SOFTWARE – PRACTICE AND EXPERIENCE
Softw. Pract. Exper. (2012)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/spe.2156

An environment for modeling and simulation of message-passing parallel applications for cloud computing

Saurabh Kumar Garg[1] and Rajkumar Buyya[2,*,†]

[1]*IBM Research Australia, Victoria, Australia*
[2]*Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computing and Information Systems, The University of Melbourne, Melbourne, Victoria, Australia*

SUMMARY

As Cloud computing is becoming a mainstream platform, it has also become important to understand the implications on customers' applications or systems when deployed on Clouds. Therefore, simulation tools become a critical requirement that not only evaluate the performance of Clouds but also help in developing Cloud computing further. However, current simulation solutions for Clouds do not support many important application paradigms such as message-passing parallel applications. This limits the usage of these solutions to study the deployment of many scientific applications on Clouds. In this paper, after recognizing the needs and requirements of the Cloud research and development community, we propose a Cloud simulation environment with a scalable network and message-passing application model that allows accurate evaluation of scheduling and resource provisioning policies and thus helps in optimizing the performance of a Cloud infrastructure. Copyright © 2012 John Wiley & Sons, Ltd.

Received 11 April 2012; Revised 4 July 2012; Accepted 9 August 2012

KEY WORDS: Cloud computing; simulation and modeling; message-passing applications

1. INTRODUCTION

Cloud computing [1] is becoming a mainstream computing platform for various communities including researchers, businesses, consumers, and government organizations. The reason is its tempting benefits, including elastic and scalable on-demand resources, that resulted in several explorations of this platform for efficient and cost-effective execution of different applications such as High Performance Computing (HPC), e-commerce, social networking, and Web applications.

To successfully exploit Cloud resources, applications need to be adapted to this new environment, and new scheduling solutions need to be developed for good performance. Similarly, Cloud providers need to determine proper configurations and scheduling policies for efficient usage of their computational, network, and storage resources such that different application types can be executed concurrently and in isolation. However, evaluation of alternative designs or solutions for Cloud computing on real testbeds is not easy because of several reasons. Firstly, public Clouds exhibit varying demand and supply patterns, system sizes, and resources (hardware, software, and network) [2]. Because of such unstable nature of Cloud resources, it is difficult to repeat the experiments and compare different solutions. Secondly, there are several factors that are involved in determining performance of Cloud systems or applications such as users' QoS requirements, varying workload, and complex interaction of several network and computing elements. Thirdly, real experiments on such large-scale distributed platforms are considerably time consuming and sometimes impossible

*Correspondence to: Rajkumar Buyya, Department of Computing and Information Systems, University of Melbourne, Melbourne, Victoria, Australia.
†E-mail: raj@csse.unimelb.edu.au

because of multiple test runs in different conditions. Therefore, a more viable solution is to use simulation frameworks that enable controlled experimentation, reproducible results, and comparison of different solutions in similar environments.

Despite the obvious advantages of simulation in prototyping applications and developing new scheduling algorithms for Cloud computing, there are a few simulators that cover all the requirements for modeling Cloud environments. Clouds currently deploy a wide variety of applications both from industrial enterprises and scientific community[3] such as climate modeling, drug design, and protein analysis. The various application paradigms vary from massively parallel applications to message-passing applications and complex business workflows. The modeling of these different application paradigms is essential for accurate evaluation of Cloud computing environments. Therefore, the two key user requirements of a Cloud simulator to support simulation and modeling of such a wide variety of applications are the following: (i) a message-passing application model specifying the structure of the target applications in Clouds, typically in terms of computational tasks and data communication between them; and (ii) a platform model of Cloud computing data centers specifying the nature of the available resources and the network by which they are interconnected. The modeling of networking support is essential to simulate execution behavior of message-passing applications and effects of switching delays and latency.

Currently available Cloud specific simulation solutions such as CloudSim [4] and GreenCloud [5] view data center resources as a collection of virtual machines (VM). As a result, they integrate a simple application model without any communicating tasks. These currently supported models, although perhaps appropriate for some type of applications such as single-server Web applications or parameter sweep applications, become inadequate when used to model a Cloud computing environment supporting execution of different application paradigms such as message-passing application model [3, 6]. For example, precise evaluation of scheduling algorithms for scientific applications, such as message-passing parallel applications, requires modeling of the data center's interconnection network. Because current Cloud simulators have no support for both of these features simultaneously, these limitations may result in either nonrealistic data center solutions or inaccurate solutions for applications with communicating tasks.

To overcome these limitations of current Cloud simulators, we develop a simulation toolkit, called *NetworkCloudSim*, that supports modeling of Cloud data centers that support message-application model and therefore enables modeling of applications such as HPC, e-commerce, and workflows. The main challenge addressed is the development of application and network models that are sophisticated enough to capture the relevant characteristics of Cloud data centers hosting wide variety of applications but simple enough to be amenable for analysis. In particular, we discuss issues and solutions in regard to modeling a data center's internal network and message-passing applications. Our simulation framework is developed as an extension of *CloudSim*. More precisely, our **contributions** are the following:

- We designed a new Cloud simulation toolkit, *NetworkCloudSim*, which is equipped with message-passing application model and thus supports simulation of a wide range of applications such as Message Passing Interface (MPI), e-commerce, and workflows. The components of the simulation framework are implemented as part of a widely used Cloud simulation toolkit, CloudSim, to support applications with communicating elements or tasks.
- We designed a network flow model for Cloud data centers utilizing bandwidth sharing and latencies to enable scalable and fast simulations. Most of the parameters of our simulator are configurable, allowing researchers to simulate a wide variety of network topologies.
- We presented an experimental evaluation to show the importance of network modeling in developing accurate scheduling and resource management mechanisms for different Cloud applications.

The rest of this paper is organized as follows. Section 2 presents the motivation for the development of NetworkCloudSim and a brief overview of CloudSim architecture, which also highlights the key components of NetworkCloudSim. Section 3 discusses the design issues and requirements in modeling message-passing applications within CloudSim. Section 4 details the design and implementation of NetworkCloudSim. This section also discusses tasks execution/scheduling algorithm

DOI: 10.1002/spe

and data communication within NetworkCloudSim. Section 5 presents validation results with a study on behavior of scheduling and resource allocation algorithms implemented in NetworkCloudSim. In Section 6, we discuss the state of art in simulation modeling of Cloud data centers and our contributions. Section 7 concludes the paper with some future works.

2. MOTIVATION AND CLoudSim

Traditionally, compute-intensive scientific and enterprise applications achieve parallelism at differing levels by using message-passing protocols such as MPI. Most of these applications were designed for execution on clusters, supercomputers, or Grid infrastructure. Although these infrastructures are highly efficient for such applications, they are constrained by limited availability of resources, and users need to wait for long periods to acquire access to the resources. Consider in such scenario, if an organization wants to explore Cloud computing infrastructure, it will look for resources that are instantly available for their users to execute parallel applications. The organization also requires that its parallel applications, which were designed for cluster-like environments, have the similar speedup. In addition, the organization wants to execute their applications in a cost-effective manner by acquiring the minimum amount of resources considering high variability in demand for resources. From Cloud providers' perspective, it is important that they attract such customers by supporting their needs and applications without compromising their revenue. However, it is not easy to deploy and benchmark each individual application as it is not only time consuming but also not cost-effective. In addition, generally detailed information about performance of user's application under different load condition is not available. One can say in a generic way that the performance of parallel applications are quite sensitive to network latencies and are affected by higher overheads under virtualized resources [7,8]. Although Cloud resources can be distributed using such an assumption, Cloud providers still need to know what the best resource allocation strategy is such that optimal resource usage can be achieved. Hence, Cloud providers require another methodology to analyze the strategies that can ensure the best performance of both users' applications and Cloud resources without having to deploy the services in the Cloud.

