2011 Fourth IEEE International Conference on Utility and Cloud Computing

NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations

Saurabh Kumar Garg and Rajkumar Buyya
Cloud Computing and Distributed Systems (CLOUDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
Email: sgarg@csse.unimelb.edu.au

***Abstract*—As interest in adopting Cloud computing for various applications is rapidly growing, it is important to understand how these applications and systems will perform when deployed on Clouds. Due to the scale and complexity of shared resources, it is often hard to analyze the performance of new scheduling and provisioning algorithms on actual Cloud testbeds. Therefore, simulation tools are becoming more and more important in the evaluation of the Cloud computing model. Simulation tools allow researchers to rapidly evaluate the efficiency, performance and reliability of their new algorithms on a large heterogeneous Cloud infrastructure. However, current solutions lack either advanced application models such as message passing applications and workflows or scalable network model of data center. To fill this gap, we have extended a popular Cloud simulator (CloudSim) with a scalable network and generalized application model, which allows more accurate evaluation of scheduling and resource provisioning policies to optimize the performance of a Cloud infrastructure.**

***Keywords*-Cloud Computing, Modelling and Simulation, Parallel Applications**

I. Introduction

Recently, Cloud computing paradigm [1] has rapidly gained the attention of various communities including researchers, businesses, consumers, and government organisations. Due to various tempting benefits such as elastic and scalable on-demand resources, Cloud service users are exploring the ability of such scalable platforms for the efficient and cost-effective execution of their applications such as High Performance Computing (HPC), e-commerce, social network and web applications. To successfully use 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.

Evaluation of alternative designs or solutions for Cloud computing on real test-beds is not easy due to several reasons. Firstly, public Clouds exhibit varying demands, supply patterns, system sizes, and resources (hardware, software, network) [2]. Due to such unstable nature of Cloud resources, it is difficult to repeat the experiments and compare different solutions. Secondly, there are several factors which are involved in determining performance of Cloud systems or applications such as users's Quality of Service (QoS) requirements, varying workload, and complex interaction of several network and computing elements. Thirdly, the real experiments on such large-scale distributed platforms are considerably time consuming and sometimes impossible due to multiple test runs in different conditions. Therefore, a more viable solution is to use simulation frameworks which will 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 for modelling real Cloud environments. For evaluating a scheduling algorithm in a Cloud computing environment, a simulator should allow users to define two key elements: (i) an application model specifying the structure of the target applications in Clouds, typically in terms of computational tasks and data communication between tasks; (ii) a platform model of Cloud computing data centers specifying the nature of the available resources and the network by which they are interconnected. Clouds currently deploy wide variety of applications both from industrial enterprises and scientific community[3]. The applications from industries vary from simple web applications to the complex business workflows. Similarly, scientific applications from areas such as climate modelling, drug design and protein analysis range from massively parallel applications to message passing applications. In terms of the platform, Cloud computing is quite different from traditional distributed computing platforms defined by service-oriented features such as resource elasticity, multiple-level of services and multi-tenancy of resources.

Currently available Cloud specific simulation solutions such as CloudSim [4] and GreenCloud [5] view datacenter resources as a collection of Virtual Machines (VM). As a result, they integrate either a very simplistic application models without any communicating tasks, or limited network model within the data center. Some of these application

978-0-7695-4592-9/11 $26.00 © 2011 IEEE
DOI 10.1109/UCC.2011.24

and platform models, while perhaps appropriate for some type of applications such as single server web applications or parameter sweep applications, become inadequate when used to model a real Cloud computing environment running different types of applications from different customers [3]. For example, precise evaluation of scheduling algorithms for scientific applications, such as message passing parallel applications or multi-tier web applications, requires modelling of the datacenter's interconnection network. Since, current Cloud simulators have no support for both of these features simultaneously, these limitations may either result in non-realistic data center solutions or in inaccurate solutions for applications with communicating tasks.

