# Chapter 6
# CloudReports: An Extensible Simulation Tool for Energy-Aware Cloud Computing Environments

**Thiago Teixeira Sá, Rodrigo N. Calheiros and Danielo G. Gomes**

**Abstract** The cloud computing paradigm integrates several technological models to provide services to a large number of clients distributed around the world. It involves the management of large data centers that represent very complex scenarios and demand sophisticated techniques for optimization of resource utilization and power consumption. Since the utilization of real testbeds to validate such optimization techniques requires large investments, simulation tools often represent the most viable way to conduct experimentation in this field. This chapter presents CloudReports, an extensible simulation tool for energy-aware cloud computing environments to enable researchers to model multiple complex simulation scenarios through an easy-to-use graphical user interface. It provides report generation features and a simple API (Application Programming Interface) that makes possible the development of extensions that are added to the system as plugins. CloudReports is an open-source project composed of five mandatory modules and an optional extensions module. This chapter describes all these modules, their integration with the CloudSim toolkit, and a case study that demonstrates an evaluation of power consumption of data centers with a power model that is created as a CloudReports extension.

**Keywords** Cloud computing · Simulation tools · Energy-aware distributed systems · Energy-aware cloud computing · Infrastructure virtualization · Data center · Infrastructure management

D. G. Gomes (✉) · T. Teixeira Sá
Group of Computer Networks, Software Engineering and Systems (GREat),
Universidade Federal do Ceará, Av. Mister Hull, s/n, Campus do Pici,
bloco 942-A, Fortaleza—CE 60455–760, Brazil
e-mail: danielo@ufc.br

T. Teixeira Sá
e-mail: thiagosa@great.ufc.br

R. N. Calheiros
Department of Computing and Information Systems, The University of Melbourne,
Parkville, VIC 3010, Australia
e-mail: rnc@unimelb.edu.au



Z. Mahmood (ed.), *Cloud Computing,* Computer Communications and Networks,
DOI 10.1007/978-3-319-10530-7_6

## 6.1 Introduction

The cloud computing paradigm proposes the integration of different technological models to provide hardware infrastructure, development platforms, and applications as services available worldwide. It involves complex scenarios composed of multiple large data centers that provide services to clients located around the world and with different sets of requirements. The management of such complex environments demand new system architectures, protocols, and policies in order to enable optimization of resources utilization and power consumption. Since the utilization of real testbeds to validate experiments on this field requires large investments and makes replication and control of experiments harder, simulation alternatives have been broadly used. However, simulation tools either generate a large amount of data as output or force researchers to develop their own techniques to collect data, which demands an extra effort to organize and extract useful results. Thus, a tool that combines the flexibility and extensibility of simulation frameworks with functionalities that facilitate modeling and data collection would represent a significant contribution to the cloud computing research field.

Aiming to provide this contribution, CloudReports has been developed as an extensible simulation tool for energy-aware cloud computing environments. CloudReports uses the CloudSim toolkit [1] as its simulation engine and enables researchers to model multiple complex simulation environments through an easy-to-use graphical user interface. CloudReports also provides report generation features which automatically organizes simulation results and presents them with a high level of details. Additionally, it provides an API that enables the creation of extensions that are loaded as plugins using the Java Reflection API. CloudReports is an open-source project designed with multiple modules. This chapter describes all these modules and how they are integrated with CloudSim. Moreover, it presents a case study that demonstrates an evaluation of power consumption of two data centers with different power models, one of which is created as a CloudReports extension.

The rest of the chapter is organized as follows. Section 6.2 reports the state-of-the-art simulation tools aimed at distributed systems and energy-aware cloud computing environments. Sections 6.3 and 6.4 provide an overview of the CloudSim toolkit; describe the proposed CloudReports thoroughly and suggest how simulation environments are modeled to depict the software architecture with all its modules. Section 6.5 presents a case study that uses reports generation and data exporting features and demonstrates its application on the evaluation of power consumption aspects of cloud data centers. Finally, Section 6.6 presents conclusions and future work opportunities.

## 6.2 Related Works

A fair amount of simulation tools aimed at distributed systems and grid computing can be found, but alternatives for simulating energy-aware cloud computing environments are still very scarce. For example, the SimGrid framework [2] provides

means to simulate parallel and distributed large-scale systems, and the GridSim toolkit [3] offers a flexible way to model distributed environments, applications, resources, and scheduling algorithms. However, these tools lack the key cloud computing concept of resource virtualization, thus creating the need of extensions or entirely new simulators. A toolkit that exemplifies such extensions is presented by Sulistio et al. [4], notwithstanding it is not specifically focused on cloud computing environments.

Regarding cloud computing simulation tools, the iCanCloud platform [5] is an open-source project written in C++ that aims to model and simulate cloud computing systems. It is based on the OMNET network simulation framework and offers a POSIX-based API for modeling applications. However, it does not provide means to model or simulate any aspect related to power consumption.