Therefore, we propose a discrete event-based simulation solution that allows Cloud providers to model a complete architecture of their data centers and simulate an accurate execution of various user applications. Using our solution, resource providers obtain answers to the following questions:

- What resource allocation strategy should be adopted to achieve optimal resource usage?
- How will execution of applications under varying load affect different resources (compute and network)? and
- Which network topology is best to serve a particular type of user request?

The answer to these questions will allow efficient deployment of applications with minimal resource wastage. In the next section, we describe the base simulator used for developing our simulation solution for parallel applications.

2.1. *CloudSim architecture*

The CloudSim is a development toolkit for discrete event simulation of Cloud computing. It has many features, which made us choose it for building our simulation environment. CloudSim supports modeling of Cloud data centers and its simulation with different hardware configurations. In addition, this simulation toolkit helps in modeling user applications having independent tasks, and design and analysis of different resource and VM provisioning and scheduling policies. Figure 1 shows components of the CloudSim architecture with the key elements of NetworkCloudSim (shown by dark boxes). The detailed design and functionality of our proposed NetworkCloudSim's components will be discussed in the later sections.

The bottommost layer of the CloudSim architecture handles the interaction between CloudSim entities and components. All components in CloudSim communicate through message-passing operations. The second layer consists of several sublayers that model the core elements of Cloud

DOI: 10.1002/spe

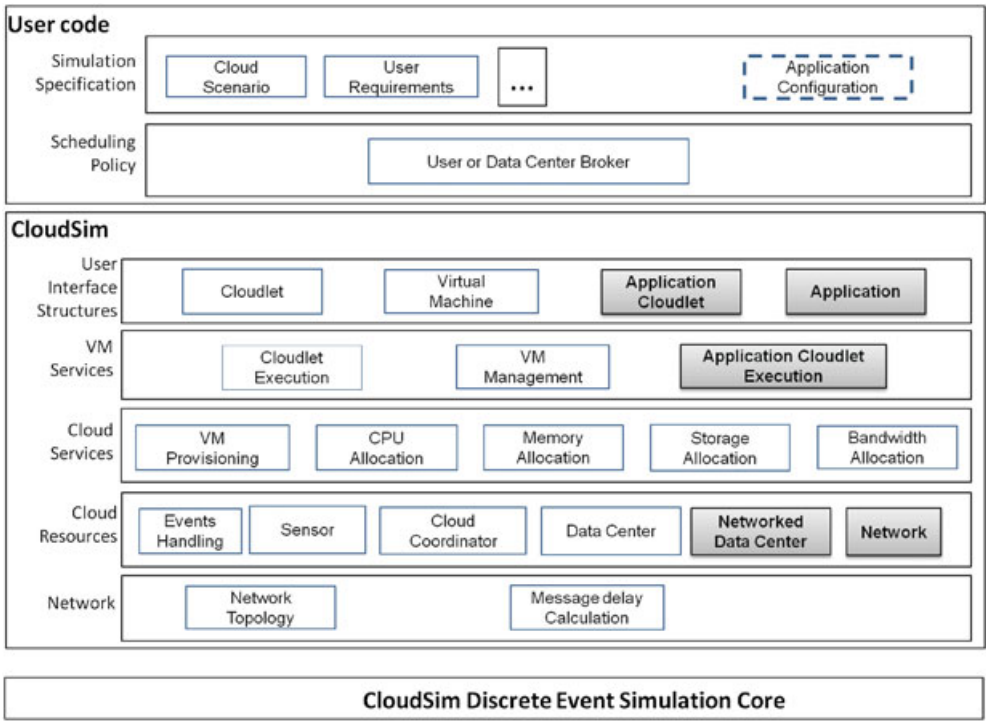


Figure 1. The CloudSim architecture with NetworkCloudSim elements. VM, virtual machine.

computing. The bottommost sublayers model data center, Cloud coordinator, and network topology between different data centers. These components help in designing Infrastructure-as-a-Service environments. The VM and Cloud services provide the functionality to design resource (VM) management and application scheduling algorithms. The layers above help users to define their own simulation scenarios and configurations for validating their algorithms. In this paper, we incorporate a generalized application model and components to design arbitrary network topologies within data centers.

3. REQUIREMENTS FOR MODELING PARALLEL APPLICATIONS

There are two main categories of issues with support for modeling and scheduling of parallel applications with CloudSim. The first set of issues are from the perspective of application configuration and modeling, and another are from Cloud data centers' perspective. In the following section, we discuss these issues individually and provide features that allow support of message-passing applications for simulation of Cloud computing environment.

3.1. Application model

The application models in Cloud computing can vary from multi-tier Web applications such as e-commerce to scientific applications [3]. Scientific applications can further vary from loosely coupled message passing to fine-grained shared memory, as well as the different possibilities for their implementation, as OpenMP, Parallel VM, or MPI.

Most simulators for Cloud computing generally offer a limited support for modeling execution of parallel and distributed applications. Although it is possible to define jobs that require more than one processor, their simulation time is just obtained by scaling them to one processor. This approach is not appropriate in more sophisticated application models such as MPI and workflow. Typically, such applications consist of several tasks, which communicate with each other. A task consists of

DOI: 10.1002/spe

computation and communication phases. This is true for both scientific applications and Web applications. A Web application consists of several tiers, with each tier running on a different server, and communication occurs between these different tiers.

CloudSim allows modeling a task by using a programming structure called 'Cloudlet', which only represents computation needs. Although one can specify input and output data associated with a Cloudlet, there is no support for specifying data communication during its execution. In this scenario, one method that could be used to represent a task of message-passing applications is via another object called 'Task', which consists of several Cloudlets. However, for large-scale experiments, this method requires initiation of several Java objects corresponding to one task and large amount of memory. In addition, the mechanism to achieve synchronization between several Cloudlets for just simulating the behavior of one message-passing task is quite complex and infeasible if we consider the current implementation of CloudSim.

Hence, in order to simulate the behavior of complex parallel and distributed applications, new structures and functionality that allow modeling of tasks that execute in different phases are required. In addition, synchronization of different tasks of the application is required during scheduling and execution. For instance, if an application consists of two tasks A and B, and both A and B simultaneously send data to each other, then the simulator should allow blocking of tasks till they receive the data.

Although the aforementioned requirements are essential to model message-passing applications in their simulation environment, still precise execution of such applications depends highly on the underlying network model. In the next section, we discuss issues and requirement in designing a network model for a data center and how it is addressed in NetworkCloudSim.

