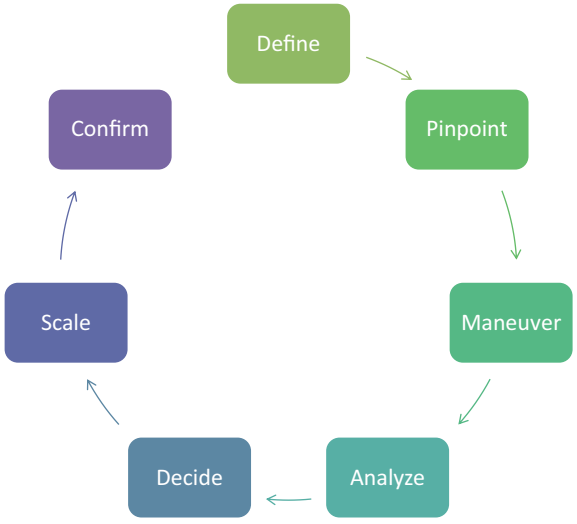



**Fig. 1.** The cloud service planning activities where a cloud simulator has a role to play.

7. **Confirm**: Use it as a last step to confirm the behavior of proposed solutions when a failed solution has high risk or high cost associations. This includes tuning performance bottlenecks before deployment. The simulator roles are illustrated in Fig. 1.

An example of a use case in which the simulator would play these roles is a scenario where the architects are developing a multi layered resource allocation policy for a cloud data center. This use case constitutes developing a solution that performs the resource allocation for the cloud in order to minimize user request latency to comply with service level agreements (SLAs). The cloud simulator first would help us define the problem by answering questions like: Which elements are involved in this experiment (environment)? Which resources are affected? When do we apply the resource allocation algorithm? Then, it would help us pinpoint potential bottlenecks that would cause this solution to underperform: Is it the VM placement? Is it the resource allocation for the VMs? Is it the chosen topology? Is it the network resources? Then, it would help us maneuver by testing multiple “What if?” scenarios in terms of testing different data sets or use cases or edge cases. Now, we analyze the effect of every factor on the resource allocation process. For example, we might notice that the algorithm performs with sparse heavy requests better than numerous light requests or that it performs better with high-CPU VMs. Based on that, we can gauge and decide which factors have more importance and give them more weight in our scheduling policy. Next step is to scale by testing our scheduling algorithm for a very large data center, large internal network, heavy communication and a large number of requests and see where/when it breaks. Finally, it can be used for final tuning before the algorithm is tested on a real test bed to schedule real requests.

The potential users taking advantage of these simulator roles include:

**1. Cloud providers and solution architects:** Naturally, cloud providers represent the main user as they would use this to develop, evaluate and improve their solutions.

**2. Cloud clients:** Typical cloud client list include large companies that have the capabilities to run their own clouds. A cloud simulator would be beneficial to compare different providers, evaluate currently deployed solutions or for studies of the client workload and to support decisions regarding private vs. public clouds.

**3. The research community:** simulators are a critical preliminary step for researchers who are developing new cloud technology before testing on a real setup.

**Table 1**
Simulator basic elements as an initial set of design decisions.

| Design element | Examples |
|---|---|
| Entities/components | Servers, VMs, racks, data centers, clouds, switches (access, aggregation, etc), links. users (clients) |
| Entity attributes | Capacity, power consumption |
| Simulation scheme | Stochastic, deterministic |
| Events | Client arrival, new task, new application, task completed |
| Event frequency and duration | Covering traffic models and client request generation process (Exponentially distributed, normally distributed, uniformly random, deterministic frequency) |
| Activities | Schedule a VM, schedule a task, migrate a VM |
| Process sequence | Relations defined between events, activities and outcomes |

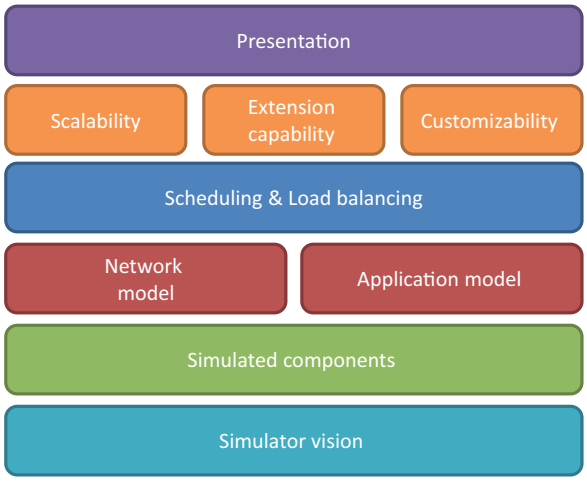



**Fig. 2.** Cloud simulator design framework.

**4. Other external players:** Simulators can be a useful supportive for any parties concerned of evaluating cloud solutions. This could involve auditing and consulting teams or government teams investigating the energy efficiency or carbon print of a cloud for example.

## 3. Comparisons methodology

### 3.1. Cloud simulator design decisions

The process of designing a simulator includes informed decisions related to: components being produced, served, or acted upon by the simulation process. Typical components in the cloud include servers, racks, switches, links, applications, and users. Furthermore, the layout should cover the simulation process flow, its associated resources and events or process steps. A special attention should be given to event frequency and duration. The choice of which probability distributions better characterize execution uncertainties and process variations is crucial. Table 1 summarizes the list of elements to be considered when designing the ideal cloud simulator. The first prototype of the simulator can be generated based on these elements.

### 3.2. Cloud simulator ingredients

A major objective of this work is to examine each cloud simulator and discuss the fundamental aspects presented by each one of them. We start by introducing a framework of design components for simulators. This cloud simulator design framework is illustrated in Fig. 2.

#### 3.2.1. Simulator motivation and vision

The motivating purpose of the developing team when constructing the simulator drives the construction process and built-in features. For example, when the team aims to test cloud computing deployment solutions, a special consideration is given to user request patterns, geographical distributions, types of resources requested and pricing packages. On the other hand, if the prime purpose is performance enhancement, the focus is shifted to request scheduling algorithms, data center network topology options, and general virtualization efficiency. Furthermore, a cloud simulator created to test power consumption methodologies, will contain extensive policies for power consumption recording, variety of power saving techniques revolving around server consolidation or hardware solutions.

#### 3.2.2. Simulated physical components and architecture

As discussed previously, the choice of simulated components included affects the simulator architecture and general organization. First, a cloud simulator by definition will contain components representing servers, server resources, and at least symbolic network representation. This is especially important as most of the simulators focus on the Infrastructure as a Service (IaaS) aspects of the cloud offerings not Platform as a Service (PaaS) or Software as a Service (SaaS). The choice of server resources to include differs from a simulator to another. Memory, storage capacity, processing units are almost always there. Some configurations add more resources like chips with a certain purpose, for example. The number and capacity of data centers come into play here. Moreover, the vision of distributing clients in terms of different locations and varying request probability distributions is a point to discuss. Virtual machine (VM) offerings and resource allocation are other critical factors. Some simulators will go with a predefined set of VMs with fixed resource amounts allocated to these VMs. This gives the user a choice of VM types or models to choose from [15]. Another alternative is assigning the VM a chunk of resources and then scaling this amount up or down based on the VM real time usage. This will require a more elaborate adaptive resource scheduling and allocation despite it seeming like a more efficient method.

#### 3.2.3. Application model

Cloud simulators vary in how they represent user requests and execution components. A subset of the simulators go for the direct method of representing user requests as a list of resource specifications (required processing time or Million Instructions per Seconds (MIPs), required memory, required storage, preferred start time, duration, and/or a deadline). An enhancement of this method can been seen when requests are abstracted into applications or application components. This scenario, to represent reality more accurately, would have to include request inter-dependability and input/output control. In addition, effects on request scheduling will have to be considered. Moreover, dynamic scalability is one of the defining attributes of the cloud. the ability to scale in and out should be accounted for in the application model. This includes increasing the number of components serving a specific application and the increasing the capabilities. Sometimes, these two options are differentiated by calling the former scale out and the latter scale up.

*3.2.4. Network model*

(a) Topology representation:

Laying out the data center network includes multiple aspects. The topology used arises as a critical issue. When a simulator supports multiple topologies and becomes open for adding new ones, it inherently supports a much larger set of experiments. Tree based topologies are common in cloud data center designs. Server centric topologies like DCell [16] and BCube [17] and switch centric topologies like jellyfish [18] and fat-tree [19,20], are all possible solutions.

(b) Network request representation:

Another issue is choosing how to represent data communication (network) requests. The alternatives here are the packet model or the flow model [13]. In the flow model, network requests are dealt with as flows from point A to point B and the aim is to find links with available bandwidth capacity. Some simulators go with an even simpler model that considers bandwidth a commodified resource in a similar manner to memory or storage. In this method, the source and destination of a network request are not looked at. Each host is assigned a bandwidth capacity and requests demand is deducted from this capacity during the request lifetime.

(c) Other network model concerns:

Many challenges that are faced when designing a cloud computing system still face software architects when they try to produce a cloud simulator. Deciding which techniques to be used for routing of a network request (both initially and rerouting) is one of them. Reducing tardiness here is the main objective. More elaborate simulators could even introduce alternatives for traffic decision similar to what is available in cloud data centers. These alternatives include: implementing traffic decision by the relay switch, implementing traffic by a central controller (similar to techniques used in Software Defined Networking(SDN) controllers) [87,88] or even letting the user make the decision. The last alternative is being promoted now using the term Routing as a Service [21].

The choice between fixed and flexible bandwidth allocation is another decision. However, this issue can be mitigated much easier in a software environment as the switch between these two methods does not prove costly or complicated.

*3.2.5. Resource scheduling, allocation and load balancing*

When faced by the task of designing a resource allocation methodology, many external and internal challenges should be considered. An attempt to summarize these challenges can be found in the article by Abu Sharkh et al. [6]. External challenges discussed include regulative and geographical challenges as well as client demands. This results in constraints on the location of the reserved VMs and restrictions impacting the data location and movements. External challenges also include optimizing the charging model in such a way that generates maximum revenue. Internal challenges also include data locality issues. The nature of the application in terms of being data intensive should be considered while placing the VMs and scheduling connections related to this application. All these factors should be put into consideration when choosing the resource scheduling policies. Supporting multiple preconfigured policies is a strong point for any simulator [89]. Moreover, allowing the user to plug in their own policies is another plus that enables users to evaluate their policies precisely. Resource scheduling has a critical role affecting power efficiency, availability, and general data center performance. Fig. 3 encompasses the elements in play in the resource allocation process within a cloud environment. The client sends requests to reserve VMs or to execute a task on a VM where this task can be a communication or computational task. The Cloud management system receives the requests and distributes them to the corresponding VM for execution. This is all done while maintaining requirements related to load balancing, high availability, energy efficiency, revenue generation and performance. It is expected that a cloud simulator is able to represent and simulate this process successfully.

*3.2.6. Extension capability and customizability*

A successful cloud simulator, like any software product, gains more client market penetration when the cost to customize is less. Changing the software to meet clients' function automation can take one of two forms:

(a) Configuration: where the cloud simulator has the required capabilities and what remains is selecting the correct setup options (configuration) or adding minor GUI components.

(b) Customization: where additional functionality or features that did not exist before are to be added.

The chief concerns regarding customizability that arise for clients requiring cloud simulators are:

(a) Lack of wide vision:

It is found that many simulators are built to serve a specific research purpose or model. These simulators are then extended/presented as a general cloud simulator when they achieve success. This leads to this simulator being beneficial to researchers/industry parties working in a similar topic. This topic could be power consumption, scheduling efficiency or cloud pricing options. However, for parties working on a different topic, the customization process proves non trivial. This could lead to increasing the customization cost to exceed the value of building new software components.

(b) Availability issues and other typical open source software challenges:

For open source cloud simulators, lack of documentation and support is a consistent issue. Besides, simulator reliability is something to consider.