The GreenCloud simulator [6] is an extension to the network simulator ns2 with additional features to analyze cloud computing environments. It offers power consumption modeling for servers and network elements such as switches and links. However, it does not support virtual machines representation and application-level aspects such as job scheduling policies.

The CloudSim framework [1], which is described in the next section, is a simulation engine that supports virtual machines representation, creation of scheduling algorithms, and power consumption modeling. CloudReports is a tool that uses CloudSim as its simulation engine and manages all the data created during experiments. Furthermore, it provides a graphical user interface for modeling and managing environments to be simulated.

Aksanli et al. [7] performed a comparative study where they analyzed data center simulation tools in order to evaluate green computing performance. The study highlights the features of eight simulators according to the types of resources that are simulated, how workloads are modeled, the queuing model that is used, the ability to simulate power models, the support to virtual machines simulation, the licensing applied to the project, and the type of information that each simulator generates as output. It also introduces a new simulator (GENSim) and evaluates its use to analyze different green energy integration methods in a data center in order to find the most energy-efficient solution. Additionally, Kocaoglu et al. [8] explore some of the key aspects of green computing and communications as it stresses the importance of simulation tools for evaluating new system architectures and protocols. Finally, opportunities and challenges that arise with the advent of energy-aware cloud computing environments simulators are discussed by Buyya et al. [9], and results obtained from the use of these tools are presented in the derived works [1, 10, 11].

## 6.3 CloudSim Toolkit

CloudSim is a toolkit for modeling and simulation of cloud computing environments. It offers abstractions representing physical hosts, data centers, virtual machines, and costumers of cloud services. Latest versions of the tool also support modeling of internal data center networks and energy consumption of different physical elements.

Abstractions provided by the toolkit support mainly simulation of IaaS-related (Infrastructure as a Service) components, but they can be extended by users to support simulation of PaaS (Platform as a Service), and SaaS (Software as a Service).

A simulation is constituted by the interaction between cloud providers (represented as data centers) and cloud users (modeled in the form of brokers, that may represent one or more users generating requests for the cloud providers). Users can query data center about its capabilities, request creation of virtual machines, and submit requests for execution of applications (named *Cloudlets* in CloudSim). The decision about how the requested virtual machines are mapped to the data center's hosts is defined by a *provisioning policy*. Similarly, decisions on how the host resources are divided among VMs (Virtual Machines) running on the host, and how resources assigned to a VM are divided among applications running on it are defined by *VM scheduling* and *Cloudlet scheduling* policies, respectively. A few default policies are part of CloudSim, and users can develop and evaluate their own policies for these purposes.

The modeling of application execution is achieved with a field *length* in the Cloudlet object that represents the amount of computing instructions required to complete the execution of the Cloudlet. CPU cores, which are other characteristic of hosts, have a processing capacity expressed in instructions per second. Notice that both the properties are generic in the sense that no specific unit for measuring the processing capacity and processing requirement is specified. Therefore, cores can either be modeled based on well known CPU benchmarks, such as the SPEC CPU, or can assume a user-defined arbitrary value to represent relative computing capacity among different processors and relative execution time among Cloudlets. When a Cloudlet is scheduled to a specific VM, an estimation of the required execution time is computed based on the amount of resources allocated to the VM, the specific scheduling policy in place, and the number of other Cloudlets executing on the same machine. Once the estimation is calculated, an internal event is generated in the data center entity and scheduled for the estimated finish time. When such an event is triggered, executions of Cloudlets are updated, and the number of instructions already computed is updated. When all the instructions of a Cloudlet are computed, the Cloudlet is considered completed. At each update round, the expected completion time of Cloudlets is also recalculated and update events are generated accordingly because the number of Cloudlets in a VM may have changed, and thus more resources might have become available for other Cloudlets, that might have reduced the expected time for completion.

Besides abstractions to model cloud-related entities, CloudSim contains a discrete-event simulation core that coordinates interactions between cloud providers and cloud users. The core receives messages from the entities, controls clock advance, and delivers messages to the destination entities respecting event delivery time stamps. In the earlier versions of CloudSim, the SimJava [12] library provided the simulation core. However, the utilization of such a library imposes restrictions on the scalability and performance of CloudSim. This is because the SimJava engine is based on threads, and in fact, three threads were generated for each entity: one for the input channel (to receive messages from other entities), the second for

the output channel (to send messages to other entities) and the third for the entity itself (to control the entity operation). As threads are scarce resources that are managed by operating systems, there is a limit on the maximum number of threads that can run in the operating system at any moment. This indirectly limits the scalability of the simulation, as the number of users and data centers are bounded by such a limit. Furthermore, utilization of threads generates inefficiencies at the operating system scheduling process because, eventually threads that have no operation to perform will receive CPU time.