3.2. Network model

To accurately evaluate performance and scheduling of applications with communicating tasks and VM migration, it is necessary to model the network topology and bandwidth. In addition, for a Cloud provider, modeling comprehensive realistic network topologies within the data center is an important consideration because the data latency due to over-subscription of resources such as network can affect the QoS offered to Cloud customers. In fact, in many applications, this is the main factor to characterize the performance. Even though there is a network model within CloudSim, which helps in designing different topologies, it is limited to communication between different data centers and does not model bandwidth sharing on the network links. This network cannot be extended to model network within a data center. Several Cloud features, such as VM migration, create significant network overhead [9]. However, currently, CloudSim supports only models for CPU overhead. Therefore, we aim to design scalable network models to support evaluation of message-passing applications. There are mainly two issues in designing such a network:

- **Which model should be chosen?** There are two ways to design a network: packet network model or flow network model [10]. Both of these models are widely used in simulation environments for distributed systems and have their advantages and disadvantages. Among these two models, the flow network model results in low computational overhead in comparison with its counterpart but at the cost of accuracy. However, it is found that in most of the situations, the flow network model can give a good approximation to real network models such as TCP/IP [10, 11]. Because time is one of the key factors in simulation environments, the flow network model is designed within NetworkCloudSim.
- **What should be the topology of VM interconnections?** It is assumed, in the current CloudSim implementation, that each VM is connected with all other VMs. The drawback of fully connected model is that it fails to model realistic data center environments. The communication links are generally shared and interconnected through fat tree-type of network architecture [12]. To tackle this issue, we have added three levels of switches: root, aggregate, and edge level. Users can design customized type of switches and their ports according to the data center environment they want to simulate.

DOI: 10.1002/spe

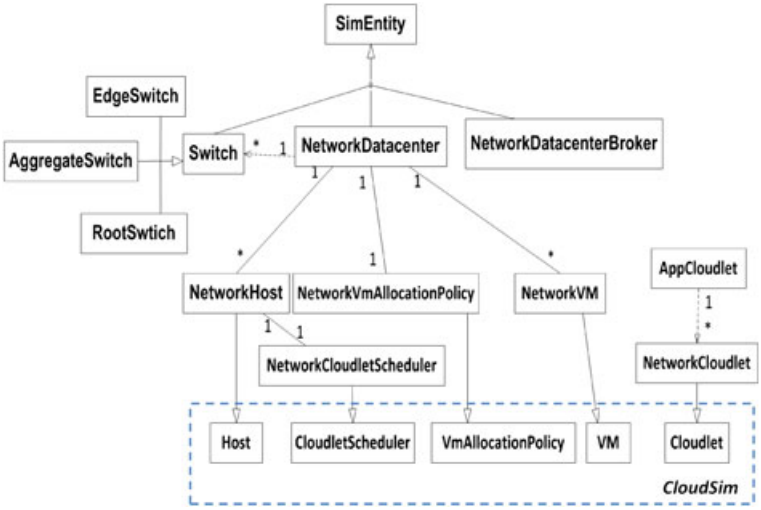


Figure 2. Class diagram of NetworkCloudSim. VM, virtual machine.

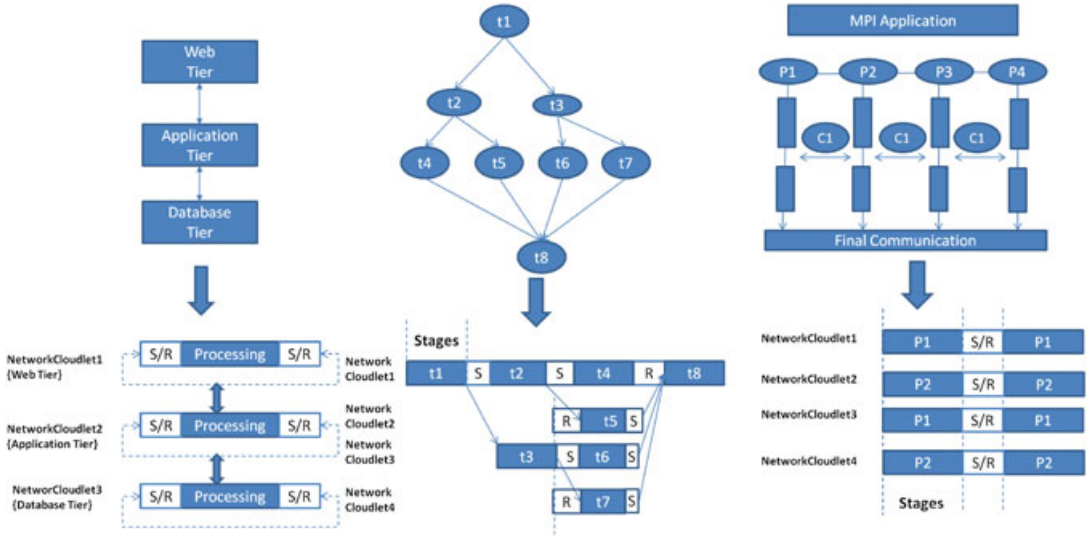


Figure 3. Modeling of applications in NetCloudSim. S stage is for sending the data, and R stage is for receiving the data. MPI, Message Passing Interface.

4. DESIGN AND IMPLEMENTATION

NetworkCloudSim extends CloudSim's [4] functionalities with the introduction of concepts that model message-passing application behavior and internal network of a data center. There are three main actors (or entities) in the NetworkCloudSim: Switch, NetworkDatacenter, and NetworkDatacenterBroker. The design of NetworkCloudSim as shown in Figure 2 contains the following main components: Switch, NetworkDatacenter, AppCloudlet, NetworkCloudlet, NetworkHost, and NetworkCloudletScheduler.

4.1. Message-passing application modeling

For helping users to model applications with communicating tasks, we designed the *NetworkCloudlet* class that represents a task executing in several phases/stages of communication and computation. Figure 3[‡] shows how we can model these applications in NetworkCloudSim. To model

[‡]S-Stage for sending the data and R-Stage for receiving the data.

DOI: 10.1002/spe

the application itself, a basic and general structure (i.e., a Java class) called *AppCloudlet* is defined. Each *AppCloudlet* object consists of several communicating elements (*NetworkCloudlets*). Each element runs in a single VM and consists of communication and computation stages. Each computation stage can be defined by either the number of Millions of Instructions Per Second (MIPS) or the seconds involved in it. The communication is characterized by the amount of transferred data.

In summary, to model generalized applications and simulate communication between different tasks, the following classes have been designed.

- **NetworkCloudlet:** The Cloudlet class has been extended to represent a generalized task with various stages (TaskStage). Each stage consists of one computation, data sending, or data reception. This class also contains information of the application to which this Cloudlet belongs. Each NetworkCloudlet represents the smallest entity executing on a VM.
- **AppCloudlet:** It represents an application with multiple tasks (NetworkCloudlet(s)).

All scheduling classes are extended to make them aware of tasks with communication and thus being able to simulate their execution.

4.2. Network flow model design

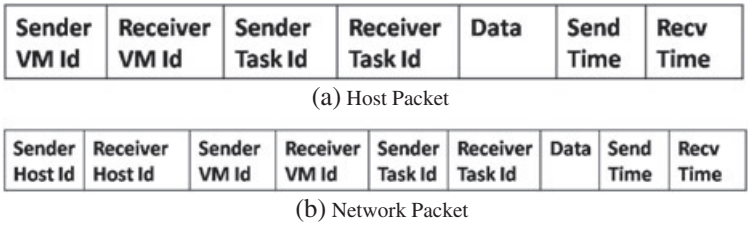
We consider a network flow model that captures the steady-state behavior of network transfers. In the system, the point-to-point communication of data from one entity (u) to another (v) is called a flow ($f = size_f, u, v$), where $size_f$ is the number of bytes in the flow. If bw is the bandwidth available between two entities and lat is the network latency, then duration of a single network flow is computed as $delay = lat + size_f/bw$. This approach significantly improves the speed of simulations in case of large network transfers. In NetworkCloudSim, we model only delays between two directly connected entities. This feature allows more accurate calculations than when flow duration is calculated over multiple links.