*3.2.7. Scalability*

The ability of the simulator to scale up to realistic use cases is an important factor when considering whether to build a new simulator from scratch or extend an already existing one. A simulator should have the ability to handle complex topologies and a number of requests large enough to represent real cloud environments. The used algorithms' space and time complexity is a prime indicator here. The type of environments/programming languages a simulator is based on may add scalability challenges as well. Some cloud simulators are based on simulation engines that govern event creation/succession and status updates. This base engine might impose limitations on the size of the solved problem, the number of nodes or the request arrival rates. A simulator that might solve a problem in an acceptable amount of time might not be able to do the same when faced with a problem of larger scale. Determining the acceptable period of time to get a solution for an instance of the problem depends mainly on the client tolerance and the problem urgency.

*3.2.8. Presentation issues*

Well designed appearance and detailed GUI are desirable features in a simulator. Although a new GUI can be built for basically any simulator if it is deemed to have the required core capabilities, having that in place saves time and effort. An issue cloud simulator architects should consider here is including all the simulator critical input values in the input forms/screens. This would save effort for any teams working to extend the simulator in the future.

## 4. Cloud simulator detailed review

In this section, a select set of simulators are put under a magnifying glass. The objective is to take a closer look at the Why, What, and How of their implementation. More specifically, a focus on the motivation, distinct features not available elsewhere,

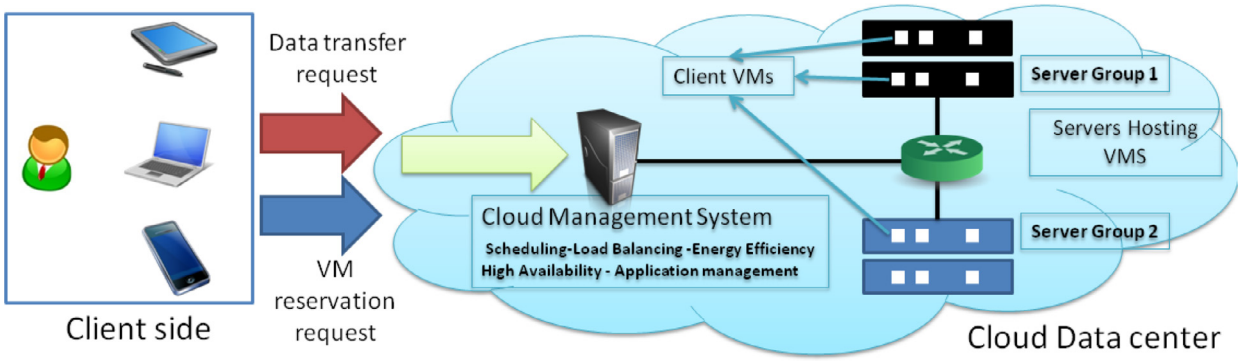



**Fig. 3.** Cloud simulated environment and components.

strong points and limitations. This survey does not, by any means, target exhausting every single cloud simulator. The aim, instead, is to focus on simulators with high impact and interesting design concepts in order to cover a high percentage of the variations of available simulators. The covered simulators include, CloudSim, NetworkCloudSim, GreenCloud, iCanCloud, TeachCloud, GroudSim, CloudAnalyst, CDOSim, MDCSim, GDCSim, SPECI, and BigHouse. In the interest of saving space and not repeating the content in each simulator, we have explored the ingredients of the first three simulators in more detail. Then, for the remaining simulators, we included the first ingredient (motivation for building the simulator). Moreover, in the second subsection (main features), we focused on the unique features and major additions appearing in each simulator and not discussed before in the previous simulators. This includes major added features related to the simulated components, network model, application model, power efficiency or other aspects of cloud environments.

### 4.1. CloudSim

#### 4.1.1. Description and motivation

CloudSim is one of the most commonly used cloud simulators. CloudSim is a simulation toolkit and an application that enables modeling of single clouds or cloud networks. The designers of the system cited the existing distributed system simulators inapplicability to evaluate: “the performance of cloud provisioning policies, application workload models, and resources performance models in a repeatable manner under varying system and user configurations and requirements” [7], as their leading motive to develop CloudSim. CloudSim is not a standalone fixed-scenario simulator. CloudSim users have the ability to develop the cloud scenario that most fits their needs, design the input parameters and evaluate the output patterns. CloudSim uses robotics simulator Gazebo and is based on an underlying toolkit called SimJava [22]. SimJava uses a discrete event simulator. It includes facilities for representing simulation objects as animated icons on screen.

#### 4.1.2. Simulated physical components and architecture

Through CloudSim, large cloud data centers can be simulated. This includes hosts, virtual components, federated cloud scenarios. Moreover, CloudSim enables users to define and control resource allocation and provisioning policies, virtualization techniques, and energy consumption management techniques. On the operational side, CloudSim supports adding elements dynamically and pausing/resuming the simulation.

(a) CloudSim architecture:

Studying CloudSim design would give potential architects the insight they need in terms of the required components and layers they will need to build. In their CloudSim introduction article [7] the authors explain the layered organization of CloudSim in detail [7]. SimJava is the discrete event simulation engine that administers tasks like queuing and event processing, creates the system components, and manages the clock. GridSim toolkit is above SimJava. It performs two roles: (i) implementing infrastructure components similar to the ones used in grid applications like networks and traffic distributions; (ii) critical operational components like resource types, data sets, and workload traces. CloudSim is implemented at the highest level containing core cloud functionalities related to data center design and virtualized cloud resources. Finally, on top of the simulation stack comes the user code. The user defines parameters related to resource configuration and the number of servers, number of users, and cloud deployment in terms of brokering options.

(b) Simulated physical components:

The model used by CloudSim employs a set of classes to represent basic cloud functionality. We will consider two of the critical classes mentioned in CloudSim introduction article [7] as an example. First, the DataCenter class is a core class to the simulator functionality. A similar class is expected to be seen in every major cloud simulator. Server sets can be homogeneous or heterogeneous in terms of resource types and available allocations. Moreover, a DataCenter object instantiates a generalized resource provisioning component that implements a set of resource allocation policies. Attributes of a data center include: Architecture, Operating system, List of machines, Allocation policy, Time or space-shared, and Resource price per time unit(which represents cost of memory, storage and bandwidth(BW).

Second, is the DatacenterBroker class. The broker’s responsibility is standing between service providers and cloud clients. The purpose is to find the most suitable solution package to the client’s required resources and Quality of Service (QoS) conditions.

#### 4.1.3. Application model

User application in CloudSim is represented through Cloudlet class. Application size (complexity) is represented based on computational demands (instruction length). This is translated into two numbers: instruction length and amount of data transfer (both pre and post fetches totaled). CloudSim does not specify any VM dependencies or data exchange requirements apart from that. Properties of a Cloudlet include: *(i)* length (in Million instructions - MI); *(ii)* file size; *(iii)* output size; *(iv)* number of CPUs.

#### 4.1.4. Network model

CloudSim reads the data center topology using an input file in the BRITE format. For every node, the file contains attributes specifying x and y-axis coordinates, in-degree and out-degree of the node. For each edge, it includes specifications of the source, destination, Euclidean length, propagation delay, bandwidth and type.

**Table 2**
A summary of CLoudSim features.

| Feature | Available? | Details |
|---|---|---|
| Ability to model user requests | Partially | Applications represented by a workload object that contains user workloads (by MIPS) |
| Ability to model inter-VM dependency | No | – |
| Ability to model multiple DCs | Yes | – |
| Ability to model servers | Yes | With fixed set of attributes |
| Ability to model network elements | Limited | Data center network is modeled using BRITE format but not used. Internal network not represented. |
| Ability to model VMs | Yes | Resource configuration and placement |
| Ability to model inter-VM connectivity | No | BW required by a VM is treated as a fixed commodity |
| Ability to model failures/recoveries | No | – |
| Ability to model energy power sources mix (24 h source types | No | – |
| Ability to model power usage per VM, server, facility | Yes | Multiple power management methods are implemented Basic power usage statistics are available |
| Ability to model security measures (attacks, firewalls) | No | – |
| Ability to model network flows | No | – |

**Table 3**
CloudSim test parameters.

| Data Center Parameter values | VM parameters |
|---|---|
| a-system architecture = "x86" | a-image size = 10000 MB |
| b-operating system = "Linux" | b-VM memory = 512 MB |
| c- processing resource cost= 3.0 | c- MIPS = 1000 |
| d- memory resource cost = 0.05 | d-BW = 1000 Mbps |
| e- storage resource cost = 0.1 | e-number of CPUs = 1 |
| f- BW resource cost = 0.1 | f-VMM name = "Xen" |
| Host parameters | Cloudlet(application) parameters |
| a-host memory (2048 MB) | a-length = 1000 MI |
| b-host storage = 1,000,000 | b-file size = 300 |
| c-BW= 10000 Mbps | c-output size = 300 |
| | d-required CPUs = 1 |

Despite reading the topology details, this information was not mainly used in the CloudSim until NetworkCloudSim was added. The network inside the data center is not explored in detail either (Table 2).

#### 4.1.5. Power consumption features

CloudSim contains basic energy consumption recording statistics (energy consumed, CPU utilization, etc.). The CloudSim team members have performed a detailed implementation of multiple energy conservation policies. These policies are mostly based on two energy conservation concepts:

*(i)* DVFS: A simulation of a heterogeneous power-aware data center that only applied Dynamic voltage and frequency scaling (DVFS), but no dynamic optimization of the VM allocation; and

*(ii)* Server consolidation/VM migration: Simulations of heterogeneous power-aware data centers testing multiple VM allocation policies and VM selection policies (like migrated VM selection method). The goal is to arrive at the combination that performs best in terms of energy consumption.

These used policies include: *(i)* VM allocation techniques (like Inter Quartile Range (IQR), Local Regression (LR), Local Regression Robust (LRR), Median Absolute Deviation (MAD) and Static Threshold (THR)); *(ii)* VM selection policies like (Maximum Correlation (MC), Minimum Migration Time (MMT), Minimum Utilization (MU) and Random Selection (RS)). There was no consideration for different energy power sources in CloudSim.

#### 4.1.6. Scalability testing results

We tested CloudSim on our machine that has 8 cores and 64 GB of memory. Parameters are included in Table 3.

(a) Execution times:

We scaled the problem up by changing the number of VMs, Cloudlets and hosts. Apart from the problem size, many factors may affect the speed of the execution. For example, a considerable increase in the time delay is noticed when the VM/application load is more than the hosts could handle. This might be due to limitations in the VM placement algorithms. It is noticed that if the simulator cannot allocate the VM, it will go through every single host and report that they do not have enough space to allocate it.

(b) Testing different types of loads/applications:

To further specify the factors that affect CloudSim runtime, we performed additional tests on a load of 20K VMs, 40K Cloudlets and 10K hosts with four processors per host. This time, the controlled parameters included: *(i)* simulated data center host processing capacity; *(ii)* simulated VM assigned processing speed (in Million instructions Per Second (MIPS)); *(iii)* simulated requested CLoudlet size (in MI). The results in Table 5 show that the load type does not have a significant effect on CloudSim runtime if the load amount does not change. If we look at one of the major cloud providers, we can find that "Amazon data centers house between 50,000 and 80,000 servers, with a power capacity of between 25 and 30 megawatts." [8] In comparison, Google's major data centers are supported by at least 50 megawatts of electric power, with some estimates ranging as high as 103 megawatts [9] which supports the 2013 estimates in Miller's report [10] which put their server count at 900,000 in 13 data centers at the time. (Averaging around 70,000 servers per data center). When Rackspace reached 70,000 servers, it was only the sixth company to reach this number (in total servers operated) [11]. This number is in all of its data centers in six cities around the world [12]. Therefore, the largest experiment we have conducted on CloudSim (30,000 servers) is fairly close to the biggest data centers on the planet and would cover most of the other data centers.

### 4.2. NetworkCloudSim

#### 4.2.1. Description and motivation

NetworkCloudSim is an extension of CloudSim that focuses on network capabilities. The major motive is enhancing CloudSim with complex application models such as message passing applications and workflows in addition to supporting a scalable network model for cloud data centers. As for the first point, other simulators like CloudSim and MDCsim model application requirements in the form of application size in terms of the number of instructions or other variations of computational resource requirements. A more detailed definition of the client applications is required to cover scenarios like parallel applications and workflows.