To counter the above factors limiting the scalability of the simulator, CloudSim, since its version 2.0, contains a single-threaded simulation core that replaced the SimJava library. In order to keep backward compatibility with simulations written with earlier versions of CloudSim, the new core implements the same APIs than SimJava and contains equivalent objects that are accessible by user-generated code. Therefore, data centers and users still extend a *SimEntity* class whose message passage is controlled by the simulation core; and messages are *SimEvents* that contain a destination, tag, send time, and a generic payload, which is unpacked and interpreted by each entity. The new core also adds new features to the simulation engine such as the possibility of defining predicates that enable filtering events based on their characteristics, such as source, destination, and type. Filtered events can be handled in a different way by the system, if required for meeting particular demands of CloudSim users.

At the end of a simulation, information about execution time of Cloudlets, cost related to resource usage, and other user-defined information are available in objects generated during the execution. CloudSim users are responsible for writing the code for extracting such information from objects and presenting them. Nevertheless, the only type of output offered by CloudSim toolkit is printing in a command line terminal. Similarly, the only native way offered to CloudSim users to write a simulation is writing the corresponding Java code. Therefore, if richer visualization or more intuitive methods for expression of simulations are required, they have to be written by users. This motivated the design and development of CloudReports, which is detailed in the next section.

## 6.4 CloudReports Simulation Tool

CloudReports is a highly extensible simulation tool for energy-aware Cloud Computing environments. The tool uses the CloudSim toolkit as its simulation engine and provides features such as a graphic user interface, reports generation, simulation data exportation, and an API that enables researchers to develop their own policies by creating extensions. CloudReports simplifies the creation and configuration of simulation environments which can be manipulated and saved for later use. Researchers can create multiple data centers with different amount of resources and configure each of their hosts individually. Moreover, client behavior can be customized by setting the amount of virtual machines to be deployed and specifying

the applications (Cloudlets) that will run on them. The resources required by each virtual machine is also entirely customizable.

CloudReports allows simulations to be executed in batches, which means that researchers can determine how many realizations must be executed and the amount of time that will be simulated. After completion of all simulations, the tool generates a full report composed of a log of operations and several charts with detailed information related to resources usage, virtual machine allocations, Cloudlet execution, and data center energy consumption. Furthermore, additional files are created to enable output data to be exported to third-party applications such as MATLAB and Octave.

As previously mentioned, CloudSim uses scheduling policies and provisioning policies to make decisions during the simulation process. CloudReports provides an API that enables researchers to develop new policies which are loaded during execution time using the Java Reflection API. In order to develop an extension, researchers do not need to make any modifications to the CloudReports source code whatsoever. Notwithstanding, due to the modular characteristics of CloudReports architecture, new scheduling and provisioning algorithms can be created separately without loss of generality while making use of all CloudSim features. The following subsections address CloudReports simulation environments, its core entities, the extensions functionalities, how simulations are managed, the persistence layer, the reports manager, and the graphical user interface.

### *6.4.1 Simulation Environments*

CloudReports manages one or more simulation environments simultaneously. These environments reproduce the interaction between IaaS providers and cloud users. As depicted in Fig. 6.1, the provider owns a cloud with an arbitrary number of data centers, which are modeled according to their operating systems, processors architecture, hypervisors, available network bandwidth, utilization costs, and virtual machines allocation policies. Moreover, each data center is composed of a customizable number of hosts that are configured according to their available RAM, network bandwidth, storage capacity, processing power, virtual machine schedulers, and energy consumption models.

Clients are modeled through a resource utilization profile and settings regarding their virtual machines that will be deployed on hosts located at the provider's infrastructure. The resource utilization profile describes the clients' applications and a high-level policy that selects data centers to deploy virtual machines. This policy is represented by a simulation entity called broker that also defines how Cloudlets will be managed, and in which virtual machine they will be executed.

Cloudlets are modeled using characteristics such as necessary amount of processor cores, size in million instructions per second (MIPS), length of input and output files that are transferred between clients and providers, and utilization models for CPU, bandwidth, and memory. A virtual machine configuration includes its image size, number of processors, processing capacity in MIPS, amount of RAM and bandwidth, type of hypervisor, and Cloudlet scheduling policy.