Because CloudSim is an event-based simulator, the event management engine of CloudSim is utilized to induce delays in the transmission of message to entities. In case of multiple simultaneous flows, we need to calculate the appropriate bandwidth available to each flow between the entities. Currently, as a proof of concept for NetworkCloudSim's network implementation, the bandwidth is equally shared by the active flows, that is, each flow obtains bw/n bandwidth if there are n flows. Some internal network models have been characterized by attributes such as the bandwidth, the latency, and the network topology. In NetworkCloudSim, users can easily add their own more complex metrics for sharing bandwidth between multiple active flows. Another advantage of this model is that it can be easily extended to packet network model. In this case, a user needs to divide the files in small chunks and send through the network. At the switch level, a user can maintain a queue of packets to add other features of packet-based networks such as bandwidth reservation and buffering.

Classes to model a network topology: To model a network within the data center, the following classes have been added to NetworkCloudSim.

- **Switch** represents a network entity that can be configured as a router or switch. It can model delays in forwarding any data to either host or another switch on the basis of where the data belong. Currently, to allow modeling various topologies, three types of switch are modeled: root, aggregate, and edge switch. The edge switch is directly connected to hosts and has uplinks connected to another switch. The aggregate switch has uplinks and downlinks to switches. The root switch is modeled as a network entity that is directly connected to the Internet/outside data center and has downlink to other switches.
- **NetworkPacket** and **HostPacket** are classes that represent a data flow from one VM to another in a data center. Because on each host the VMs are connected through a virtual network created by the hypervisor, the delay in transferring data from one VM to another hosted on the same server is negligible in comparison with when transferred through the data center's real network. To model the difference between these two networks, we designed two types of packets: HostPacket and NetworkPacket. HostPacket is the packet that travels through the virtual network. On the other hand, NetworkPacket is the packet that travels from one server to another. Each packet contains IDs of the sender VM and receiver VM, time at which it is sent

DOI: 10.1002/spe



(a) Host Packet

(b) Network Packet

Figure 4. Packet structure in NetworkCloudSim. VM, virtual machine.

and received, and type and virtual IDs of communicating tasks. The NetworkPacket contains an extra header that contains the information (IDs) of communicating hosts. Figure 4 shows the structure of these different packets.

To make CloudSim aware of the network, all the key classes are also extended. For example, NetworkHost extends native Host class, and NetworkVM extends VM.

The next section explains the scheduling mechanisms for simulating the execution of applications with communicating tasks.

Algorithm 1 Network Cloudlet Execution on VM [doExecution()]

Notations: *current_execution_queue*: queue containing currently executing NetworkCloudlet on a VM; *currstage*: current stage in which a NetworkCloudlet is; *totalNumStages*: total number of stages NetworkCloudlet execution has; *waiting_queue*: queue having NetworkCloudlet scheduled on VM but has not started execution; *NetworkDatacenterBroker*: the entity that has submitted the NetworkCloudlet for execution on VM; *packet_recv_queue* and *packet_send_queue*: queues having packets that are received and are to be sent by the VM

```
1:  for Each NetworkCloudlet cl in current_execution_queue do
2:      if cl.currstage = 'Execution' then
3:          Execute the NetworkCloudlet up to the next simulation tick
4:          send an event to notify completion of computation stage
5:          cl.currnumstage++
6:          update the current stage
7:      end if
8:      if cl.currstage = 'Send' then
9:          make a flow packet
10:         insert the packet into packet_send_queue
11:         cl.currnumstage++
12:         update the current stage
13:     end if
14:     if cl.currstage = 'Recv' then
15:         check the packet in packet_recv_queue
16:         cl.currnumstage++
17:         update the current stage
18:     end if
19:     if cl.currstage = cl.totalNumStages then
20:         cl.currstage = Finished
21:     end if
22:     if cl.currstage = 'Finished' then
23:         update the total NetworkCloudlet execution time
24:         remove the NetworkCloudlet from current_execution_queue
25:         insert another NetworkCloudlet from waiting_queue to current_execution_queue
26:         notify the NetworkDatacenterBroker about the completion of NetworkCloudlet.
27:     end if
28: end for
```


DOI: 10.1002/spe

Algorithm 2 NetworkCloudlet Scheduling on VM: *doScheduling_NonOverlap*()

```
1: for Each NetworkCloudlet cl in current_execution_queue do
2:     repeat
3:         doExecution()
4:     until cl.currstage == 'Send'
5: end for
```

Algorithm 3 NetworkCloudlet Scheduling on VM: *doScheduling_Overlap*()

```
1: for Each NetworkCloudlet cl in current_execution_queue do
2:     repeat
3:         doExecution()
4:         if cl.currstage ==' Recv' then
5:             execute the NetworkCloudlet at the top of waiting_queue
6:         end if
7:     until cl.currstage == 'Send'
8: end for
```

4.3. Scheduler and execution of NetworkCloudlet

As discussed previously, the VM scheduler needs to take into account the communication and computation stages of applications. Therefore, a new scheduler is implemented for each VM within NetworkCloudSim. NetworkCloudSim has two levels of scheduling, one at the host level (i.e., scheduling of tasks on VMs) and another at the VM level where real applications are executed. First, at the host level, NetworkDatacenterBroker maps NetworkCloudlets (each task) on different VMs on the basis of a user-defined mapping policy. Then, at the VM level, NetworkCloudlets can run in either time-shared or space-shared mode. We have currently implemented space-shared scheduling policies, which are more widely used. The scheduling mechanisms at the host level and at the VM level are implemented within NetworkVmAllocationPolicy and NetworkCloudletScheduler class as shown in Figure 2.

Algorithm 1 describes the execution process of a NetworkCloudlet on a VM. For each NetworkCloudlet, the scheduler checks the current stage of execution. NetworkCloudlet has four execution phases: *Send*, *Recv*, *Execution*, and *Finished*. If the current stage is *Execution*, the NetworkCloudlet's execution time is updated on the basis of the next scheduling interval. If the current stage is *Send*, the packet is constructed by the VM scheduler and submitted to the send-packet queue of the VMs. After the execution stage of each NetworkCloudlet is updated, the VM scheduler forwards these packets to either VMs on the same host or to the switches (to forward to the host containing the VM). If the stage is *Recv*, the scheduler checks whether there is any packet in *packet_recv_queue*. If a packet is received, the current stage of NetworkCloudlet is updated. If the stage is *Finished*, the total execution time of the NetworkCloudlet is calculated, and it is removed from the execution queue. The messages are sent to NetworkDatacenterBroker to notify about the NetworkCloudlet (task) completion. In the current implementation of VM scheduler, the nonblocked send approach is adopted, such that a sender will not be blocked even though the corresponding receiver VM is not ready in receiving the packet. On the receiver VM's side, if the packet is available, there is no communication delay for the receiver VM; if the packet is not available, the receiver has options either to process other tasks or to be blocked until the message arrives. This communication model allows the simulation of the nonblocking message-passing paradigm (such as *MPI_Isend*() and *MPI_Irecv*()), which is a common practice in parallel applications.

There are two scheduling strategies that can be configured by the user in NetworkCloudSim. Algorithms 2 and 3 present these strategies, that is, *doScheduling_NonOverlap*() and *doScheduling_Overlap*(). In *doScheduling_NonOverlap*(), execution of all tasks is blocked if there is any Cloudlet that is waiting to receive some data from other tasks. In contrast, *doScheduling_Overlap*() is designed in a way in which CPU scheduling is carried out by

DOI: 10.1002/spe

operating systems. When a task is waiting for input/output, another task in waiting queue is given a chance for execution. This will avoid any wastage of CPU cycles in the VM.