As an extension to CloudSim, it is similar in operation to it. NetworkCloudSim is capable of simulating Data center networks

**Table 4**
CloudSim Execution times for different load amounts.

| Number of VMs | Number of apps (cloud lets) | Number of hosts | Execution time |
|---|---|---|---|
| 20 | 40 | 10 | 3 s |
| 2k | 4k | 1k | 4 |
| 10k | 20k | 5k | 1 min 44 s |
| 20k | 40k | 10k | 8 min 43 s |
| 100k | 200k | 30k | 267 min |

**Table 5**
CloudSim execution times for different load types when for 10 K hosts, 20 K VMs,40 K cloudlets with four processors per hosts.(the changed parameter in each line is put in bold).

| VM capacity (in MIPS) | Cloudlet size (in mi) | Host processing capacity (in MIPS) | Average execution time |
|---|---|---|---|
| 1000 | 1000 | 2000 | 8.33 s |
| **2000** | 1000 | 2000 | 8.23 s |
| **100** | 1000 | 2000 | 8.30 Ss |
| 1000 | **10000** | 2000 | 8.29 s |
| 1000 | **100** | 2000 | 8.21 s |
| 1000 | 1000 | **5000** | 8.28 s |

(DCNs) an applications of communicating tasks such as a message passing interface (MPI). The authors have designed a network flow model for cloud data centers utilizing bandwidth sharing and latencies to enable scalable and fast simulations [13].

*4.2.2. Simulated physical components and architecture*

Being an extension of CloudSim, many aspects of NetworkCloudSim are largely similar to the original simulator specially in the simulated physical components and architecture. We will not traverse these similar aspects in the interest of saving space. However, significant contributions are seen in the application model and network model. we will focus on those instead.

*4.2.3. Application model*

In typical simulators, you can send a request to define a task on more than one processor. What really happens is that they are scaled to the computational time of one processor which is not the real scenario. Tasks in a real cloud computing scenario have to communicate. NetworkCloudSim introduced the NetworkCloudlet class to represent a task executing in several phases/stages of communication and computation.

Fig. 4 shows how the areas/classes in CloudSim architecture are affected by the changes/extensions of NetworkCloudSim. To model the application precisely, a class called AppCloudlet is introduced. Each application contains several communicating elements (NetworkCloudlets). Each element runs in a VM and consists of stages where it either performs data exchange tasks (communicating) or computing tasks. These stages are named: *execute, send data, receive data* or *finished*. Computing stages can be defined by their MIPS requirements while data transfer tasks are characterized by the amount of data. When in the *send* stage, the VM scheduler submits a packet to the send-packet-queue of the VM. After the execution stage of each NetworkCloudlet, VM scheduler forwards theses packets either to VMs on the same host or to switches. In NetworkCloudSim's implementation, no messages will be blocked even if the destination is not ready to receive the message. The receiver VM has the option either to process other tasks or to be blocked until the message arrives. The authors summarize: "This communication model allows the Simulation of the non-blocking message passing paradigm (such as MPI Isend() and MPI Irecv()), which is a common practice in parallel applications" [13].

*4.2.4. Network model*

Quality of Service conditions required by a cloud client often take a hit because of network request latency. NetworkCloudSim aims at modeling realistic network requests in terms of topology, request size, and hierarchy. CloudSim lacks the ability to facilitate intra–data center communication. Bandwidth sharing on network links is not modeled. This is an obstacle to modeling features like VM migration. There are two questions posed while extending network capabilities. These questions where addressed in detail in NetworkCloudSim introduction paper [13]. First, designers should choose if they want to go with a flow model or a packet model. A flow model has the advantage of lower computational overhead despite having less details. The second issue is VM interconnection topology. A fully connected model is not common in real data centers. NetworkCloudSim tackled this issue by adding root, aggregate and edge (access) level switches. This gives users the freedom of configuring switches and ports according to their use cases. A bandwidth allocation algorithm should be included here in the case that multiple simultaneous flows use the same link. The simulator adds present latency based on the link length as well.

To model a network within the data center, the Switch class has been introduced as a network entity (switch or a router) that can also model forwarding latency. Moreover, NetworkPacket and HostPacket classes represent data flow out of the VM. A HostPacket is transferred using the virtual network. A NetworkPacket goes from a server to another through the physical network (Table 6).

### 4.3. GreenCloud

*4.3.1. Description and motivation*

GreenCloud is a cloud simulator with a focus on energy efficiency and enhanced capabilities for network communications. The prime purpose cited for building GreenCloud is mitigating overprovision issues. Overprovision happens in a data center due to the changing loads on its computational and network resources. The average load can be as low as 30% of the data center server and network capacity [23]. This, in turn, causes the data center to systematically use more power than the optimal value.

GreenCloud is an open source extension of the network simulator NS-2 [85]. Data transfer processes are modeled on a packet level. Most of the code (80%) is in C++ and the rest is in Tool Command Language (TCL). The simulator records energy consumption for servers, switches, and links as well as having workload patterns. On the other hand, a challenge faced by GreenCloud is the limited scalability to small data centers due to very large simulation time and high memory requirements.

**Table 6**
A summary of NetworkCloudSim features.

| Feature | Available? | Details |
|---|---|---|
| Ability to model user requests | Yes | Can model communicating/dependent applications |
| Ability to model inter-VM dependency | Yes | – |
| Ability to model multiple DCs | Yes | – |
| Ability to model servers | Yes | With fixed set of attributes |
| Ability to model network elements | Yes | Internal DC network is modeled including switches |
| Ability to model VMs | Yes | Resource config and placement |
| Ability to model inter-VM connectivity | Yes | – |
| Ability to model failures/recoveries | No | – |
| Ability to model energy power sources mix (24 h source types | No | – |
| Ability to model power usage per VM, server, facility | Yes | Multiple power management methods are implemented, basic power usage statistics are available |
| Ability to model security measures (attacks, firewalls) | No | – |
| Ability to model network flows | Yes | The designers chose flow model over packet model to send/receive data |

*4.3.2. Simulated physical components and architecture*

The increasing scale to which data centers are growing (tens of thousands of hosts) and the increase in the percentage of internal communications (70% of all communications performed by data center components) [23] call for more attention to the data center architecture robustness and efficiency.

GreenCloud offers three options for the possible data center topology:

(a) Two-tier data center architecture:

Racks of servers form the tier-one of the network. Network Layer-3 (L3) switches facilitate full mesh connectivity using 10 GE links. Two-tier data centers in Greencloud may support up to 5500 nodes.

(b) Three-tier data center architectures:

Access, aggregation, and core layers are included in this architecture which increases supported number of servers to 10,000. This architecture supports an 8-way equal cost multi path routing (ECMP) with 10 GE Line Aggregation Groups (LAGs). This gives the client the ability to address several links with a single MAC address. However, LAGs are known to limit network flexibility and performance. As the authors put it: "LAGs make it difficult to plan the capacity for large flows and make it unpredictable in case of a link failure. In addition, several types of traffic patterns, such as ICMP and broadcast are usually routed through a single link only" [23].

(c) Three-tier high-speed data center architecture:

This architecture adds 100 GE links (IEEE 802.3ba) between aggregation and core switches. This leads to reduction in the core switches and avoiding the disadvantages of LAGs as well as cabling reduction and supporting data center scale ups.

*4.3.3. Application model*

Host objects represent servers containing typical data center resources. Scheduling techniques include round robin and energy aware scheduling. Tasks can either be scheduled on the hosts directly or on the VMs residing on the hosts.

(a) Task and user request specification:

A workload object consists of a computational part that is measured by MIPS and a network request part. GreenCloud enables choosing a distribution to configure the task arrival pattern (like exponential or Pareto) or even generating data from trace log files.

(b) Types of tasks:

User applications are modeled in objects called tasks. Three types of tasks or workloads are available in GreenCloud: Computationally Intensive Workloads (CIWs), Data-Intensive Workloads (DIWs) and Balanced Workloads (BWs) [14]. The comparison of the varying types in Table 7 is especially helpful when faced by the task of setting up user workloads.

(c) Power consumption features:

Power consumption efficiency is GreenCloud's primary target. Basic statistics are available for all data center components power consumption (computational and network resources). The equation used to calculate server power consumption is taken from the article by Rivoire et al. [25]. It calculates the power consumed as a function of the resource utilization of each resource type (Table 8). The equation introduced to calculate the power consumed by a switch is as follows:

$$P_{switch} = P_{chassis} + n_{linecards} + P_{linecard} + \sum_{i \in R} n_{ports,r} + P_r$$

$P_r$ represents power consumed by a port running at a rate valued r. GreenCloud implements three energy efficiency techniques:

*(i)* Dynamic voltage and frequency scaling (DVFS);*(ii)* Dynamic power management (DPM); *(iii)* A hybrid scheme of both.

(d) Testing results:

We tested GreenCloud on our machine that has 8 cores and 64GB of memory. GreenCloud works on Linux Ubuntu. The results shown in Fig. 4 are for a relatively small problem (144 servers) and it took around a minute (65 s). When moving from the testing architecture (3-tier-debug) to the two larger architectures (3-tier and 3-tier-high-speed) that contain 1536 servers while keeping the other factors without change, the simulator times increase vividly to around 20 min. Execution times are shown in Table 9. Clearly, GreenCloud commands more execution time than CloudSim. This is due to multiple design factors including NS-2 dependency and using the packet network model. A detailed representation of the multiple power consumption metrics is displayed as soon as the simulation is over. A sample output is shown in Fig. 4.

*4.3.4. Network model*

As discussed earlier, GreenCloud offers multiple options for topologies to be used. In addition, Users can control transmission rates to support power saving with several options available (for GE links, 10 Mb/s, 100 Mb/s, and 1 Gb/s are available).

The authors set the task network demands to consist of three parts. First, the task size which is transmitted from the root node to the hosting server. This represents the task code or instructions along with the input data. Second, the data the task communicates with other servers in the data center. Third, the server transmits the task output to the client (represented by sending it to the root node).

Due to the lack of space, we will henceforth conduct a high level analysis of the remaining simulators. We will illustrate the standing out aspect of each simulator in terms of purpose or functionality. We will also take a look at the scalability of the rest of the simulators at the end of Section V.