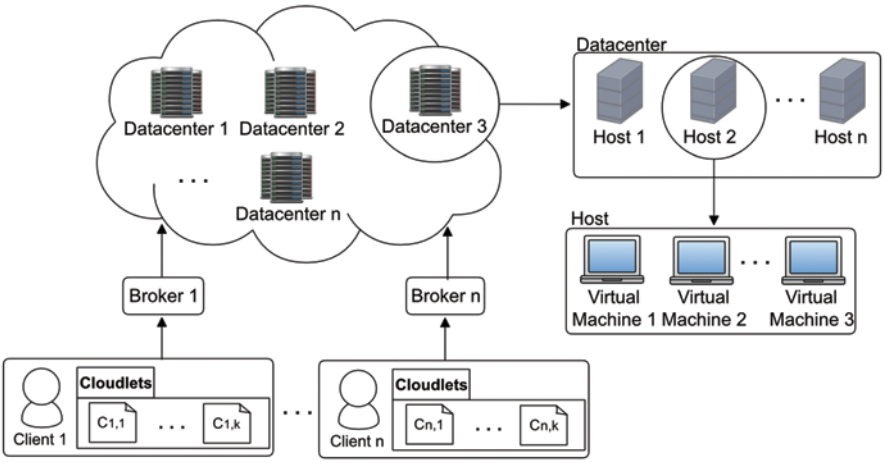


**Fig. 6.1** CloudReports' simulation environment

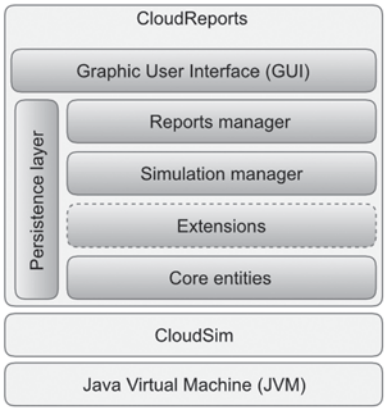


**Fig. 6.2** Modular software architecture of CloudReports

## *6.4.2 Software Architecture*

The CloudReports software architecture follows a modular design as depicted in Fig. 6.2. It currently contains five mandatory modules and an optional extensions module. The next sections describe in detail the functionalities of each of these elements and how they interact with each other.

### 6.4.2.1 CloudReports Core Entities

The core entities define the basic structure supporting CloudReports operation. They consist of classes that represent entities such as customers, data centers, physical

**Table 6.1** List of enumerations, classes, and interfaces used to develop CloudReports extensions

| CloudReports enumeration | Extensions | |
|---|---|---|
| | Must inherit from | Must implement |
| AllocationPolicy | VmAllocationPolicy | VmAllocationPolicy—Extensible |
| BrokerPolicy | Broker | – |
| BwProvisioner | BwProvisioner | – |
| RamProvisioner | RamProvisioner | – |
| PeProvisioner | PeProvisioner | – |
| VmScheduler | VmScheduler | – |
| CloudletScheduler | CloudletScheduler | – |
| UtilizationModel | – | UtilizationModel |
| PowerModel | – | PowerModel |

machines, virtual machines, networks, and storage area networks. Although being a part of CloudReports, these classes work in tandem with the Simulation Manager module to translate environments created through the graphical user interface into CloudSim entities, which are the only entities that are used during simulation time. Therefore, CloudReports works as an abstraction layer that helps users to manipulate simulation data easily, whereas CloudSim remains as the simulation engine.

Additionally, some of CloudReports core entities help in the management aspect of simulation events and settings of the software itself. These include virtual machine migrations, true random number generation, reports data, and settings such as number of simulations to perform and e-mail notifications.

In order to enable researchers to create new policies, CloudReports provides a simple API that consists of a set of enumerations, interfaces, and an extensions loader. The enumerations classify all kinds of extensions that CloudReports support, whilst the extensions loader is responsible for loading all extensions during the execution time using the Java Reflection API.

Table 6.1 lists all types of enumerations and shows which classes must be inherited as well as which interfaces must be implemented in order to develop an extension.

The AllocationPolicy extension extends VmAllocationPolicy and implements VmAllocationPolicyExtensible. It determines how data centers allocate virtual machines among servers. The BrokerPolicy extends the Broker class and describes a set of rules that clients make use to define how virtual machines are sent to allocation and how Cloudlets are sent to execution considering all the available data centers. The BwProvisioner, RamProvisioner, and PeProvisioner extensions inherit from CloudSim's namesake classes and define how servers provide bandwidth, RAM, and processing elements to the virtual machines they allocate. Moreover, the VmScheduler extension inherits from CloudSim's VmScheduler and describes how servers schedule the execution of these virtual machines. The CloudScheduler inherits from CloudSim's CloudletScheduler that determines how virtual machines schedule the Cloudlets they run. The UtilizationModel extension implements CloudSim's UtilizationModel interface and enables defining how Cloudlets

make use of the resources provided to them. Finally, the PowerModel extension implements CloudSim's PowerModel interface and makes it possible to create new CloudSim power models.

#### 6.4.2.2 Extensions

The extensions module is entirely composed of user-implemented code. Although its existence is not mandatory, it represents one of the main features of CloudReports, as it enables researchers to simulate their own algorithms without modifying CloudSim or CloudReports source code. By following a small set of rules, researchers can add new virtual machine allocation policies, data center brokers, Cloudlet schedulers, resource utilization models, virtual machine schedulers, processing elements, RAM, and bandwidth provisioners. Moreover, this module also enables the development of new power consumption models. CloudSim 3.0 already offers over a dozen types of power consumption models including options with specific hardware specifications. Researches can either extend these models or create entirely new ones as long as they follow the rules set by the extensions API provided by CloudReports.