To overcome these limitations of current Cloud simulators, we develop a simulation framework *NetworkCloudSim* which supports modelling of real Cloud data centers and generalized 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 real Cloud data centers, but simple enough to be amenable for analysis. In particular, we discuss issues and solutions in regard to modelling a data center's internal network and applications. Our simulation framework is developed as an extension of *CloudSim*. More precisely, our **contributions** are the following:

- We have designed a new Cloud simulation framework, *NetworkCloudSim*, which is equipped with more realistic application models than any other available Cloud simulator. The components of simulation framework are implemented as part of a widely used Cloud simulator, CloudSim to support applications with communicating elements or tasks such as MPI, and workflows.
- We have 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 have presented an experimental evaluation to show the importance of network modelling in developing accurate scheduling and resource management mechanisms for different Cloud applications.

II. Related Work

In this section, simulation frameworks closely related to our work are discussed. In the area of distributed computing, from past many years Grid computing vastly interested scientific community due to its advantages in delivering high-performance services for compute- and data-intensive scientific applications. To support the research and development of Grid middleware, several simulators such as SimGrid [6] and GridSim [7] have been proposed. Although these environments are capable of effectively and comprehensively modelling 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 multi-layer services (SaaS, PaaS and IaaS) of Cloud computing. In these tools, there is almost no support for modelling virtualized resources and application management environment.

In spite of 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 [8]. 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 modelling of communication aspects of the data center network. Being build on top of NS2, it implements a full TCP/IP protocol reference model which allow integration of different communication protocols such as IP, TCP and UDP with the simulation. However, this is also a disadvantage since it limits its scalability to only small data centers due to large simulation time and high memory requirements. MDCSim [8] 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 form of a directed graph. However, no network topology is designed for modelling an internal data center network.

All these three 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 & 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 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.

III. Simulating Applications with Communicating Tasks

A. CloudSim Architecture

The CloudSim simulator is currently the most sophisticated discrete event simulator for Clouds. It has many features which made us choose it for building our simulation environment on top of it. Figure 1 shows components of the CloudSim Architecture with the key elements of NetworkCloudSim (shown by dark boxes). In this section, we outline the functionality of different layers of CloudSim. The detailed design and functionality of NetworkCloudSim's elements will be discussed in the later sections.

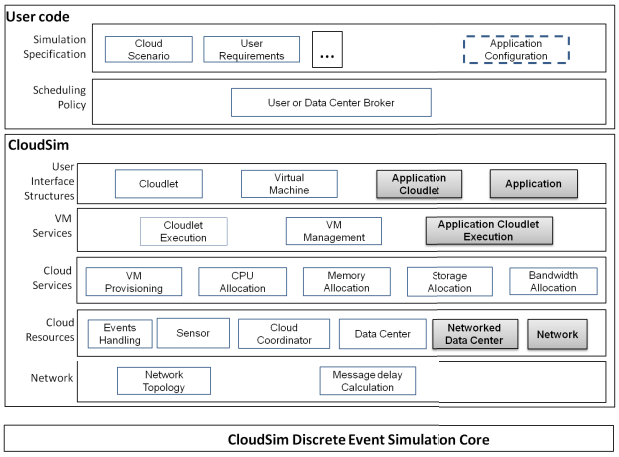


Figure 1: The CloudSim Architecture with NetworkCloudSim elements

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 sub-layers that model the core elements of Cloud computing. The bottommost sublayers model datacenter, Cloud coordinator and network topology between different datacenters. These components help in designing IaaS infrastructure. The VM and Cloud Services provide the functionality to design resource (Virtual Machine (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 datacenters. We will discuss the issues and solutions to support such features in the next section.

B. Design issues and solutions

Despite of several useful features of CloudSim, it cannot be used to model very complex data centers with different application models and networked resources. To make this environment more realistic, the following features and issues related to them are resolved.

1) Application Model: As discussed previously, simulation of parallel and distributed applications is not fully considered in most of current Cloud computing simulators. 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. The application models in Cloud computing can vary from multi-tier web applications such as e-commerce to scientific applications [3]. Typically such applications consist of several tasks, which communicate with each other. A task consists of some computation and communication phases. This is true for both scientific applications and web applications. A web application consists of several tiers, each tier running on a different server and communication is between these different tiers.