**Table 7**
Types of workloads/tasks at GreenCloud.

| | Computationally Intensive Workloads (CIWs) | Data-intensive workloads (DIWs) | Balanced workloads (BWs) |
|---|---|---|---|
| Real life application modeled | High-performance computing (HPC) applications that solve advanced computational problems | Applications like video sharing (for each request there is a streaming process.) | Applications with computing communication requirements (geographic information systems) |
| Computational vs. network load | High computing load, low network load | No computing load, high network load | Load the computing servers and communication links proportionally. |
| Scheduling central point | Server energy efficiency is critical (server consolidation) | No network congestions | Network becomes the bottleneck constant exchanged feedback is a must- Both computational and network portions. |

**Table 8**
A summary of GreenCloud features.

| Feature | Available? | Details |
|---|---|---|
| Ability to model user requests | Yes | CIWs,DIWs,BWs |
| Ability to model inter-VM dependency | No | Can configure Tasks to communicate with a random server |
| Ability to model multiple DCs | No | – |
| Ability to model servers | Yes | With fixed set of attributes |
| Ability to model network elements | Yes | Three different topologies available switches and links modeled packets sent over the network |
| Ability to model VMs | Yes | Resource configuration and placement |
| Ability to model inter-VM connectivity | Yes | Can configure tasks to communicate with a random server |
| Ability to model failures/recoveries | No | No mention of recovery in their documentation |
| Ability to model energy power sources mix (24 h source types | No | |
| Ability to model power usage per VM, server, facility | Yes | Multiple power management methods are included Basic power usage statistics are available |
| Ability to model security measures (attacks, firewalls)? | No | – |
| Ability to model network flows? | Yes | It is based on NS2, TCP/IP enabled. |

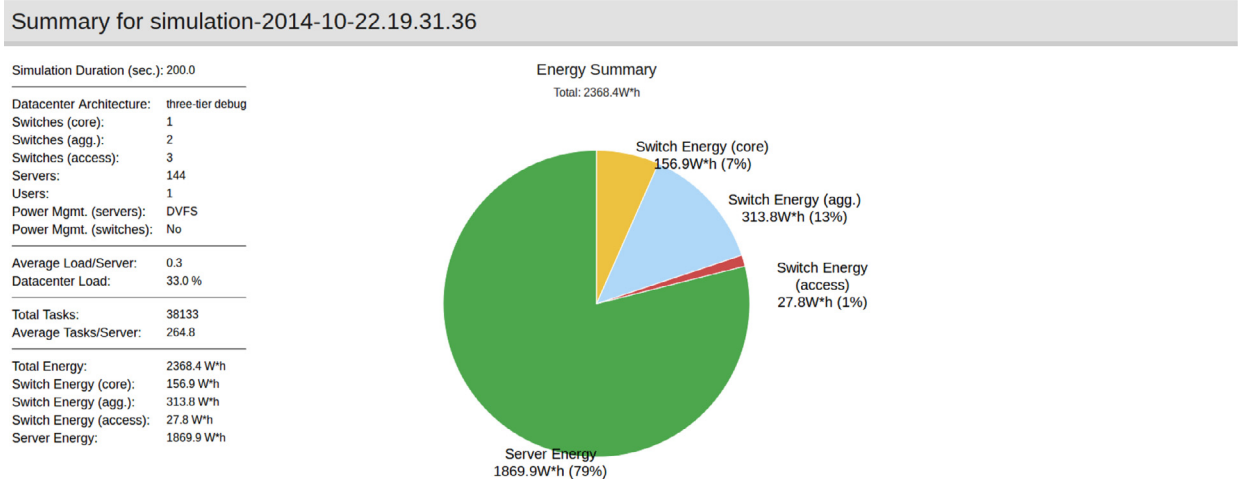



**Fig. 4.** GreenCloud output.

**Table 9**
A sample of GreenCloud execution times for different architectures with a data center load of 70% .

| Architecture | Server count | Execution time |
|---|---|---|
| 3-tier-debug | 144 | 56 s |
| 3-tier-hi-speed | 1536 | 19 min 8 s |
| 3-tier (default) | 1536 | 19 min 20 s |

### 4.4. iCanCloud

#### 4.4.1. Description and motivation

The designers of iCanCloud used Simcan framework to build their cloud simulator [26]. They have built a full GUI where users can configure, manage and run VMs, data centers and experiments for different cloud scenarios. Simulating distributed applications is supported through the GUI as it enables use to manage a repository of preconfigured VMs, manage a repository of pre-configured Cloud systems, manage a repository of pre-configured experiments, launch experiments from the GUI, and generating graphical reports. The challenge of predicting and structuring cost in cloud data centers is well studied.

#### 4.4.2. Main features

Apart from the basic cloud simulation, iCanCloud combines a set of features that stand out among cloud simulators. iCanCloud simulates the hypervisor module used by cloud providers. The hypervisor module handles brokering between cloud clients who send jobs and cloud data centers where these jobs can be served. Data centers represent a set of VMs, each one configured with

pre-defined features such as CPU, storage, memory, and network. Cloud clients in iCanCloud represent entities that submit a set of jobs to be executed on specific VM instances. Those submissions arrive directly to the hypervisor module. The hypervisor manages the brokering process by insuring efficient distribution of the jobs to the data centers and VMs inside them. This is done using preset brokering policies which can be amended by customization. The available brokering policies are cost-based policies, execution time-based policies and resource allocation-based policies. The hypervisor also handles managing VMs and the execution of the jobs on VMs, and defining cost policies for each VM instance type. This aids in integrating and testing brokering policies.

Moreover, features like flexibility in modeling architectures and VMs that represent single or multi-core hosts are supported. Storage gets a special attention in iCanCloud as it supports models for local or remote storage systems as well as parallel storage and RAID designs. In regards to the development environment, POSIX API and an adapted MPI library is available to build and run simulation experiments. In their article [26], the authors explain that they experimented on an application named Phobos where processors are used to calculate the trajectories of Phobos, the Martian moon, over a tracing interval. This is done by dividing the overall tracing interval in identical subintervals, each of them executed by a different task. The same task load distribution was simulated on iCanCloud and produced the same results in terms of the calculated performance cost.

### 4.5. TeachCloud

#### 4.5.1. Description and motivation

TeachCloud was built for a specific purpose, namely education [27]. TeachCloud shows a GUI through which researchers or industrial engineers can configure and perform experiments based on cloud scenarios.

#### 4.5.2. Main features

TeachCloud is an extension of CloudSim. On top of the CloudSim Framework, the designers added a GUI and a module that generates cloud related workload. Architectures like VL2, BCube, Portland and Dcell are supported. The simulator contains modules that monitor data center components, show the impact on system effectiveness and allow reconfiguration of the experiments.

### 4.6. GroudSim

#### 4.6.1. Description and motivation

GroudSim is a discrete event simulator that runs simulations specific to scientific applications on grid and cloud environments. The simulation core processes are performed by SimEngine. Groups of events can be created using features of the SimEventReference [29].

#### 4.6.2. Main features

GroudSim can be integrated into ASKALON environment [30] and that enables users to import experiments that represent real applications from that environment. Although GroudSim's vision involves executing jobs for the grid, it still includes basic cloud computing features like task implementation, cost estimation and resource loading and scheduling. Failures of grid sites and data transfer between cloud resources are features that can be seen in GroudSim.

However, not many bases are covered on the network side. In addition, performance issues arise when the scale of the experiment reaches hundreds of nodes.

### 4.7. CloudAnalyst

#### 4.7.1. Description and motivation

CloudAnalyst is another extension of CloudSim. The motive behind CloudAnalyst is analyzing and evaluating geographically distributed user workloads [31]. Requests that come from distant locations with heterogeneous distributions and sizes constitute a challenge when cloud providers plan their cloud deployments solutions. These scenario is also challenging when choosing the resource allocation policies for the data centers. In this case, the quality of service received by multiple users distributed all over the map is evaluated while varying the experiment parameters.

#### 4.7.2. Main features

Through CloudAnalyst, experiment conditions can be collected. The use of Java Swing to extend CloudSim makes it easier to extend CloudAnalyst. Additionally, CloudAnalyst provides a GUI where users specify the experiment parameters, the data center distribution and user location along with the network setup.

#### 4.7.3. Configurable parameters

CloudAnalyst offers a wide range of configurable parameters. All CloudSim parameters are included in Table 10. In addition, Internet characteristics, simulation configuration, Data center configuration and user distribution/load parameters are supported. The output from CloudAnalyst compare the response times/user experience from different geographically distributed bases.

### 4.8. CDOSim

#### 4.8.1. Description and motivation

Cloud Deployment Simulator (CDOSim)'s purpose is, as the name indicates, optimizing cloud deployment options. This includes measuring delays, Service Level Agreement violations and the subsequent costs based on a specific cloud deployment option from a client perspective. CDOSim provides features that support client decision making in terms of choosing the cloud provider, the runtime deployment policies and VM resource configuration.

#### 4.8.2. Main features

CDOSim enables cloud clients to compare the cost and effectiveness of a specific cloud solution with those of other solutions. This aims at mitigating the cloud clients' lack of knowledge over cloud platform options. Features like real life user trace integration and independence of programming languages are available [32].

### 4.9. MDCSim

#### 4.9.1. Description and motivation

Multi-tier Data Center Simulator (MDCSim) was built to constitute a scalable platform to conduct system evaluation and power consumption measurements for cloud environments. The authors demonstrated the abilities of the simulator using studies that included three types of applications [33].

#### 4.9.2. Main features

As a general purpose cloud simulator, MDCSim support data center component definition and configuration. Moreover, it includes features related to analyzing and evaluating of Infiniband Architecture (IBA) and 10GigE performance under varying cluster sizes, network load, and with different tier configurations. It also supports basic power measurement and configuration features. Still, more work is required to enhance the network model and strengthen power efficiency features for network component. In addition, the proprietary nature of MDCSim has affected its market penetration and extensibility.

**Table 10**
CloudAnalyst test parameters.

| Simulation parameters | Environment parameters |
|---|---|
| Internet characteristics<br>1-Region/region delay matrix : transmission delay between regions in (milliseconds)<br>2-Region/region bandwidth matrix (available bandwidth between regions) | User parameters<br>1-User grouping factors in user bases (how many simultaneous users from the same base)<br>2-User grouping factors in data centers (how many simultaneous users on the same host)<br>3-Instruction length/ request (bytes)<br>4-Load balancing policies a-round robin b-equally spread current execution load c-throttled |
| Simulation configuration<br>1-Simulation time<br>2-User bases: a-average number of users at peak time b-average number of users at off-peak time c-region d-request/user/hour and request size. e-daily peak hours<br>3-Broker service policy: a-closest data center b-optimize response time c-reconfigure dynamically | Data center configuration<br>1-Region<br>2-Architecture<br>3-OS<br>4-VMM<br>5-VM cost<br>6-Storage cost<br>7-Data transfer cost<br>8-Physical hardware units<br>9-Server configuration: a-memory b-storage c-BW d-number of processors e-processor speed f-VM policy (time shared or space shared) |

### 4.10. GDCSim

#### 4.10.1. Description and motivation

Green Data Center Simulator (GDCSim) is another simulator whose central purpose is reaching the completely green cloud. The designers aim at creating a framework to evaluate new resource allocation policies or energy efficiency techniques. Conditions to be tested include topologies, workload distribution, platform power management schemes, and scheduling algorithms. In fact, both statements are correct. GDCSim contains a CFD component and its contribution and results are validated against traditional CFD simulators. The authors of GDCSIm clarify that GDCSim -when used in the context of cloud data center energy efficiency analysis- has four major advantages compared to a typical CFD simulator. Those are automated processing, online analysis, and more importantly, workload management and cyber physical interdependency. Compared to other cloud simulators, GSCSim is an example of a simulator with more focus on the thermal side of energy efficiency. Feedback on temperature and air flow patterns in the data center is used by the management algorithms.

To avoid the challenge of high complexity and slow execution of CFD simulation, the authors opted to use the CFD component GDCSim is built upon, BlueSim, only for a partial task. The CFD simulator module, BlueSim, was used to generate the thermal map of the data center and was not used in conjunction with the resource management module. Furthermore, GDCSim was validated against established Computational Fluid Dynamics (CFD) simulators like Flovent CFDSim [34].

#### 4.10.2. Main features

GDCSim supports online analysis and adjustment as users can manage the simulation scenarios while the simulator is running. This gives users the chance to adjust and modify based on physical resource changes. GDCSim also supports thermal analysis features, in which the thermal conditions at a given moment are recorded and analyzed. Cooling policies can be tested in detail. Power modes can be controlled in terms of servers' status. An added feature here is considering "cyber-physical interdependency, which enables feedback of information on temperature and air flow patterns in the data center to the management algorithms and the closed loop operation of the servers and cooling units (CRAC) to achieve energy efficient operation" [34].

### 4.11. SPECI

#### 4.11.1. Description and motivation

Simulation Program for Elastic Cloud Infrastructures (SPECI) is another cloud simulation tool that allows studying large data centers closely [35]. SPECI studies the performance of the data center nodes as a unit. It focuses on the efficiency of the middleware policies used in a cloud data center. The authors define middleware here as the layer of software that handles job scheduling, load-balancing, security, virtual networks, and resilience. It is basically the management layer of the data center. SPECI focuses on handling communication policies between the simulated hardware components inside a data center specially in the case the number of these components (or nodes) increases sharply. These nodes communicate with each other and therefore need to be aware of each others' states. Each of these nodes can be functioning ("alive") or ("dead") and this state needs to be communicated to nodes that work with this node. SPECI offers an environment to simulate this process in order to evaluate communication protocols and data center load.

Basically, nodes (which can represent cloud hosts) are connected through a network with a configurable topology. Nodes are susceptible to random failures. Any component which cooperates with a set of other components, is thus interested in the aliveness of each of these components and performs queries to find about their states. This state retrieval gets more complicated and it causes more data exchange over the network as the number of components (nodes) increases.