In order to create a new extension, the researcher first needs to identify which of the aforementioned extension categories better models the algorithm that needs to be simulated. For instance, if the researcher needs to simulate a new broker policy, Table 6.1 states that it is necessary to implement a new class that inherits from CloudReport's Broker class. This new class will only contain code that is related to the new broker policy. Therefore, the researcher will be able to focus entirely on creating the new algorithm instead of having to deal with code that is related to simulator configuration. After creating this new class, a JAR (Java Archive) file needs to be created with the implementation of the new broker policy including all possible code dependencies it may have. Finally, a descriptive XML (Extensible Markup Language) file is created with all information that is necessary for CloudReports to load the new extension. Technically detailed information regarding development of extensions can be found in CloudReports project's official repository on GitHub.

#### 6.4.2.3 Simulation Manager

The simulation manager module consists of two basic elements, namely an entity factory and a simulations handler. The entity factory is responsible for turning CloudReports environments into a set of CloudSim entities, which will then be used during simulation time. The simulations handler retrieves all settings related to simulation execution and starts the simulation process. After the execution of each simulation instance, it triggers the generation of the respective report and then starts the next realization. The module is also responsible for sending e-mail notifications and handling simulation time errors.

#### 6.4.2.4 Persistence Layer

The persistence layer stores all application and simulation data in a single SQLite database file per environment. This approach facilitates the management of multiple environments as each file can be used independently and handled without execution of the application for means of backup. Moreover, since each environment makes use of a different database, it prevents tables from getting too large and keeps data access time reasonably low. However, since SQLite databases are not suitable for applications that need to process very large amounts of data, it is recommended that researchers replace the persistence layer module with more robust database solutions if they wish to perform highly scalable data-intensive simulations and keep data access time at low levels.

#### 6.4.2.5 Reports Manager

The reports manager collects, organizes, and processes simulation data from database files and generates simulation reports. The reports are composed of HTML (HyperText Markup Language) and raw data files. The HTML files contain general information about data centers and customers, which include overall and per host power consumption. The report manager uses all simulation data to generate charts automatically and include them in the HTML report files. Raw data files consist of a compilation of simulation data in a single text file that is ready to be imported by third-party applications such as MATLAB and Octave. This module acts every time the simulation manager triggers a report generation. After completing a report, it notifies the simulation manager so the next simulation realization can take place.

#### 6.4.2.6 Graphical User Interface

As the topmost module, the graphical user interface provides a simple way for researchers to manage environments and keep track of simulation progress. The GUI (Graphical User Interface) allows creation and manipulation of data centers, hosts, storage area networks, customers, virtual machines, and network links. Furthermore, researchers can set data centers' costs of operation, modify application settings, select scheduling and provisioning policies, and resource utilization models, and also determine which environments should be used during the next simulations batch. This module is written using the Swing Java GUI widget toolkit, thus, it is also platform-independent and highly customizable. Figure 6.3 shows a screenshot of CloudReports GUI.

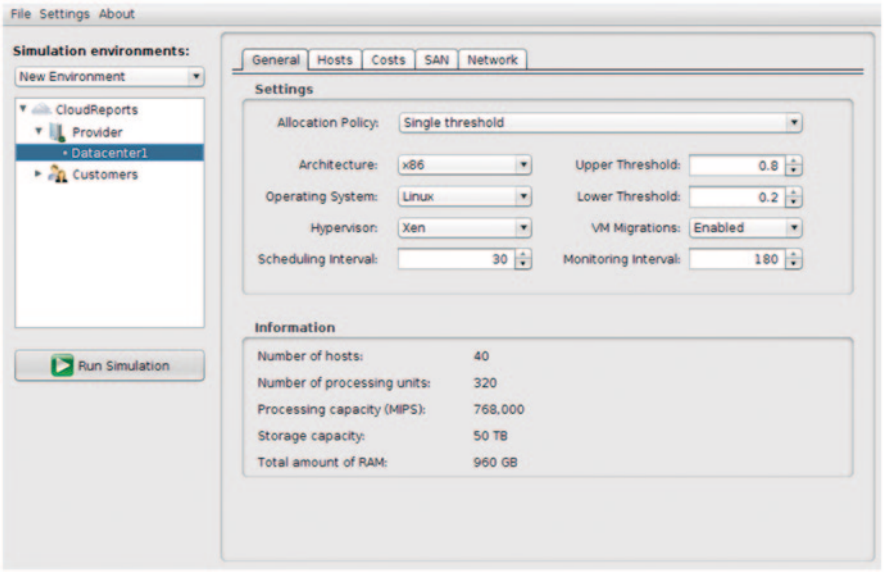


**Fig. 6.3** A screenshot of CloudReports graphical user interface

## 6.5 Case Study

The simulation environments created using CloudReports are composed of an IaaS provider and an arbitrary amount of cloud users. The IaaS provider may have one or more data centers, each of which are modeled independently with characteristics such as virtual machines allocation policies, operational costs and resource utilization thresholds. Moreover, it is possible to configure every data center's host individually. The cloud users are modeled as a set of virtual machines to be allocated by the infrastructure managed by the IaaS provider and a utilization profile. Each virtual machine can be configured using characteristics such as CPU and memory demand, type of hypervisor, and a Cloudlet scheduler. The utilization profile determines how Cloudlets are going to behave regarding resource utilization once they are executed. Furthermore, it provides a brokering policy through which it is possible to determine which data center is going to deploy a specific virtual machine.