For scheduling multi-tier applications, the currently implemented algorithm requires some modifications because the nature of these applications is quite different from scientific applications. Multi-tier applications are event based. Therefore, when a packet is received (Line 21), on the basis of the packet 'type', the processing of the NetworkCloudlet and data communicated between different tiers will vary. Therefore, in Figure 3, networkCloudlets have a circular array with three stages. NetworkCloudSim allows the users to configure and implement this event-based logic according to their requirements.

4.4. Data center network

In this section, we describe a typical example of communication between different entities within NetworkCloudSim. It is presented using a sequence diagram in Figure 5. By extending this example, one can model even more complex networks. Each VM sends packets from its *send_packet_queue* to its host for further processing. As shown in Figure 4, a HostPacket contains the information about the communicating tasks (NetworkCloudlet IDs), the VMs between which the data will be transferred, the data, and the time at which the data are sent and received. The VM IDs help the host to find which VM the packet needs to be sent, whereas NetworkCloudlet IDs help VM to recognize which task the data belong to. This strategy is similar to the way packets are transferred between a local area network and a wide area network.

Therefore, using VM ID field, a NetworkHost server decides whether the packet is to be forwarded to a local VM or not. If the packet is to be forwarded to a local VM, it is inserted in the received packet list, and the field representing the received time is set. During the next scheduling cycle, the task scheduler (NetworkCloudletScheduler) of the VM will obtain the packet from the received packet queue by using the receiver's NetworkCloudlet ID and update the execution of the corresponding task.

Otherwise, this packet will be encapsulated within a network packet by adding a header as shown in Figure 4, and then the host forwards it to the switch (generally called edge switch) where NetworkHost is connected. The edge switch (Switch1) decides whether the packet belongs to a VM in its domain or to another switch. NetworkCloudSim provides a facility to users to design their own routing algorithms and configure network and switching latencies. They can add routing/switching delays by sending an event to 'switch' itself. On the basis of the decision, the packet is forwarded either to other switches or to a connected host with a delay, which is calculated on the basis of the available bandwidth and packet (data) size. The host further forwards these packets to the VM. The packet from a host to its local VM is forwarded without any communication delay.

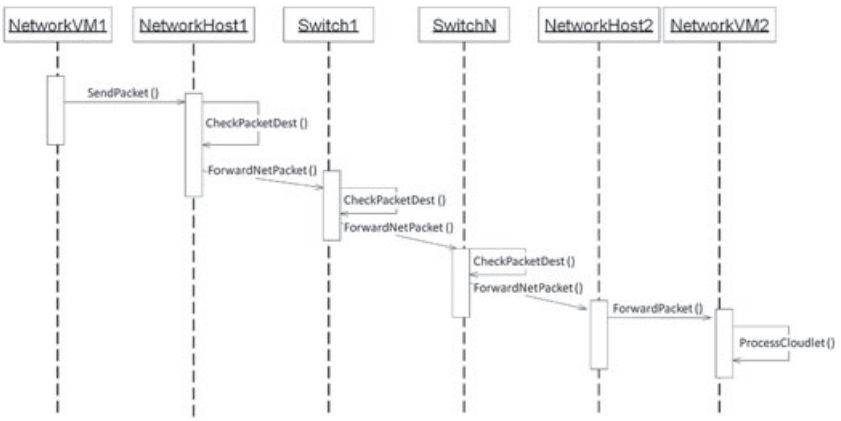


Figure 5. Sequence diagram: communication between NetworkCloudSim entities.

DOI: 10.1002/spe

5. VALIDATION AND PERFORMANCE EVALUATION

To validate and understand the behavior of NetworkCloudSim, we conducted three sets of experiments. Firstly, we compare the simulated execution time from the NetworkCloudSim with the execution of a controlled MPI program on a real infrastructure. In the second set of experiments, we present the experiments to understand the behavior of network components and the packet scheduling algorithms. In the final set of experiments, we present a case study on how NetworkCloudSim can be used to model and simulate an HPC as a Service (HaaS) Cloud environment.

5.1. Comparison of simulated and real execution times

To validate the accuracy of the simulation results of NetworkCloudSim, we compare the execution of an MPI application on a real infrastructure with simulated execution in NetworkCloudSim. For the experiments, we obtain traces of a controlled MPI program on a small-scale real infrastructure by using the *mpilog* tool using the methodology given by Miguel-Alonso *et al.* [13]. In the MPI application, the main process generates several random numbers and then sends the data to all other processes/VMs. The topology and configuration of the infrastructure are given in Figure 6. Each host has two Xen VMs, each one with two cores and 1.5-GB RAM. All the hosts are connected by a 100-Mb switch. Each process of the MPI application is executed on individual VMs. For comparison, the same configurations are used for NetworkCloudSim. For evaluation, two experimental scenarios are considered: (i) varying number of communicating processes with $1,000,000$ MPI_INT elements transferred from one process to another and (ii) varying the amount of data transferred from the main process to the other seven processes. Figure 7(a,b) shows the

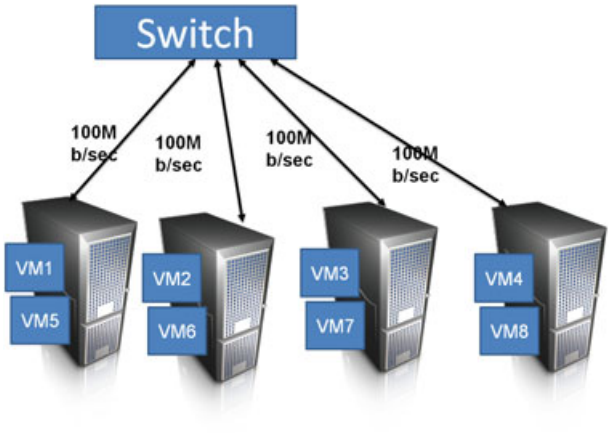
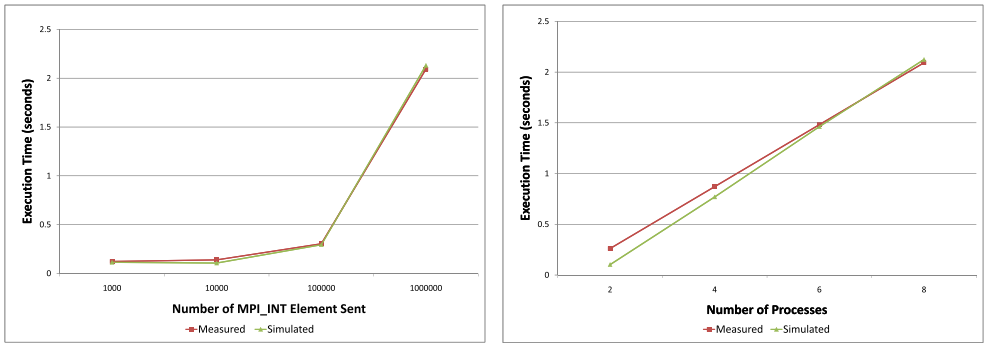


Figure 6. Experimental data center infrastructure. VM, virtual machine.



(a) Effect of Data Sent (b) Effect of Communicating Processes

Figure 7. Measured and simulated execution times of the Message Passing Interface application.