In order to simulate the behaviour of such complex parallel and distributed applications, new structures and functionality are added to CloudSim in this work. For helping users to model such communicating tasks, we designed the *NetworkCloudlet* class that represents a task executing in several phases/stages of communication and computation. Figure 2[1] shows how we can model these applications in NetworkCloudSim. To model 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 (*NetworkCloudlet*). Each element runs in a single virtual machine and consists of communicating and computing stages. Each computation stage can be defined either by the number of MIPS or seconds involved in it. The communications are characterized by the amount of transferred data.

Although, the above features enable users to model complex applications in their simulation environment, still the precise execution of such applications depend highly on the underlying network model. In the next section, we will discuss issues in designing a network model for a data center and how it is addressed in NetworkCloudSim.

2) Network modelling: Modelling comprehensive realistic network topologies within data center is an important consideration because the data latency due to oversubscription of resources such as network can affect Quality of Service offered to the Cloud customer. In addition, to accurately evaluate applications with communicating task and virtual machine migration, it is also necessary to take into account the network topology and bandwidth. In fact, in

[1]S-Sending the data stage and R-Receiving the data stage

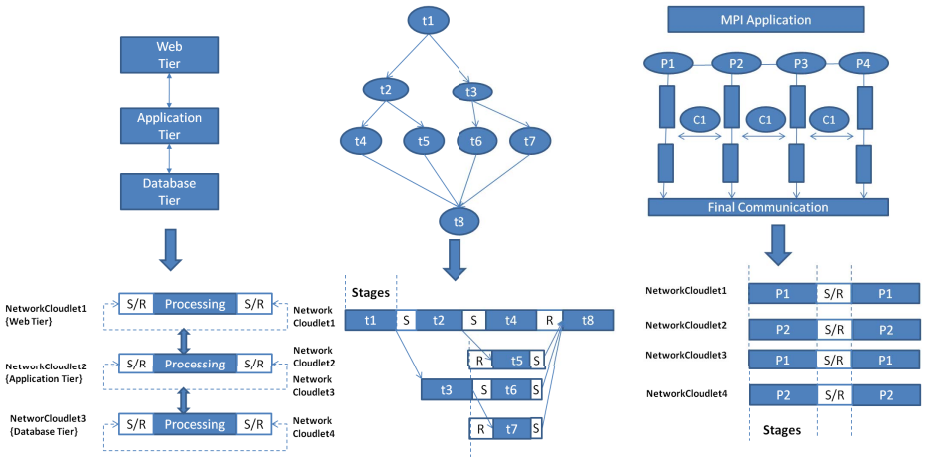


Figure 2: Modelling of Applications in NetCloudSim

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 virtual machine migration create significant network overhead [9], however currently in CloudSim only the CPU overhead is modelled. There are two issues in designing network:

- **Which model should be chosen?:** There are two ways to design a network within CloudSim: packet network or flow network model [10]. Both of these models are widely used in simulation environments for distributed systems and has their advantages and disadvantages. Among these two models, the flow network model results in a very low computational overhead in comparison to its counterpart. The flow network model also gives a good approximation to real network models such as TCP/IP [10] [11]. Since, the time is one of the key factor in simulation environments, the flow network model is designed within NetworkCloudSim.
- **What should be the topology of virtual machine interconnections?:** It is assumed subtly in the current CloudSim implementation that each VM is connected with all other virtual machines. The drawback of fully connected model is that it fails to model realistic data center environment. The communication links are generally shared and interconnected through fat-tree type of network architecture [12]. To tackle this issue, we have added three level 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.

Network flow Model Design: We consider a network flow model that captures the steady-state behaviour 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 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 can be 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.

Since CloudSim is an event-based simulator, where different system models/entities communicate via sending events, the event management engine of CloudSim is utilized to induce delays in transmitting 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, i.e each flow gets 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.

IV. Implementation

NetworkCloudSim extends CloudSim's [4] functionality with the introduction of concepts that model generalized 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 3 has the following main components.