Several architectures of state or heartbeat retrieval can be configured. Examples include having a central node sharing all the states, a hierarchical design where data is shared or a P2P data sharing model. SPECI aims at observing the behavior of the whole system under different architectures, setup parameters and protocols especially as the number of nodes scales up.

#### 4.11.2. Main features

SPECI is based on the SimKit framework [36] which offers the core simulation functionalities. The challenge faced in this scenario is scalability. The data center size increases and some middleware properties do not scale in a linear way with the number of components.

SPECI consists of two packages. The first one deals with data center architecture and topology. The second contains the modules dealing with simulation and measurements. Status updates

are critical to the designers as they aim to evaluate the impact on component failure and policy changes. In terms of failure observance, the notions of hardware components being alive or dead are presented. These states are exchanged in a subscription based scheme. When the simulator is initialized, an object is created for each node and network link in the data center. Each node is subscribed to the states of a list of other nodes based on a predefined distribution. Next, the communication policy is loaded and the rest of the simulation is run by the event queue.

### 4.12. BigHouse

#### 4.12.1. Description and motivation

BigHouse is another simulator with some unique features. BigHouse developing team introduced it as a simulator with a higher degree of abstraction. This is coupled with a focus on the statistical view and execution time minimization. The motivation is to use simulation to approximate the performance of complex queuing models like G/G/1 or G/G/k queues (generalized inter-arrival and service time distribution with 1 or k servers).

#### 4.12.2. Main features

In BigHouse, like most of the main simulators, tasks are represented by random variables that describe their parameters including resource requirements, arrival and service time details. Examples of the workloads used include Departmental DNS and DHCP server under live traffic, Departmental POP and SMTP server under live traffic, Shell login server under live traffic, executing a variety of interactive tasks, Leaf node in a Google Web Search cluster, HTTP server under live traffic. Once workload distributions have been generated, a system model in terms of power consumption and performance distribution is specified and then estimate results can be derived.

BigHouse is based on the stochastic queuing simulation (SQS) methodology. This deals with the requests/tasks as high level components that run in time in the order of milliseconds instead of on an instruction by instruction level. Consequently, "BigHouse can simulate server systems in minutes rather than hours" [37]. Architecturally, BigHouse contains two main models. The first one handles constructing the data center components and the event sequences like any discrete event simulator. The second one mainly contains the reporting tools and handles the statistical modules. The authors have shown some results showing the parallel performance of BigHouse which is interesting in terms of scalability of the simulator [37]. Running time can be tweaked by changing the limit of accuracy or confidence in results.

A notable limitation to BigHouse is the network model. The examples previously mentioned model client-server scenarios. More realistic models (including a step to three tier topology for instance) are not included unless changes to the simulator are made.

## 5. Simulator comparison

When looking at the previous efforts, we can divide them into two sets:

### 5.1. Simulators with a specific purpose or working on a limited scale

This purpose could be educational like TeachCloud or related to a specific functionality like CDOSim. We note here that some of the simulators are not only developed by academic teams but also are meant to serve academic purposes mainly. In the case of some projects built for academic purposes, they do not generate much traction or get continuous attention from the developing team on the medium to the long term. The team's focus on the project really starts to diminish after achieving publications or other academic purposes. This is something that should be closely investigated by a potential user. This issue could affect factors like documentation, upgrades and active community around the project.

### 5.2. Simulators operating on a more general scale

This set includes simulators that include features in which users can test general cloud computing problems and solutions. The criteria here are: the simulator's extensibility, the simulator containing the critical scheduling and energy efficiency recording abilities and the simulator including a rich networking model. In Tables 11 and 12, a comparison of the general features, strength and limitations of each of the simulators discussed previously can be found.

### 5.3. When do I need a new simulator?

The major reasons that might push a research or an industrial team towards the choice of building a new cloud simulator include one of the following points.

#### 5.3.1. Scalability

There is need for a simulator that is proven to work for large scales reaching data centers with a size in the order of 10,000 to a 100,000 nodes(hosts). Intense comprehensive testing is a key requirement. We have taken a look at the scalability properties of CloudSim and GreenCloud. More on the scalability of the rest of the surveyed simulators in the following subsection.

#### 5.3.2. Portability

Many available simulators are subject to platform, environment, and hardware limitations. These limitations need to be examined and the new simulator should consider all the environments/conditions that may be called upon in the required future work.

#### 5.3.3. Customizability

An in-house developed simulator offers advantages in terms of customization to the ever expanding needs of R&D departments.

#### 5.3.4. Training and knowledge transfer

An in-house simulator also offers an advantage in terms of training/knowledge transfer over other simulators. This covers even open source simulators as documentation and support challenges usually arise.

### 5.4. A look at simulators' scalability

By examining each of the cloud simulators' experiments [13,26,27,29,31–35,37], we gain a sense of the scale each simulator works on. For iCanCloud for example [26], the simulator was tested for experiments to run up to 250,000 jobs on up to 5000 VMs. Those experiments used jobs whose input size is 5 MB, output size is 30 MB, and processing length is 1,200,000 MI. VMs' computing capacity was up to 9500 MIPS, which simulates a standard small instance type provided by Amazon EC2. The largest experiment finished in about 10,000 s (2 h 45 min).

For NetworkCloudSim [13], the experimental setup contained four servers, eight VMs and up to eight processes per VM. An MPI application was modeled, with the main process generating several random numbers and then sending the data to all other processes. Each host has two Xen VMs, each one with 2 cores and 1.5 GB RAM. All the hosts are connected by a 100 Mbps switch. A varying number of communication messages up to 1,000,000 MPI INT elements was transferred from one process to another. This took

**Table 11**
A comparison of the cloud simulator scalability using solved problem sizes.

| Simulator | Number of Nodes | Experiment Parameters | Execution Time |
|---|---|---|---|
| iCanCloud | 5000 VMs | 250,000 jobs, input size is 5 MB, output size is 30 MB, processing length is 1,200,000 MI, VMs computing capacity up to 9500 MIPS | 10,000 seconds (2 hours and 45 minutes) |
| NetworkCloudSim | 4 servers, 8 VMs and up to 8 processes per VM | Each host has 2 Xen VMs, VM has 2 cores, 1.5 GB RAM, Hosts have 100 Mbps switch, Communication messages up to 1,000,000 MPI elements, arrival rate of 200 requests per second | 2.5 seconds |
| CloudAnalyst | Up to 3 data centers containing up to 75 VMs | Total number of users ranges from 108 to 1,080, users make a request every 5 simulation minutes and each 100 request are gathered together, VM size is 100 MB, VMs have 1 GB of RAM, VMs have 10 Mbps of bandwidth, Servers have 2 GB of RAM and 100 GB of storage, Each machine has four CPUs, Each CPU has a capacity power of 10,000 MIPS, Each user request requires executing 25,000 instructions. Simulated period is one day | Overall average response time for a combined request 121.07 milliseconds (A total of 2 minutes) |
| GroudSim | On a number of cloud instances from 8 to 32,000. | Jobs ranging from 5000 to 32,000, 1000 MI per second (MIPS) for each CPU, 100,000 jobs, with minimal size | 16 seconds for 1,000,000 jobs of minimal size, 50 seconds per job consisting of 3500 activities (of fixed MI), 0.2-0.3 seconds per job for jobs consisting of 200 activities (of fixed MI ) |
| CDOSim | A data center ranging in sizes between 4 and 7 nodes . | Problem representation is slightly different, uses a workload intensity function that originates from a service provider producing 5000 requests (method calls) per minute | Around a day and a half in total |
| MDCSim | Up to 128 nodes in 3 tiers, communicating clients up to 6400 clients | Major components include clients, Web Server (WS) tier, Application Server (AS) tier and DB tier. Round-robin web switch emulation to distribute client requests to the web server tier nodes, message size 5 kb | Depending on the scenario the latency reaches around 0.9 seconds per client amounting to an experiment time of around an hour and a half |
| GDCSim | 2 rows of 5 industry standard 42U racks in each row, racks consisted of 5 7U chassis each | GDCSim's main focus is on the structural side of the data center in terms of the heat dustribution and other thermal factors not the task/job scheduling side. Therefore, data centers tested on GDCSim tend to be smaller. | 9,000 seconds experiment time (2 hours 30 minutes) |
| SPECI | 100,000 nodes | Simple message exchange (state=alive/dead), Heartbeat retrieval events drawn from a uniform distribution with a delay between 0.8 and 1.2 seconds and reschedule themselves with a delay from the same distribution, number of subscriptions = square root of the number of nodes,run for 3600 simulation time seconds, SPECI-2 shows 5.5 GB RAM JVM memory footprint when executed for similar experiments | hours |
| BigHouse | 10-10,000 servers | number of simulated events takes from 0 to 6,000, the experiment varies both processor and memory settings, the performance setting space is 2-dimensional; for each variation the number of maximum queries (request) per second (QPS) is increased on a scale of percentages of a preset maximum load, the measured result is the 95th percentile latency when applying for the specific variation and QPS | 10 servers cluster takes less than a minute, 10,000 server data center takes from hours up to a full day based on the scenario |

**Table 12**
A summary of the cloud simulators features: external view.

| Feature/ parameter | Underlying toolkit/ platform | Programming language | Availability |
|---|---|---|---|
| CloudSim | SimJava | Java | Open source |
| GreenCloud | NS2 | C++/TCL | Open source |
| iCanCloud | OMNET, MPI | C++ | Open source |
| MDCsim | CSIM | C++/Java | Commercial |
| NetworkCloudSim | CloudSim | Java | Open source |
| CloudAnalyst | CloudSim | Java | Open source |
| GroudSim | SimEngine | Java | Academic |
| TeachCloud | CloudSim | Java | Open source |
| CDOSim | CloudSim | Java | Academic |
| GDCSim | BlueSim | Java/XML | Academic |
| SPECI | SimKit | Java | Open source |
| BigHouse | Mac/ Linux/ Windows | Java/Python | Open source |

around 2.5 s. In terms of experimenting on scheduling policies, the same topology was used to test scheduling requests with an arrival rate of 200 per second.

For CloudAnalyst [31], the problem can be described as follows: up to three data centers containing up to 75 VMs receive connection requests from users distributed along 6 user bases around the world. The total number of users can range from 180,000 to 1,800,000 depending on peak hours. But that number is actually grouped by a factor of 1000. This means the actual number of users in the experiments translates to a number between 180 to 1800 users. The VM size is 100 MB. VMs have 1 GB of RAM and have 10 Mbps of bandwidth. Servers have 2 GB of RAM and 100 GB of storage capacity. Each machine has four CPUs, and each CPU has a computing power of 10,000 MIPS. A time-sharing policy is used to schedule resources to VMs. Each user request includes executing 250 instructions. Requests are also grouped by a factor of 100.

For GroudSim [28] and [29], a number of jobs ranging from 5000 to 32,000 was executed on a number of cloud instances from 8 to 32,000. The experiments were done for three types of jobs. In the first one, a higher number of jobs with minimal required resources was tested. In the second one, each job consisted of a 3,500 activities (tasks) of fixed length (measured in MI). In the third one, a job consisted of 200 activities only. "The type of cloud resources is not relevant to these experiments, as we are only interested in how fast a certain number of jobs on a certain amount of resources can be simulated, independent of their real execution time" [29]. If we look at CDOSim [32], the problem representation is slightly different as it uses a workload intensity function originating from a service provider for digital photos producing 5000 requests (method calls) per minute on a data center ranging in size between 4 and 7 nodes. The experiment took around a day and a half in total.

For MDCSim [33], the case study's major components included clients, WS tier, AS tier and DB tier. A simple round-robin web switch emulation was implemented to distribute client requests to the web server tier nodes. This included up to 128 nodes in the 3 tiers with a number of communicating clients up to 5600 clients. The exchanged message size was around 5 kb. In the GDCSim experiment [34], there were two rows of five industry standard 42U racks in each row, laid out in hot/aisle cold aisle configuration. These racks consisted of five 7U chassis each. The data center tends to be smaller due to GDCSim interest in the energy efficiency and thermal component of the data center operation.