As workload modeling plays a decisive role on the results of simulation experiments, it is necessary to use a model that is as similar as possible to real data center environments. Therefore, the experiments presented in this case study made use of data collected from the Google Cluster Data project which makes publicly available a set of resource utilization traces from a real cluster with approximately 12,000 machines managed by Google.

The workload applied to the simulated environments is modeled in CloudSim as tasks which are represented by the Cloudlet class, to be run on virtual machines that

**Table 6.2** Instance types of simulated virtual machines

| Instance type | CPU | RAM |
|---|---|---|
| Extra-small | Single 1 GHz shared core | 768 MB |
| Small | Single 1.6 GHz core | 1.75 GB |
| Medium | Two 1.6 GHz cores | 3.5 GB |
| Large | Four 1.6 GHz cores | 7 GB |

are allocated in hosts. On the other hand, the traces extracted from Google Cluster Data have information regarding the use of resources (e.g., CPU, memory, and disk) and are presented as jobs run on real machines from the monitored cluster. In order to use these traces on the simulation experiments, information from the jobs was represented as Cloudlets. Thus, it was possible to simulate environments with up to 10,000 hosts with a workload that was similar to the usage of a real data center.

Based on related works, the experiments made use of power consumption traces collected from a benchmark of real machines that is made available by the Standard Performance Evaluation Corporation. In order to use the benchmark information in the experiments, it was necessary to develop a new class that implements the CloudSim's PowerModel interface to represent a Dell PowerEdge R820 machine. Therefore, all data centers represented in the experiments of this case study are composed of a set of machines of the same model. As the benchmark data provides power consumption information in Watts based on discrete levels of load applied to a machine, creating this new power consumption model on CloudSim was straightforward as the framework already deals with power consumption based on load levels applied to the simulated hosts.

Four different virtual machine allocation policies were used in the experiments. These policies determine how the controller node should distribute virtual machines among all the available hosts. Therefore, such policies play a decisive role on the overall power consumption of the data center. The simulated policies are listed below:

- *Single Static Threshold (SST)*: this policy has a single utilization threshold that determines if a host is overloaded.
- *Double Static Threshold (DST)*: this policy has two utilization thresholds. The first determines if a host is overloaded and the second is used to identify underused hosts.
- *Median Absolute Deviation-Minimum Migration Time*: this policy has dynamic utilization thresholds and was extracted from a related work [13].
- *Local Regression-Minimum Migration Time*: this policy also has dynamic utilization thresholds and, such as the previous policy, was extracted from a related work [13].

Regarding the virtual machines configuration, the experiments used four types of profiles based on services from a real IaaS provider. Table 6.2 shows detailed information about computing capacity and available memory for each of the four profiles. All experiments made use of equal amounts of virtual machines for each of the profiles.