DOI: 10.1002/spe

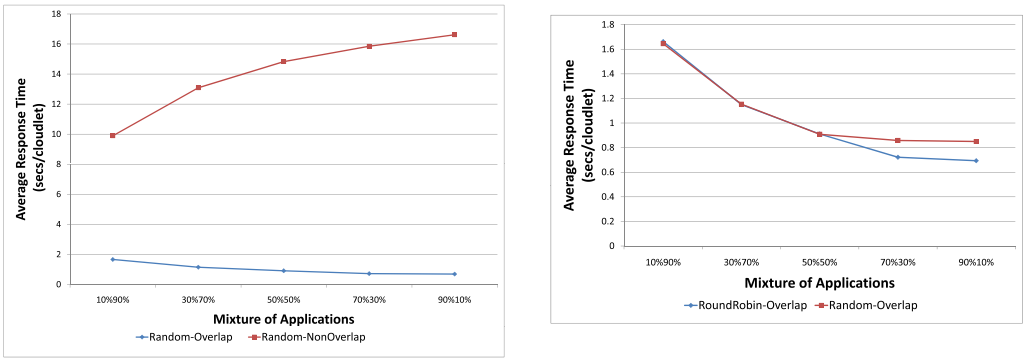
experimental results. The results from the real execution are shown as 'Measured', whereas the results from NetworkCloudSim simulation as 'Simulated'. The execution time of the application includes the computation and communication time. When we change the number of communicating processes or data transferred, the communication time increases as the bandwidth is shared. This leads to an increase in the total execution time. Figure 7(a,b) clearly shows such behavior for the measured results, which are closely followed by the simulation ones. The difference between both the results is very low, particularly when the amount of data is changed with eight communicating processes. As the number of processes increases, the simulation results match very closely to the measured ones. Therefore, we can conclude that NetworkCloudSim is applicable in the execution of parallel applications.

5.2. *Evaluation of task assignment and scheduling policies*

We have conducted experiments to further study the scheduling and resource allocation policies designed for NetworkCloudSim. For this set of experiments, we have considered a mixture of applications (parallel and parameter sweep applications) submitted to the data center. The arrival rate of applications is 200 per second. Again, a very small-scale data center with the configuration used previously is modeled. Firstly, we compare the effect of the resource allocation to each task of the application, and secondly, we compare the effect of scheduling each task on the allocated VM. For the first set of results, we compare two scheduling policies:

- **Random-NonOverlap:** In this scheduling policy, a VM executes the NetworkCloudlets in a first-come first-serve basis; therefore, other NetworkCloudlets are not executed unless the currently executing one is finished.
- **Random-Overlap:** In this scheduling policy, a VM starts executing the NetworkCloudlets in the front of a waiting queue if the currently executing task is just waiting for the data (packet) to arrive from other peer tasks.

In the aforementioned policies, *Random* denotes that allocation of a VM to a task is carried out randomly. Figure 8(a) presents the first set of results, where the X-axis represents the ratio of the two types of applications (a%b%, i.e., a% of parallel applications and b% of parameter sweep applications). We observe from the figure that the average response time (execution time + waiting time) of NetworkCloudlets for Random-Overlap is quite low in comparison with that for Random-NonOverlap. This is because of communication delays in receiving data by the VM, which causes large average response time in case of Random-NonOverlap. Another thing that we can observe from Figure 8(a) is the impact of different mixtures of applications. With the increase in ratio of applications having communicating tasks, the average response time due to the Random-NonOverlap policy is increasing, whereas it is decreasing in the case of the Random-Overlap scheduling policy. The reason for such a behavior in the case of the Random-NonOverlap scheduling



(a) Effect of Scheduling Processes (Cloudlet)

(b) Effect of Application Allocation

Figure 8. Performance of scheduling algorithm.

DOI: 10.1002/spe

policy is the increase of the communication delay with the increased proportion of applications having communicating tasks. In the case of the Random-Overlap scheduling policy, this delay increase is neutralized by the overlapping of computation of one task and the communication of another task.

In the second set of experiments, we compare the impact of the resource allocation policy to understand how scheduling of communication tasks in different locations of the data center impacts the response time. Figure 8(b) presents the results comparing the two allocation policies Random-Overlap and RoundRobin-Overlap. In the case of the RoundRobin-Overlap resource allocation and scheduling policy, tasks are assigned to VMs in a round-robin manner, whereas for the other policy, tasks are randomly assigned to VMs. It can be noticed from Figure 8(b) that initially when the proportion of communicating tasks is low, both the policies behave very closely. But as the number of communicating tasks increases, the average response time for the Random-Overlap policy becomes higher than the RoundRobin-Overlap policy. The reason for this is the shared network where different packets compete for the available bandwidth to reach their destination. Clearly, this can lead to higher communication delays and thus high response time. In summary, we can conclude from the aforementioned observations that the modeling of network is an essential part of the Cloud simulations.

5.3. Case study: High Performance Computing as a Service

Because of the popularity of Cloud computing, HPC users have started exploring Cloud as an alternate platform to run their scientific applications. Hence, many researchers are trying to provide this support in Clouds, which have been built up to now mainly for supporting Web or e-commerce applications. In this section, through a case study, we discuss how NetworkCloudSim can help such researchers to develop advanced resource management policies to support scientific applications. This case study also tests the capability of NetworkCloudSim in such scenarios. For this study, we have considered three typical scientific applications for modeling: parametric sweep application (consists of four independent tasks), MPI application (consists of four MPI tasks communicating using MPIALLTOALL), and workflow application (consists of three tasks). Figure 9 shows task dependency in workflow and MPI applications. Because there is no publicly available workload for simulating HaaS [14], we generated application workload by using uniform distribution. We consider that 10, 000 applications of different types are submitted to the data center for execution. The execution time of each task of an application is assumed as 1000 s, and the data communicated between two tasks is 10, 000 bytes.

From the cloud infrastructure perspective, we have considered a data center containing commodity hardware with hierarchical tree network topology [15]. Therefore, in our experiments, each simulated server is equivalent to an Intel Core i7-920 with four cores and eight threads (virtual cores), 2.66 GHz with 8-GB RAM. We have deployed two VMs per server, each with four cores and 4-GB RAM. By default, each server is connected to the edge network with 1-Gb Ethernet link. Servers are placed in racks, and each rack has one edge switch. We have three types of switches: root switch, aggregate switch with one uplink, and edge switches. The edge and aggregate switches are connected by 10-Gb Ethernet links. The network bandwidth between aggregate switches is 20 Gbps. The default number of servers in each rack is 10; therefore, up to 160 VMs are simulated. The switching delays for different number of hops are based on the work by Kandalla *et al.* [16], that is,

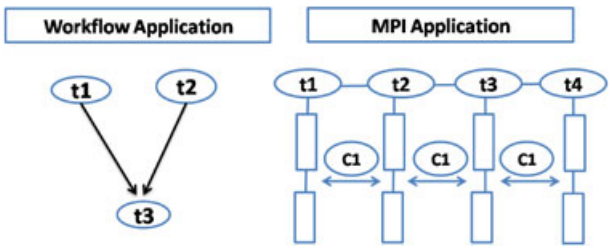


Figure 9. Workflow and Message Passing Interface (MPI) applications.

DOI: 10.1002/spe

the intra-rack communication delay is 1.57 μs, the delay due to communication through aggregate switch is 2.45 μs, and the delay due to communication through root switch is 2.85 μs.