As for SPECI [35], scaling is easier in terms of the number of components as it focuses on the component livelihood status rather than execution of complex requests. Simulations of data center networks of the size of 100,000 hardware components are possible.

Lastly, when looking at the scalability of BigHouse [37], it is found that simulation of a 10 servers cluster takes less than a minute. The time increases approximately linearly until biggest recorded size which 10,000 server data center where the simulation finishes in hours rather than days. These results are summarized in Table 11 in addition to experimental results shown for CloudSim in Tables 4 and 5 and for GreenCloud in Table 9.

### 5.5. Previous experiments and problems solved using surveyed cloud simulators

The previously discussed simulators have been used variably for multiple projects. Some of these projects were attempts to solve a specific problem or demonstrate certain cloud scenario using the simulator. Some of these projects were done by the same team with the purpose of building upon the simulator and some was done by different teams who used the simulators mostly under the open source licensing schemes.

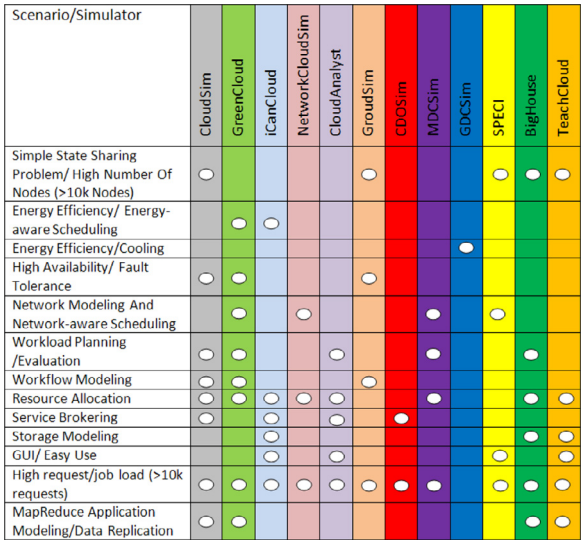



**Fig. 5.** Recommended cloud simulators depending on user priorities.

A look at the literature and the websites of active simulators would give us an idea of the sort of audience following these projects and the sort of extensions/solutions built upon them. In Table 14 we provide the list of topics and problems each simulator was used to evaluate or solve. This is another step towards helping the reader in the process of deciding which simulator is the most suitable to their project.

### 5.6. A summary of cloud simulators comparison

Finally, based on our analysis in this paper and based on multiple factors including scalability, main features, environments, and previous experiments, we have provided a general recommendation that would help the reader directly choose the most suitable simulator(s) to start from based on the requirements or purpose. These are summarized in Fig. 5.

However, looking at the nature of the topic and vast list of variations between every cloud related problem and the other, this recommendation should be looked at as a general guideline to the reader instead of being a definite (right or wrong) fact. It should not be forgotten that the nature of software development process is that with enough time and effort, a capable team can extend any software to include many more features.

## 6. Open research challenges

As we weigh pros and cons of building a new simulator versus using a current one, research challenges, and learned lessons present themselves. We try to summarize these in the following points to serve as a blueprint of issues that need attention or open research topics.

### 6.1. Lose the grid perspective

It is common for simulators that were based on previous grid simulators to inherit much functionality. The point here is not inheriting the vision too. The grid model is principally different than the cloud. The job submission and waiting model does not apply to the cloud. The cloud has more of an interactive environment of leased machines. Simulators based on the grid should be revised to adopt new packaging that is based on unified commercialized packages of PC-like resource components. This is more accurate

**Table 13**
A summary of the cloud simulators features:strengths and limitations.

| Simulator/Parameter | Focus/ strength | Limitations |
|---|---|---|
| CloudSim | Most commonly used (many extension projects)<br>Supports large scale problems Inclusive support for DCs, VMs and resource provisioning techniques. | Communication model on the packet level is not supported directly |
| GreenCloud | Focus on power management and energy consumption techniques testing<br>Packet level communication supported | Runtime<br>Detailed brokering model |
| iCanCloud | Trade-off between costs and performance.<br>Full GUI<br>Storage modeled in more detail | Cost policies focus is on pay-as-you-go policies<br>No focus on energy consumption<br>No full network model |
| MDCsim | General purpose functionalities available | Commercial<br>No full network model |
| NetworkCloud -Sim | Extension of CloudSim<br>Extra communication features like message passing<br>Full application model | No networking on packet level.<br>Less focus on power/cost models<br>Issues of CloudSim apply |
| CloudAnalyst | Has a full GUI<br>Focus on geographical factors | Focus on Specific Purpose.<br>No full network model |
| GroudSim | Works for Grid and cloud systems.<br>Works for large scale problems (high number of requirements)<br>General features available | Basic Network functionalities<br>Basic power consumption optimization functionalities |
| TeachCloud | For Educational purposes.<br>Simple to use GUI | Basic features<br>Only for academic purposes |
| CDOSim | Focus on Deployment Options<br>Extension of CloudSim | Unproven for general purposes and on a large scale |
| GDCSim | Focus on low level power saving methods and cooling | Unproven for general purposes and on large scales |
| SPECI | Focuses on component status update exchange and middleware policies | Does not cover computational resource reservation or variety of cloud applications, focused on the number of failures in a DC |
| BigHouse | showcases queuing models, parallel performance, result accuracy can be traded off with runtime | offered network model is limited |

**Table 14**
Topics and problems cloud simulators were used to tackle .

| Cloud Simulator | Topic/Problem |
|---|---|
| CloudSim | Cloud workflow preparation and execution, allocation based on mixed-integer programming, VM dynamic performance change, auction-based services, cloud provider migration planning, web session modeling, MapReduce modeling simulation [38], fault tolerance [39] |
| GreenCloud | Power efficiency, communication models and architecture evaluation[40,42–46,49], data replication solutions [41], Network-as-a -Service (NaaS) implementation [48], scheduling for opportunistic grids [50], high availability in the cloud [47] |
| iCanCloud | Efficient service brokering [51], storage modeling [52], live migration [53], energy efficiency [54] |
| NetworkCloudSim | Cloud network modeling [55], cloud resource allocation [56,57], |
| CloudAnalyst | Service brokering [58,60] , load balancing polices [59,61] |
| GroudSim | Efficient scheduling of scientific workflows [24] and [62–64], fault tolerant execution of scientific workflows [65] |
| CDOSim | Cloud deployment options [66] , cloud provider migration [67] |
| MDCsim | Cloud resource allocation [68] and [70], malicious activity detection through predictive modeling [69] , |
| GDCSim | Energy efficiency and cooling inside a data center [34] |
| SPECI | Component subscription networks monitoring [71,72], hierarchical cloud network modeling [73], |
| BigHouse | Cloud workload modeling and planning [74–76] , resource allocation [74,76], MapReduce modeling simulation [77] |
| TeachCloud | Resource allocation [78,80], MapReduce modeling simulation[79] |

to the cloud client expectation than the distributed cluster/super computer assumptions of grid clients. This will generally affect the way resources are modeled, service packaging and pricing, user data modeling, and resource scheduling and allocation in the cloud simulator. The user experience in the cloud is more engaging; reliability and availability is a concern on the scale of a millisecond.

### 6.2. More work towards realistic user application patterns

Although multiple current simulators allow simulator users to test using any data sets they require, that does not seem to be enough. More work is required towards designing the user application models and data sets. We have seen efforts discussing the âÇ£typesâÇ¥ of applications user request data specify in terms of being communication intensive or computationally intensive (Table 13). However, there is certainly a demand for a detailed application stack that represents interacting software components. The generic data exchange request or computational request does not represent the dynamic process that is a full user application.

For example, the four indicators of Big Data: volume, velocity, variety and veracity [1] have a big say in specifying the final form of the client application. The impact will not stop at the real client data though; it will spread to affect simulated use cases, request size, percentage of error and application components distribution. The simulator designers will have to put into consideration more complex scenarios where the degree of interdependency will reach a new level. Security considerations under Big data conditions are another topic that will influence this process heavily.

### 6.3. Cloud deployment and pricing

Deployment options and pricing packages are covered with the current set of simulators. Brokering is offered through CloudSim. The primary objective there is choosing the provider that satisfies client's resource request and service conditions with minimal price. Assuming a client can move between providers freely is not realistic in this case. Provider lock-in is a well documented issue that can cause client reluctance. The complications of moving between providers on the technical side (delay, portability, etc) and business side (release clauses, notice time constraints, price difference) should be considered in an ideal cloud simulator.

**Table 15**
Research challenges to be tackled by future cloud simulators.

| Research challenge | Areas most included/affected |
|---|---|
| Lose the grid perspective | Resource modeling, service packaging, pricing, user data modeling, resource scheduling, reliability |
| More work towards realistic user application patterns | Application models, user data modeling, data exchange design, network model |
| Cloud deployment and pricing | Portability, brokering, user geographical distribution, cloud dynamic pricing, user SLAs |
| Deeper look at reliability and high availability | Component failure, repair and recovery, Redundancy, interdependency, user SLAs |
| Widen your horizons: outside the data center | Network model, user geographic distribution, power consumption, power sources, security |
| Exploiting High Performance Computing features | Data center topology, resource modeling, service pricing, user request modeling |
| Modeling Cloud Storage | Redundancy, storage distribution, resource modeling |
| Modeling Containers along with VMs | Resource modeling, performance monitoring |

### 6.4. Deeper look at reliability and high availability

Every hardware or software component is bound to fail sometime. Minimizing this time is a must in an environment where the 5 nines guarantee (99.999% of the time availability) is becoming a precondition sooner rather than later [90]. A deeper, more structured look at simulating component failure, repair, recovery and their potential effect on the applications/services is required. Providers will prefer any simulator with the opportunity to simulate and test redundancy policies as well. This does not only mean storage components, but also servers, racks, VMs, and software components. Moreover, availability of network components should be included as part of a more inclusive simulation scenario.

### 6.5. Widen your horizons: outside the data center

It is noticed that most of cloud simulators do not consider the source of the cloud client data outside the data center. This is due to the fact that cloud clients access data center through the Internet. However, with the diverse set of connections seen in a cloud scenario [91], it would be interesting to study the effect of the connection method. Will the client connect from their private clouds? From a hand held device connected to a base station? From a PC? The client geographic distribution might have an effect as well. Enabling the simulator users to implement this sort of layout will add great value. The source of energy could have an effect on the consumption evaluation. It would be interesting to know the effects of alternating between sources of energy that differ in their "Greenness" as the amount of energy allowed to be used can change also.

### 6.6. Exploiting high performance computing features

Building a new simulator gives the chance to investigate HPC technology features. HPC offers robust and scalable high computing power methods including using a hybrid platform of CPUs/GPUs and modular design concepts. HPC technologies are not modeled in any of the aforementioned cloud simulators despite some of them supporting distributed architecture (Table 15).

As seen in the previous sections, most of the simulators model the computing power either as:

- Reserved/not reserved processing unit (termed time shared)
- A million instruction per seconds (MIPs) commodity resource (termed space shared)

In the second method, requests or VMs reserve some of this capacity. This is understandable because it is easier to treat a resource as a simple integer.

However, a more sophisticated model of processing units would be beneficial. It would be interesting to have a simulator that models a cloud facility that offers HPC capabilities to clients through the cloud. The major impact of implementing HPC in the cloud would come from the flexibility offered by virtualization. A cloud setup offers the ability to customize the virtual machine as per the scientists' specific needs offered by a cloud setting compared to the strictly-preserved system software in traditional HPC offerings. In the cloud simulator, the ability to model these differences would assist HPC system designers or supercomputer designers in tackling challenges like pricing, scheduling model changes from their traditional grid-like model (submit a job and wait) and other traditional cloud challenges customized to the specific environment of supercomputing (security, portability, etc).

In his article on high performance computing in the cloud [81], Milojicic surveys some of the previous efforts tackling the challenges of coupling HPC and the cloud. Studying and solving these challenges would be assisted by a simulation tool that precisely represents HPC resources and typical applications. This should include modeling features of the application like the percentage of code that can be executed in a parallel setting.