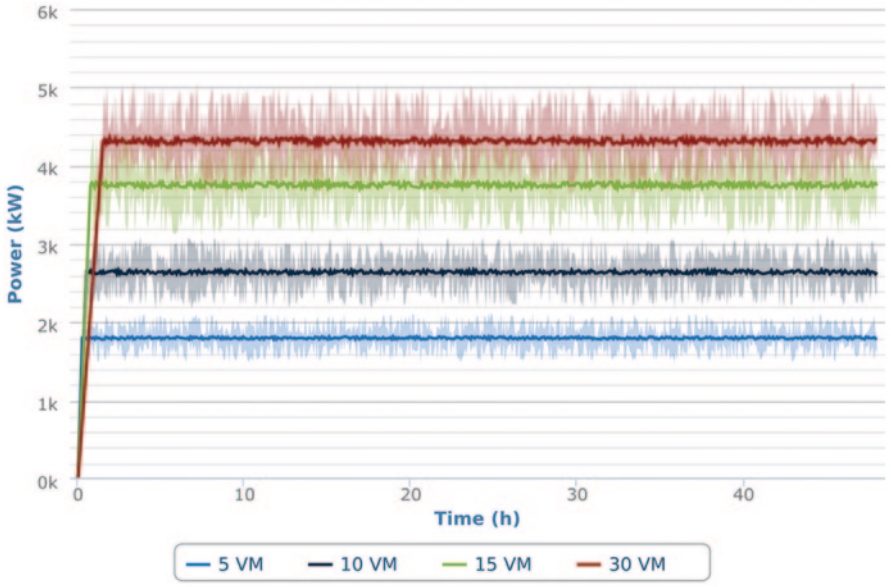


**Fig. 6.4** Power consumption of a 10,000 nodes data center with a Single Static Threshold allocation policy

Figure 6.4 shows the power consumption of a data center with 10,000 hosts during a 48 h period of operation with the Single Static Threshold allocation policy. Each line represents a different rate of virtual machines allocated per host. The shaded areas around the lines represent a 90% level confidence interval. The chart shows that power consumption increases proportionally with the amount of virtual machines allocated per host. Such behavior was expected, since, the higher the load applied to the system, the higher will be the level of resource usage, which increases the overall power consumption of the data center.

Figure 6.5 shows simulation results for an environment similar to the aforementioned but using the Double Static Threshold allocation policy. In this case, it is possible to identify a nearly linear relation between the amount of virtual machines allocated per host and the Consumption Stabilization Interval (CSI), which is defined as the period of time from the beginning of the simulation setup interval until the moment when the power consumption of the data center reaches a stable level. For the specific rate of 30 virtual machines per host, the DST policy performance is very similar to the SST policy. This happens because this rate of virtual machine allocation always keeps the data center with overloaded hosts, which undermines the DST capacity to identify underused hosts and reallocate virtual machines appropriately. This type of reallocations define what is commonly called consolidation techniques. For all other rates of virtual machines allocated per host, it is noticeable that immediately after the simulation setup time, virtual machines start to be consolidated which decreases the power consumption levels significantly.

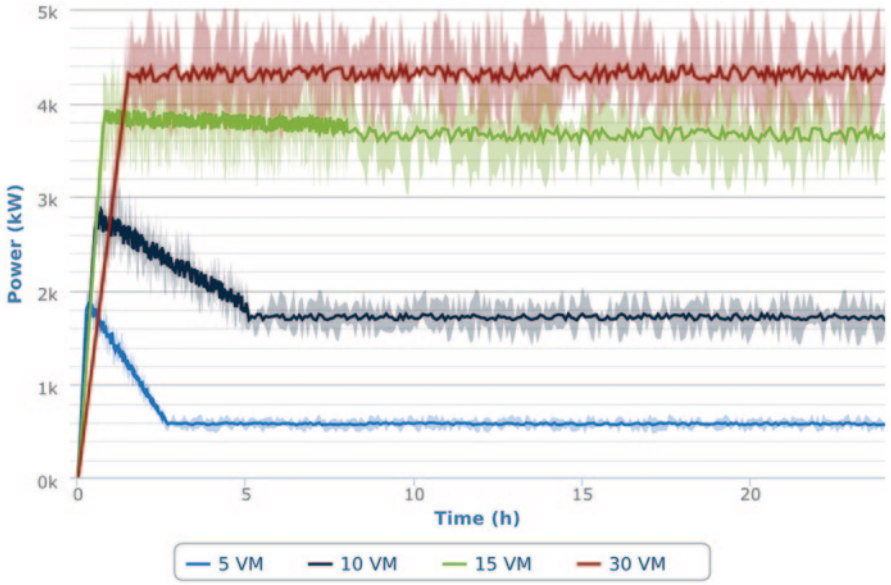


**Fig. 6.5** Power consumption of a 10,000 nodes data center with a Double Static Threshold allocation policy

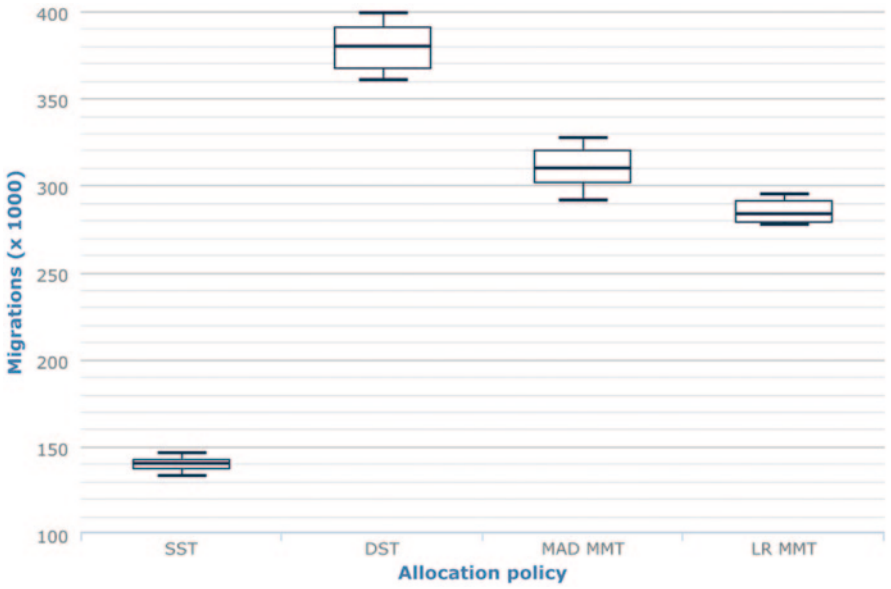


**Fig. 6.6** Number of virtual machine migrations performed on a 10,000 nodes data center

The boxplot in Fig. 6.6 shows the number of virtual machine migrations performed for each of the allocation policies. The lowest levels are shown for the SST