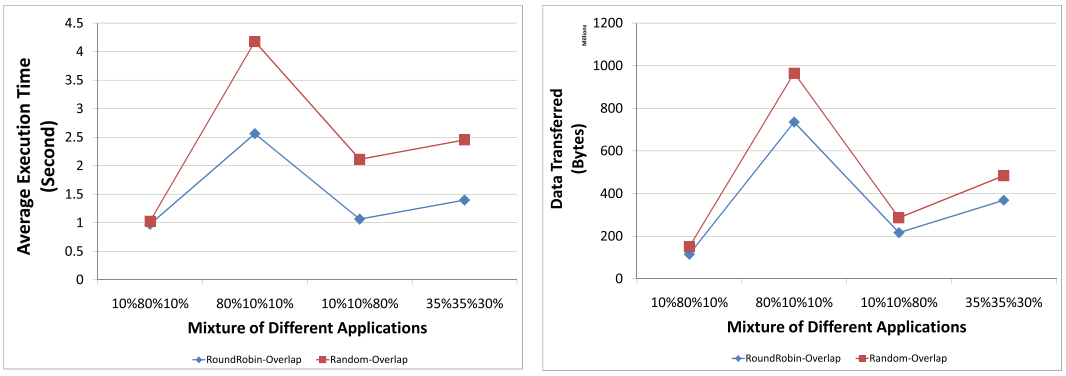
We compare performance of two application scheduling algorithms (Random-Overlap and RoundRobin-Overlap) as discussed in the previous experiment.

Performance metrics and experimental scenarios: We use two metrics to evaluate our mechanism: average processing time and network overhead. The average processing time indicates how fast our mechanism can process user requests, which is an important QoS metric for any Software-as-a-Service provider. The network overhead indicates how much data are transferred through the edge switches, and it shows the importance of the network topology and application for scheduling tasks. We examine various experimental scenarios to evaluate the performance of our scheduling mechanism, and we consider the following aspects in the experiments:

- effect of different mixture of different applications and
- effect of network topology.

Figure 10(a,b) presents how scheduling of different applications affects the performance of HaaS. In this figure, 10%80%10% represents the scenario where 10% of applications are MPI applications, 80% are parameter-sweep applications, and 10% are workflow applications. When the majority of tasks belongs to parameter-sweep applications (80%), both the scheduling mechanisms behave in a quite similar manner. In contrast, as the data communication increases when the application mixture ratio is 80%10%10%, the execution time increases drastically with much more data transfer. This experiment shows that a HaaS provider should allocate the resources on the basis of the application requirements and type for achieving high throughput and performance from the Cloud infrastructure.

Figure 11 shows how different network topologies can affect the performance of a HaaS Cloud. For this experiment, we consider application ratio as 80%10%10% because the communication



(a) (b)

Figure 10. Effect of application type.

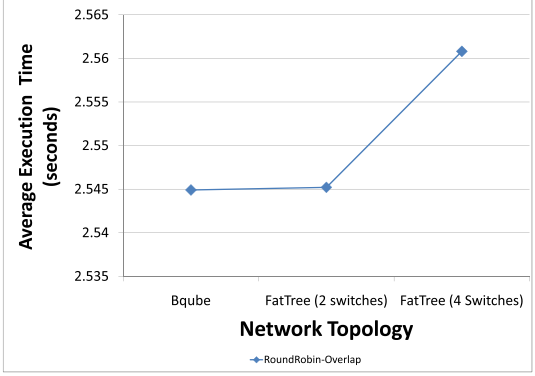


Figure 11. Effect of network topology.

DOI: 10.1002/spe

overhead is the maximum in this scenario. We consider three kinds of network topologies (BQube, FatTree with two edge switches, and FatTree with four switches) that are configured by the Cloud provider within the data center. In BQube topology, the available bandwidth between any two hosts is the same. In FatTree with two edge switches topology, we modeled two edge switches connected by one aggregate switch. In FatTree with four edge switches topology, we modeled four edge switches, two aggregate switches, and one root switch. During experiments, we found that the total data transferred through each switch remain the same; however, there is a slight difference in the average execution time of each application. As the number of switch level increases, that is, from BQube topology to FatTree with four edge switches topology, the communication delay also increases. Because of this, FatTree with four edge switches topology results in the maximum average execution time. Therefore, a HaaS provider should also consider the network topology to evaluate the performance of his or her Cloud infrastructure.

6. RELATED WORK

In this section, simulation frameworks closely related to our work are discussed. To model a network topology in general, researchers used various network-specific simulators such as NS-2 and OMNET++ [17]. These simulators can model network details very precisely but lack other features such as virtualization modeling that are essential for simulating a Cloud environment.

Other than these network-specific simulators, there are frameworks for modeling HPC platforms such as SIMCAN [18]. Using SIMCAN, one can model various hardware components such as basic systems (computing, memory managing, input/output, and networking), physical components (nodes, switches, etc.), and aggregations of components. iCanCloud [19] is a novel simulator for modeling Cloud infrastructures that is built on top of SIMCAN [18]. It offers a user friendly graphical interface for configuration of system. The current implementation of iCanCloud only supports Amazon EC2 instances; however, CloudSim is a toolkit that provides a very generic Cloud model, which can be extended to any Cloud environment. Similarly, there are simulators for modeling specific applications on Cloud computing platforms. For example, Liu *et al.* [20] presented HSim to simulate MapReduce applications in a Hadoop [21] cluster environment.

In the area of distributed computing, for many years, Grid computing vastly interested scientific community because of its advantages in delivering high-performance services for compute-intensive and data-intensive scientific applications. To support the research and development of Grid middleware, several simulators such as SimGrid [22] and GridSim [23] have been proposed. Although these environments are capable of effectively and comprehensively modeling Grid environment and applications, none of them provide a clear abstraction of application, virtual and physical machines required by Cloud computing environment. These abstractions are essential to model the multi-layer services (Software as a Service, Platform as a Service, and Infrastructure as a Service) of Cloud computing. In these tools, there is almost no support for modeling virtualized resources and application management environment.

Despite the recent development and establishment of many global Cloud computing environments by organizations such as Amazon, Yahoo, and HP, still there are very few simulation environments available for Cloud computing. Major simulators designed specifically for Cloud computing are CloudSim [4], GreenCloud [5], and MDCSim [24]. The GreenCloud [5] simulator is an extension of the NS2 network simulator for evaluation of energy-aware Cloud computing data centers. The main strength of the GreenCloud simulator is the detailed modeling of communication aspects of the data center network. Being built on top of NS2, it implements a full TCP/IP protocol reference model, which allows integration of different communication protocols such as IP, TCP, and UDP with the simulation. However, this is also a disadvantage because it limits its scalability to only small data centers because of large simulation time and high memory requirements. MDCSim [24] is a commercial discrete event simulator developed at the Pennsylvania State University. It helps user to model specific hardware characteristics of different components of a data center such as servers, communication links, and switches, which are collected from different vendors. CloudSim [4] is the most advanced among the three simulation environments. CloudSim [4] scales well and has a low simulation overhead. Its network package maintains a data center topology in the

DOI: 10.1002/spe

form of a directed graph. However, no network topology is designed for modeling an internal data center network.

All the aforementioned Cloud simulators implement user application models as simple objects describing computational requirements for the application. However, there is no support for more realistic and complex applications with communicating tasks such as parallel and data-driven applications and workflows. Currently supported workloads are more relevant to Grid networks rather than Clouds. GreenCloud [5] and CloudSim [4] specify communication requirements of the applications in terms of the amount of data to be transferred before and after a task completion. MDCSim only supports application with computation tasks.