### 6.7. Modeling cloud storage

Cloud storage in general has not gained enough attention by the surveyed simulators. In GreenCloud, for example, storage is considered to have fixed capacity for a resource provider (host, or VM) and a fixed demand by the task (just a number). "File size" is considered in CloudSim, while some simulators do not consider storage whatsoever. iCanCloud has a more detailed storage modeling among the current simulators as it supports models for local or remote storage systems as well as parallel storage and RAID designs. Missing a full network representation makes the model incomplete in this case. Cloud storage modeling should be included in a cloud simulator at large. This would cover the storage components distribution, redundancy and direct and indirect connection between computational resources like hosts and storage resources. User data storage is a critical part of the application performance and a solution cannot be evaluated without considering data locations and connectivity. The impact on pricing is also an area that is continuously drawing the attention of cloud architects and therefore should be included in a cloud simulator feature list. In addition, modeling the specific database solutions implemented in the cloud would be a valuable addition that would help understand solution performance.

### 6.8. Modeling containers along with VMs

Containers are an abstraction performed at the operating system (OS) level that promises more efficient use of the hardware resources over VMs. Containers are gaining traction in the cloud providers' circles as a replacement for VMs either fully or for a specific set of applications and use cases. When using containers, the user space gets abstracted instead of the whole hardware stack like in the VM case [82–84]. The overhead produced by running the whole operating system is not incurred every time when using multiple containers so this saves on memory and CPU compared to VMs running the same workloads. Doubts over security and efficient management are still there for containers and a balance of when to use them vs. when to use VMs is still materializing. However, none of the currently available cloud simulators offer the option to model applications inside containers covering

the resource allocation options along with performance comparison between both solutions.

## 7. Conclusion

Cloud providers are under constant pressure to deliver highly reliable and continuously inventive service. The cutting edge in this market will come from better performance metric values or newly added services that clients cannot find somewhere else. This requires a strong cloud simulation comprehensive solution. This cloud simulator would perform roles that range from defining the problems to pinpointing bottlenecks and from evaluating policies to testing the solution endurance and scalability. The process of building a simulator includes multiple design decisions and requires specifying the shape of many simulator ingredients. We have discussed these ingredients and introduced our vision for the simulator design framework. We then traversed a select set of common cloud simulators, stressing the main features and limitations within simulator environments.

Open research challenges and areas/topics that are in need of attention from the scientific community were compiled. Covering all these topics along with any new challenges related to security, and power consumption will enhance the cloud simulator position as a critical tool for cloud providers. As this area keeps growing, a cloud simulator with proven efficiency and effectiveness will continue to draw the attention both of the industry and academic parties.

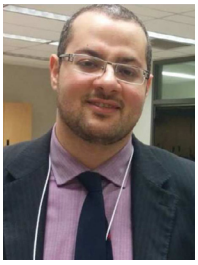
## Acknowledgments

The authors would like to express their gratitude to the Natural Sciences and Engineering Research Council of Canada (NSERC) for supporting this work. The authors would like to thank GreenCloud developing team represented by Mateusz Guzek for promptly answering questions. The authors would also like to thank the CloudSim team for an informative forum and group website.

## References

[1] The big data and analytics hub, IBM, Nov. 2015. The 4 Vs of The Big Data, [Online]. Available from: http://www.ibmbigdatahub.com/Infographic/four-vs.-big-data.
[2] M. O'Neil, Big data facts, myths and eventualities: 8 insights and implications, Jan. 2015. [Online]. Available from: http://www.insightaas.com/big-data-facts-myths-and-eventualities-8-insights-and-implications.
[3] W. Zhao, Y. Peng, F. Xie, Z. Dai, Modeling and simulation of cloud computing: a review, in: Cloud Computing Congress (APCloudCC), 2012 IEEE Asia Pacific, Nov. 2012, pp. 20–24.
[4] U. Sinha, M. Shekhar, Comparison of various cloud simulation tools available in cloud computing, Int. J. Adv. Res. Comput. Commun. Eng. 4 (Mar.(3)) (2015).
[5] S.J. Mohana, M. Saroja, M. Venkatachalam, Analysis and comparison of simulators to evaluate the performance of cloud environments, J. Nanosci. Nanotechnol. 2 (Feb. (6)) (2014).
[6] M.A. Sharkh, M. Jammal, A. Ouda, A. Shami, Resource allocation in a network-based cloud computing environment: design challenges, Commun. Mag. IEEE 51 (Nov. (11)) (2013) 46–52.
[7] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software 41 (1) (2011) 23–50.
[8] R. Miller, Inside amazon cloud computing infrastructure, Dec. 2015. [Online]. Available from: http://datacenterfrontier.com/inside-amazon-cloud-computing-infrastructure/.
[9] R. Miller, Google data center FAQ, Dec. 2015. [Online]. Available from: http://www.datacenterknowledge.com/google-data-center-faq-part-3/.
[10] R. Miller, Report: Google uses about 900,000 servers, Dec. 2015. [Online]. Available from: http://www.datacenterknowledge.com/archives/2011/08/01/report-google-uses-about-900000-servers/.
[11] R. Miller, Rackspace now has 70,000 servers, Dec. 2015. [Online]. Available from: http://www.datacenterknowledge.com/archives/2011/05/10/rackspace-now-has-70000-servers/.
[12] Rackspace, Global infrastructure and uptime guarantee, Dec. 2015. [Online]. Available from: http://www.rackspace.com/about/datacenters/.
[13] S.K. Garg, R. Buyya, Networkcloudsim: modeling parallel applications in cloud simulations, in: 4th IEEE International Conference on Utility and Cloud Computing, 2011, pp. 105–113.
[14] D. Kliazovich, P. Bouvry, S.U. Khan, Simulation and performance analysis of data intensive and workload intensive cloud computing data centers, in: C. Kachris, K. Bergman, I. Tomkos (Eds.), Optical Interconnects for Future Data Center Networks, Springer-Verlag, New York, USA, 2013. ISBN: 978-1-4614-4629-3, Chapter 4.
[15] Amazon, Amazon elastic compute cloud (amazon EC2), Feb. 2016. [Online]. Available from: http://aws.amazon.com/ec2/.
[16] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, S. Lu, Dcell: a scalable and fault-tolerant network structure for data centers, in: The ACM SIGCOMM 2008 Conference on Data Communication, Aug. 2008.
[17] C. Guo, et al., BCube: a high performance, server-centric network architecture for modular data centers, ACM SIGCOMM Comput. Commun. Rev. 39 (Oct. (4)) (2009).
[18] A. Singla, C. Hong, L. Popa, P.B. Godfrey, Jellyfish: networking data centers, randomly, in: The 3rd USENIX Conference on Hot Topics in Cloud Computing, Jun. 2011. 12–12.
[19] C.E. Leiserson, Fat-trees: universal networks for hardware-efficient supercomputing, IEEE Trans. Comput. 34 (Oct. (10)) (1985) 892–901.
[20] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, in: The ACM SIGCOMM 2008 Conference on Data Communication, Aug. 17–22, 2008. Seattle, WA, USA.
[21] B. Wang, et al., A survey on data center networking in cloud era, Submitted to, Commun. Surv. Tutor. J. (2013).
[22] R.N. Calheiros, R. Ranjan, C.A.F. De Rose, R. Buyya, CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructure and Services, Technical report, grids-tr-2009-1, Grid Computing and Distributed Systems Laboratory, the University of Melbourne, Australia, 2009.
[23] D. Kliazovich, P. Bouvry, S.U. Khan, Greencloud: a packet-level simulator of energy-aware cloud computing data centers, J. Supercomput, Special Issue on Green Networks 62 (Dec.(3)) (2012) 1263–1283.
[24] G. Kecskemeti, S. Ostermann, R. Prodan, Fostering energy-awareness in simulations behind scientific workflow management systems, in: The 2014 ACM 7th International Conference on Utility and Cloud Computing, 2014, pp. 29–38.
[25] S. Rivoire, M. Shah, P. Ranganatban, C. Kozyrakis, J. Meza, Models and metrics to enable energy-efficiency optimizations, Computer 40 (12) (2007) 39–48.
[26] A. Nunez, J.L. Vazquez-Poletti, A.C. Caminero, J. Carretero, I.M. Llorente, Design of a new cloud computing simulation platform, in: The International Conference on Computational Science and Its Applications, in: LNCS, 6784, 2011, pp. 582–593.
[27] Y. Jararweh, Z. Alshroa, M. Kharbutli, M. Jarrah, M. Alsaleh, Teachcloud: a cloud computing educational toolkit, Int. J. Cloud Comput. (IJCC) 2 (2) (2013) 237–257.
[28] S. Ostermann, K. Plankensteiner, R. Prodan, T. Fahringer, Groudsim: an event-based simulation framework for computational grids and clouds, in: Lecture Notes in Computer Science, Euro-Par 2010 Parallel Processing Workshops, 6586, 2011, pp. 305–313.
[29] S. Ostermann, K. Plankensteiner, R. Prodan, T. Fahringer, Groudsim: an event-based simulation framework for computational grids and clouds, CoreGRID/ERCIM Workshop on Grids and Clouds, Springer Computer Science Editorial, Ischia, 2010.
[30] T. Fahringer, et al., ASKALON: a grid application development and computing environment, in: 6th IEEE/ACM International Conference on Grid Computing, 2005, pp. 122–131.
[31] B. Wickremasinghe, R.N. Calheiros, Cloudanalyst: a cloudsim-based visual modeler for analyzing cloud computing environments and applications, in: 24th International Conference on Advanced Information Networking and Application, 2010, p. 446.
[32] F. Fittkau, et al., CDOSim: simulating cloud deployment options for software migration support, in: IEEE 6th International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2012, pp. 37–46.
[33] S.H. Lim, B. Sharma, G. Nam, E.K. Kim, C.R. Das, MDCSim: a multi-tier data center simulation, platform, in: IEEE International Conference on Cluster Computing and Workshops, 2009, pp. 1–9.
[34] S.K.S. Gupta, R.R. Gilbert, A. Banerjee, Z. Abbasi, T. Mukherjee, G. Varsamopoulos, GDCSim: a tool for analyzing green data center design and resource management techniques, in: Green Computing Conference and Workshops (IGCC), 2011, pp. 1–8.
[35] I. Sriram, SPECI, a simulation tool exploring cloud-scale data centers, in: 1st International Conference on Cloud Computing (CloudCom09), in: LNCS, 5931, Dec. 2009, pp. 381–392.
[36] A. Buss, SimKit: component based simulation modeling with SimKit, in: 34th Conference on Winter Simulation, 2002, pp. 243–249.
[37] D. Meisner, W. Junjie, T.F. Wenisch, Bighouse: a simulation infrastructure for data center systems, in: IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2012, 2012, pp. 35–45.
[38] The cloud computing and distributed systems (CLOUDS) laboratory, university of melbourne cloudsim: a framework for modeling and simulation of cloud computing infrastructures and services, 2016. [Online]. Available from: http://www.cloudbus.org/cloudsim/.
[39] A. Zhou, S. Wang, Q. Sun, H. Zou, F. Yang, FTCloudsim: a simulation tool for cloud service reliability enhancement mechanisms, in: Demo & Poster Track of ACM/IFIP/USENIX International Middleware Conference, Dec. 2013, pp. 1–2.