policy due to the lack of consolidation techniques as this policy cannot identify underused hosts. Hence, despite the low amount of migrations for the SST policy, the previous charts showed that the lack of consolidation techniques has a negative impact on the power consumption of the data center. On the other hand, all the other allocation policies present higher amounts of migrations, which result in lower levels of power consumption. It is important to notice that virtual machine migrations have a significant impact on the Quality of Service (QoS) provided to the end user. Therefore, there is a trade-off relationship between power consumption and QoS that must be considered while deciding which allocation policy should be applied in order to manage virtual machine migrations in a data center.

## 6.6 Conclusion

This chapter presented CloudReports as a tool aimed at facilitating the modeling of energy-aware cloud computing environments and data collection of simulation results from the CloudSim simulation toolkit. Related works were discussed in order to provide an overview of existing options for simulating energy-aware cloud computing environments. Moreover, some of CloudSim's key functionalities were addressed. As CloudSim represents the core simulation engine used by CloudReports, a description of how its components work and their evolution to the current version of the project was provided. Then, the architecture of CloudReports was fully described. In order to provide a clear and complete understanding of how the simulator works, the core entities were discussed, followed by descriptions on how to create new extensions and how CloudReports' modules work together. Moreover, the chapter presented a case study that used CloudReports and a power model extension to evaluate the power consumption of a data center with 10,000 machines. The case study applied different virtual machine allocation policies and showed that there is a trade-off between the QoS offered to the end user and the total power consumption of the data center. Furthermore, it also became clear that the virtual machine allocation policy applied in the data center has a great influence is this trade-off.

As the future work, we intend to add statistical analysis to the reports and integrate new CloudSim features to the graphic user interface such as intra-data center networks and the utilization of real workloads. As CloudReports is an open-source project, its source code is available online on GitHub, what enables researchers to create feature branches that can later be integrated to CloudReports' main project.

## References

1. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R (2011) Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 41:23–50
2. Casanova H, Legrand A, Quinson M (2008) SimGrid: a generic framework for large-scale distributed experiments. Proceedings of the tenth international conference on computer modeling and simulation, UKSIM'08. IEEE Computer Society, Washington, DC, pp 126–131

3. Buyya R, Murshed M (2002) Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurr Comput Pract Exp 14:1175–1220
4. Sulistio A, Cibej U, Venugopal S, Robic B, Buyya R (2008) A toolkit for modelling and simulating data grids: an extension to gridsim. Concurr Comput Pract Exp 20:1591–1609
5. Nez A, Vzquez-Poletti A, Caminero A, Casta G, Carretero J, Llorente I (2012) iCanCloud: a flexible and scalable cloud infrastructure simulator. J Grid Comput 10:185–209. doi:10.1007/s10723-012-9208-5
6. Kliazovich D, Bouvry P, Audzevich Y, Khan S (2010) Greencloud: a packet-level simulator of energy-aware cloud computing data centers. In: Global Telecommunications Conference (GLOBECOM 2010), IEEE, pp 1–5
7. Aksanli B, Venkatesh J, Rosing T (2012) Using datacenter simulation to evaluate green energy integration. Computer 45:56–64
8. Kocaoglu M, Malak D, Akan O (2012) Fundamentals of green communications and computing: modeling and simulation. Computer 45:40–46
9. Buyya R, Ranjan R, Calheiros RN (2009) Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: challenges and opportunities. Proceedings of the international conference on high performance computing & simulation (HPC & S'09), IEEE Computer Society, Leipzig, pp 1–11
10. Beloglazov A, Buyya R (2010) Energy efficient allocation of virtual machines in cloud data centers, 2010. In: 10th IEEE/ACM international conference on cluster, cloud and grid computing (CCGrid), Melbourne, pp 577–578
11. Kim KH, Beloglazov A, Buyya R (2009) Power-aware provisioning of cloud resources for real-time services. Proceedings of the 7th international workshop on middleware for grids, clouds and e-science, MGC'09, ACM, New York, 1:1–1:6
12. Howell F, Mcnab R (1998) SimJava: a discrete event simulation library for java. Proceedings of the first international conference on web-based modeling and simulation, SCS, San Diego, pp 51–56
13. Beloglazov A, Buyya R (2012) Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr Comput Pract Exp 24:1397–1420