The precision of a simulator and the validity of the results are highly dependent on the degree of details that its simulated components capture for imitating the behavior of their physical analogs. Therefore, it is essential to incorporate key elements such as a generic application model and a data center network model in the Cloud simulation tools. This work aims at developing such a tool to simulate the scheduling of complex applications such as MPI on Cloud data centers. We built our simulator on top of CloudSim toolkit, leveraging the features of the original framework and integrating various application models and flow-level networking models. However, the principles outlined in this work could be applied to other simulation platforms as well.

7. CONCLUSIONS

Use of simulation frameworks such as CloudSim is becoming increasingly popular in the Cloud computing community because of their support for flexible, scalable, efficient, and repeatable evaluation of provisioning policies for different applications. These frameworks allow fast evaluation of scheduling and resource allocation mechanisms within Cloud data centers, which are sometimes not easy to access. Thus, considering the needs of today's Cloud researchers, we presented a simulation framework that supports the modeling of essential data center resources such as network and computational resources, and a wide variety of application models such as parallel application, workflow, and parametric sweep. Our simulation framework is built on top of a widely used simulation toolkit, that is, CloudSim.

We presented the main components of the NetworkCloudSim with their functionality and how different network topologies and parallel applications can be modeled. The evaluation results show that NetworkCloudSim is capable of simulating Cloud data center network and applications with communicating tasks such as MPI with a high degree of accuracy. The further evaluation of task assignment and scheduling policies shows how NetworkCloudSim can help in building advance scheduling and resource allocation mechanisms for Clouds. Through a case study of HaaS, we also showed that by observing the impact of shared network on the performance of data centers, researchers can optimize the data center usage. This can, in turn, help in the rapid development of more power-efficient resource management schemes before committing time and resources in building complex software and network systems that operate within Cloud data centers.

Even though flow network model is sufficient for most network calculations, still it is not very accurate when compared with packet level model, particularly with the number of hops that a packet encounters. In future, we will integrate packet level network model in NetworkCloudSim so that users can simulate those Cloud applications that require precise network configurations. In the current version of NetworkCloudSim, we have implemented a simple packet routing scheme. For modeling and simulating different network topologies other than FatTree, we plan to integrate more complex routing schemes for topologies where one edge switch is connected to more than one aggregate switch. We also plan to implement the support for modeling multi-tier Web applications, which could be implemented by extending the current NetworkCloudlet implementation.

ACKNOWLEDGEMENTS

The authors would like to thank Anton Beloglazov, Rodrigo Calheiros, Ruppa Thulasiram, and Parimala Thulasiraman for their guidance during the development of NetworkCloudSim reported in this paper,

DOI: 10.1002/spe

which is a substantially extended version of our earlier conference paper [25]. This work is carried out as part of a project funded by the Australian Research Council (ARC).

REFERENCES

1. Buyya R, Yeo C, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems* 2009; **25**(6):599–616.
2. Napper J, Bientinesi P. Can cloud computing reach the top500? In *Proceedings of the Combined Workshops on Unconventional High Performance Computing Workshop plus Memory Access Workshop*, Ischia, Italy, 2009; 17–20.
3. Varia J. *Cloud Computing: Principles and Paradigms*. Wiley Press: New York, USA, 2011. ch. 18: Best Practices in Architecting Cloud Applications in the AWS Cloud, pp. 459–490.
4. Calheiros R, Ranjan R, Beloglazov A, De Rose C, Buyya R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 2011; **41**(1):23–50.
5. Kliazovich D, Bouvry P, Khan S. GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing* 2010. Available from http://dx.doi.org/10.1007/s11227-010-0504-1 [last accessed 20 September 2012].
6. Ekanayake J, Fox G. High performance parallel computing with clouds and cloud technologies. In *Proceedings of the First International Conference on Cloud Computing*, Munich, Germany, 2009; 20–38.
7. Evangelinos C, Hill C. Cloud computing for parallel scientific HPC applications: feasibility of running coupled atmosphere–ocean climate models on Amazon's EC2. In *Proceeding of the First International Conference on Cloud Computing and its Applications*, Chicago, USA, 2008; 2–34.
8. Armbrust M, Fox A, Griffith R, Joseph A, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M. A view of cloud computing. *Communications of the ACM* 2010; **53**(4):50–58.
9. Liu H, Jin H, Liao X, Hu L, Yu C. Live migration of virtual machine based on full system trace and replay. In *Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing*, Munich, Germany, 2009; 101–110.
10. Broberg J, Buyya R. *Grid Computing: Infrastructure, Service, and Applications*. CRC: Boca Raton, FL, US, 2009. ch. Flow Networking in Grid Simulations.
11. Casanova H. Network modeling issues for grid application scheduling. *International Journal of Foundations of Computer Science* 2005; **16**(2):145–162.
12. Greenberg A, Hamilton J, Jain N, Kandula S, Kim C, Lahiri P, Maltz D, Patel P, Sengupta S. VL2: a scalable and flexible data center network. *ACM SIGCOMM Computer Communication Review* 2009; **39**(4):51–62.
13. Miguel-Alonso J, Navaridas J, Ridruejo F. Interconnection network simulation using traces of MPI applications. *International Journal of Parallel Programming* 2009; **37**(2):153–174.
14. Shainer G, Liu T, Layton J, Mora J. Scheduling strategies for HPC as a service (HPCaaS). In *Proceedings of 2009 IEEE International Conference on Cluster Computing and Workshops*, CA, USA, 2009; 1–6.
15. Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture. *SIGCOMM Computer Communication Review* Oct. 2008; **38**(4):63–74.
16. Kandalla K, Subramoni H, Vishnu A, Panda D. Designing topology-aware collective communication algorithms for large scale InfiniBand clusters: case studies with scatter and gather. In *10th Workshop on Communication Architecture for Clusters (CAC'10)*, Atlanta, USA, April 2010.
17. Varga A. The OMNET++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference on Simulation and Modelling: Enablers for a Better Quality of Life*, Germany, 2001.
18. Núñez A, Fernández J, Garcia J, Garcia F, Carretero J. New techniques for simulating high performance MPI applications on large storage networks. *The Journal of Supercomputing* 2010; **51**(1):40–57.
19. Nuñez A, Vázquez-Poletti J, Caminero A, Carretero J, Llorente I. Design of a new cloud computing simulation platform. In *Proceeding of 11th International Conference on Computational Science and Its Applications (ICCSA 2011)*, Santander, Spain, 2011; 582–593.
20. Liu Y, Li M, Alham N, Hammoud S. HSim: a MapReduce simulator in enabling cloud computing. *Future Generation Computer Systems* 2011. Available from http://www.sciencedirect.com/science/article/pii/S0167739X11000884 [last accessed 26 May 2011].
21. White T. *Hadoop: The Definitive Guide*. Yahoo Press: CA, USA, 2010.
22. Casanova H. SimGrid: a toolkit for the simulation of application scheduling. In *Proceedings of First IEEE/ACM International Symposium on Cluster Computing and the Grid*, Brisbane, Qld. , Australia, 2001; 430–437.
23. Buyya R, Murshed M. GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience* 2002; **14**(13-15):1175–1220.
24. Lim S, Sharma B, Nam G, Kim E, Das C. MDCSim: a multi-tier data center simulation, platform. In *Proceedings of IEEE International Conference on Cluster Computing*, New Orleans, Louisiana, USA, 2009; 1–9.
25. Garg S, Buyya R. NetworkCloudSim: modelling parallel applications in cloud simulations. In *Proceedings of the 4th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2011)*, Melbourne, Australia, 2011; 105–113.

DOI: 10.1002/spe