[40] C. Fiandrino, D. Kliazovich, P. Bouvry, A.Y. Zomaya, Performance and energy efficiency metrics for communication systems of cloud computing data centers", IEEE Trans. Cloud Comput. PP (Apr. (99)) (2015). 1–1.
[41] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, A.Y. Zomaya, Energy-efficient data replication in cloud computing datacenters, Springer Cluster Comput. 18 (1) (2015) 385–402.
[42] M. Guzek, D. Kliazovich, P. Bouvry, HEROS: energy-efficient load balancing for heterogeneous data centers, in: IEEE International Conference on Cloud Computing (CLOUD), Jun. 2015.
[43] D. Kliazovich, S.T. Arzo, F. Granelli, P. Bouvry, S.U. Khan, e-STAB: energy-efficient scheduling for cloud computing applications with traffic load balancing, in: IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing (GreenCom), 2013, pp. 7–13.
[44] D. Kliazovich, P. Bouvry, S.U. Khan, DENS: Data center energy-efficient network-aware scheduling, Cluster Computing, special issue on Green Networks 16 (1) (2013) 65–75.
[45] D. Kliazovich, J.E. Pecero, A. Tchernykh, P. Bouvry, S.U. Khan, A.Y. Zomaya, CA-DAG: modeling communication-aware applications for scheduling in cloud computing, in: 2013 IEEE Sixth International Conference on Cloud Computing (CLOUD), 2013c, pp. 277–284.
[46] D. Kliazovich, S.T. Arzo, F. Granelli, P. Bouvry, S.U. Khan, Accounting for load variation in energy-efficient data centers, in: IEEE International Conference on Communications (ICC), 2013d.
[47] M. Abu Sharkh, A. Shami, P. Ohlen, A. Ouda, Simulating high availability scenarios in cloud data centers: a closer look, in: 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom), Nov. 2015, pp. 617–622.
[48] J. Zhihua, Greencloud for simulating QoS based NaaS in cloud computing, in: 2013 9th International Conference on Computational Intelligence and Security (CIS), 2013, pp. 766–770.
[49] X. Li, L. Nie, S. Chen, Approximate dynamic programming based data center resource dynamic scheduling for energy optimization, in: 2014 IEEE International Conference on Internet of Things (iThings), and Green Computing and Communications (GreenCom), Sept. 2014, pp. 494–501.
[50] I.D. Faria, M.A.R. Dantas, M.A.M. Capretz, W.A. Higashino, Energy-aware resource selection model for opportunistic grids, in: 2014 IEEE 23rd International WETICE Conference (WETICE), Jun. 2014, pp. 167–172.
[51] A. Nunez, et al., Design of a flexible and scalable hypervisor module for simulating cloud computing environments, in: 2011 International Symposium on Performance Evaluation of Computer & Telecommunication Systems (SPECTS), 2011, pp. 265–270.
[52] C. Esposito, M. Ficco, F. Palmieri, A. Castiglione, Smart cloud storage service selection based on fuzzy logic, theory of evidence and game theory, IEEE Trans. Comput. PP (99) (2015) 1–14.
[53] T. Hirofuchi, A. Lebre, L. Pouilloux, Adding a live migration model into simgrid: one more step toward the simulation of infrastructure-as-a-service concerns, in: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), 2013, pp. 96–103.
[54] G. Castanea, A. Nunez, P. Llopisa, J. Carreteroa, E-mc$^2$- a formal framework for energy modelling in cloud computing, Simul. Model. Pract. Theory, S. I.Energy efficiency in grids and clouds 39 (Dec. (PP)) (2013) 56–75.
[55] I. Moreno, P. Garraghan, P. Townend, J. Xu, An approach for characterizing workloads in Google cloud to derive realistic resource utilization models, in: 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering, 2013, pp. 49–60.
[56] P. Samimia, Y. Teimourib, M. Mukhtar, A combinatorical double auction resource allocation model in cloud computing, Inf. Sci. (2014). In press.
[57] I.S. Moreno, P. Garraghan, P. Townend, J. Xu, Analysis, modeling and simulation of workload patterns in a large-scale utility cloud", IEEE Trans. Cloud Comput. 2 (Apr.) (2014) 208–221.
[58] D. Limbani, B. Oza, A proposed service broker strategy in cloudanalyst for cost effective data center selection, Int. J. Eng. Res. Appl. (IJERA) 2 (Feb. (1)) (2012) 793–797.
[59] A. Ahmed, Y. Singh, Analytic study of load balancing techniques using tool cloud analyst, Int. J. Eng. Res. Appl. (IJERA) 2 (Mar. (2)) (2012) 1027–1030.
[60] R.K. Mishra, S.N. Bhukya, Service broker algorithm for cloud analyst, Int. J. Comput. Sci. Inf. Technol. 5 (1) (2014) 3957–3962.
[61] B. Mondal, K. Dasgupta, P. Dutta, Load balancing in cloud computing using stochastic hill climbing-a soft computing approach, in: 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on Feb. 2012, Procedia Technology, 4, 2012, pp. 783–789.
[62] S. Ostermann, K. Plankensteiner, D. Bodner, G. Kraler, R. Prodan, Integration of an event-based simulation framework into a scientific workflow execution environment for grids and clouds, in: Towards a Service-Based Internet, Springer, Berlin Heidelberg, 2011, pp. 1–13.
[63] G. Kecskemeti, S. Ostermann, R. Prodan, Fostering energy-awareness in simulations behind scientific workflow management systems, in: The 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, IEEE Computer Society, 2014, pp. 29–38.
[64] H. Fard, R. Prodan, J. Barrionuevo, T. Fahringer, A multi-objective approach for workflow scheduling in heterogeneous environments, in: The 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID 2012), 2012, pp. 300–309.
[65] K. Plankensteiner, R. Prodan, Meeting soft deadlines in scientific workflows using resubmission impact, IEEE Trans. Parallel Distrib. Syst. 23 (May (5)) (2012) 890–901.
[66] S. Frey, F. Fittkau, W. Hasselbring, Optimizing the deployment of software in the cloud, in: The Conference on Software Engineering & Management 2015, Kollen Druck+Verlag, Mar. 2015.
[67] S. Frey, F. Fittkau, W. Hasselbring, Search-based genetic optimization for deployment and reconfiguration of software in the clouds, in: 2013 35th International Conference on Software Engineering (ICSE), 2013, pp. 512–521.
[68] S. Lim, et al., A dynamic energy management in multi-tier data centers, in: 2011 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2011, pp. 257–266.
[69] A. Almaatouq, A. Alabdulkareem, M. Nou, M. Alsaleh, A. Alarifi, A malicious activity detection system utilizing predictive modeling in complex environments, in: 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC), 2014, pp. 371–379.
[70] N. Thanh, P. Nam, T. Truong, N. Hung, L. Doanh, Enabling experiments for energy-efficient datacenter networks on openflow-based platform, in: 2012 Fourth International Conference on Communications and Electronics (ICCE), 2012, pp. 239–244.
[71] I. Sriram, D. Cliff, Effects of component-subscription network topology on large-scale data centre performance scaling, in: 2010 15th IEEE International Conference on Engineering of Complex Computer Systems, 2010, pp. 72–81.
[72] I. Sriram, D. Cliff, SPECI-2: an open-source framework for predictive simulation of cloud-scale data-centres", in: Simulation and Modeling Methodologies, Technologies and Applications Conference (SIMULTECH11), 2011, pp. 418–421.
[73] I. Sriram, D. Cliff, Hybrid complex network topologies are preferred for component-subscription in large-scale data-centres, in: Complex Networks, in: of the series Communications in Computer and Information Science, vol. 116, 2011, pp. 130–137.
[74] A. Banerjee, J. Banerjee, G. Varsamopoulos, Z. Abbasi, S.K.S. Gupta, Hybrid simulator for cyber-physical energy systems", in: Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), May 2013, pp. 1–6.
[75] C. Hsu, et al., Adrenaline: pinpointing and reining in tail queries with quick voltage boosting, in: 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), 2015, pp. 271–282.
[76] Z. Zhang, H. Yang, Z. Luan, D. Qian, Request squeezer: mitigating tail latency through pruned request replication, in: 2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC), 2015, pp. 357–362.
[77] P. Tandon, M.J. Cafarella, T.F. Wenisch, Minimizing remote accesses in mapreduce clusters, in: 2013 IEEE 27th International Symposium on Parallel & Distributed Processing Workshops and PhD Forum, 2013, pp. 1928–1936.
[78] M. Quwaider, Y. Jararweh, Cloudlet-based efficient data collection in wireless body area networks, Simul. Model. Pract. Theory 50 (Jan.) (2015) 57–71.
[79] Q. Althebyan, O. ALQudah, Y. Jararweh, Q. Yaseen, Multi-threading based map reduce tasks scheduling, in: 2014 5th International Conference on Information and Communication Systems (ICICS), 2014, pp. 1–6.
[80] Y. Jararweh, et al., Cloudexp: a comprehensive cloud computing experimental framework, Simul. Model. Pract. Theory 49 (Dec.) (2014) 180–192.
[81] D. Milojicic, H.P. Labs, High performance computing (HPC) in the cloud, Nov. 2015. [Online]. Available from: http://www.computer.org/web/computingnow/archive/september2012#sthash.n5vLzk7W.dpuf.
[82] K. Townsend, Containers: the pros and the cons of these VM alternatives, Dec. 2015. [Online]. Available from: http://www.techrepublic.com/article/containers-the-pros-and-the-cons-of-these-vm-alternatives/.
[83] S.J. Vaughan-Nichols, Containers vs. virtual machines: how to tell which is the right choice for your enterprise, Dec. 2015. [Online]. Available from: http://www.itworld.com/article/2915530/virtualization/containers-vs-virtual-machines-how-to-tell-which-is-the-right-choice-for-your-enterprise.html.
[84] IBM, Docker: build, ship, run, an open platform for distributed applications for developers and sysadmins, Dec. 2015. [Online]. Available from: https://www.docker.com/.
[85] S. McCanne, S. Floyd, The network simulator (ns-2), Dec. 2015. [Online]. Available from: http://www.isi.edu/nsnam/ns/.
[86] R. Malhotra, P. Jain, Study and comparison of various cloud computing simulators available in the cloud, Int. J. Adv. Res. Comput. Sci. Softw. Eng. 3 (Sept. (9)) (2013) 347–350.
[87] M. Jammal, T. Singh, A. Shami, R. Asal, Y. Li, Software defined networking: State of the art and research challenges, Elsevier Computer Networks 72 (2014) 74–98.
[88] H. Hawilo, A. Shami, M. Mirahmadi, R. Asal, NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC), IEEE Network 28 (Nov. 6) (2014) 18–26.
[89] M. M. Abu Sharkh, A. Ouda, A. Shami, A Resource Scheduling Model for Cloud Computing Data Centers, IWCMC2013 (2013) 213–218.
[90] M. Jammal, A. Kanso, A. Shami, CHASE: Component High Availability-Aware Scheduler in Cloud Computing Environment, IEEE 8th International Conference on Cloud Computing (2015) 417–484.
[91] M. Kalil, K.A. Meerja, A. Refaey, A. Shami, Virtual Mobile Networks in Clouds, Advances in Mobile Cloud Computing Systems (2015) 165.



**Mohamed Abu Sharkh** received his B.Sc. degree in computer science from the Faculty of Science, Kuwait University, in 2005, and his M.Sc. degree in computer engineering from the Faculty of Engineering and Petroleum, Kuwait University, in 2009. He has five years of professional experience as a software engineer and business analyst, then as an ERP consultant. Since January 2012, he has been with Western University, London, Ontario, Canada, where he is currently a Ph.D. candidate in the Department of Electrical and Computer Engineering. He serves as the chair of IEEE Young Professionals affinity group in London. His current research interests include cloud computing data center management, high availability in the cloud, software engineering and natural language processing.

**Ali Kanso** received his his M.Sc. and Ph.D. degrees in computer software engineering from Concordia University, Montreal , Canada in 2008 and 2012 respectively. In October 2012, he joined Ericsson. As an experienced researcher, he specializes in building highly available systems in cloud infrastructure.

**Abdallah Shami** received his B.E. degree in electrical and computer engineering from the Lebanese University in 1997 and his Ph.D. degree in electrical engineering from the Graduate School and University Center, City University of New York, in September 2002. In September 2002, he joined the Department of Electrical Engineering at Lakehead University, Thunder Bay, Ontario, Canada as an assistant professor. Since July 2004, he has been with Western University, where he is currently a professor in the Department of Electrical and Computer Engineering. His current research interests are in the areas of network optimization, cloud computing, and wireless networks.

**Peter Ohlen** is a research project manager at Ericsson, Stockholm, Sweden with more than 15 years of experience in the telecommunication field.