1) **Classes to model a network topology:** To model a network within the datacenter, the following classes

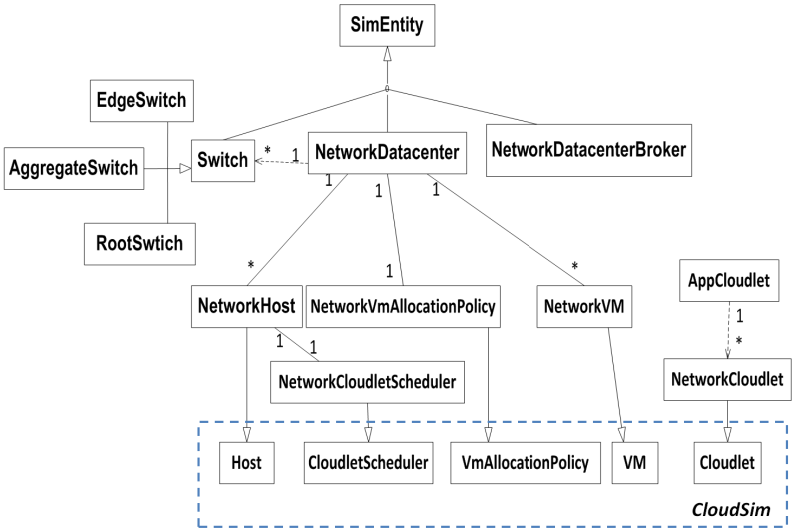


Figure 3: Class Diagram of NetworkCloudSim

have been added to the NetworkCloudSim.

- **Switch** represents a network entity which can be configured as a router or switch. It can model delays in forwarding any data to either host or another switch based on where the data belongs. Currently, to allow modelling various topologies, three types of switch are modelled: root, aggregate and edge switch. The edge switch is directly connected to hosts and has uplinks connected to another switch. Aggregate switch has uplinks and downlinks to switches. The root switch is modelled as a network entity that is directly connected to the Internet/outside data center and has downlink to other switches.
- **NetworkPacket and HostPacket:** These classes represent a data flow from one VM to another in a data center. Since 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 to 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. Whereas NetworkPacket is the packet which travel from one server to another. Each packet contains ids of the sender VM and receiver VM, time at which it is send and received, type and virtual ids of tasks, which are communicating.

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.

2) **Classes for Application Modelling:** 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 can be computation, sending some data or receiving some data. 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.

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

V. Scheduler and Execution of NetworkCloudlet

As discussed previously, 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 Host level (i.e. scheduling of tasks on VMs) and another at the VM level where real applications are executed. For scheduling any application, networkcloudlets are scheduled on different VMs by the NetworkDatacenterBroker. At the VM level, networkcloudlet

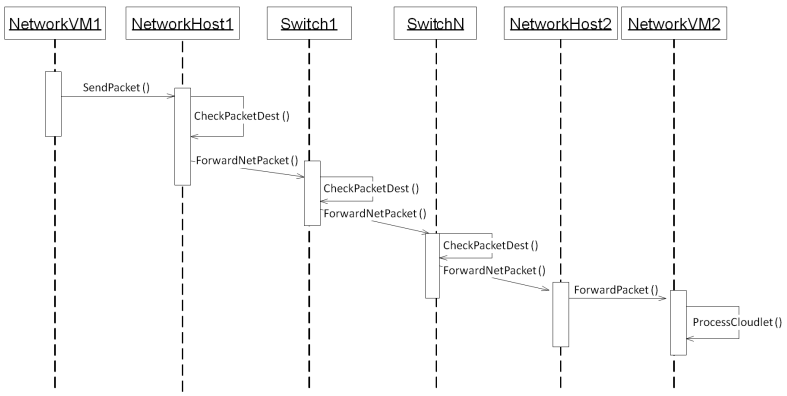


Figure 4: Sequence Diagram: Communication between NetworkCloudSim Entities

can run either in time shared or space shared mode. We have currently implemented a space shared scheduling policy which is more widely used. Algorithm 1 describes the execution process of a NetworkCloudlet on a VM. For each networkcloudlet, the scheduler checks the current stage of execution. There are four execution phases in which a networkcloudlet can be: $Send$, $Recv$, $Execution$ and $Finished$. If stage is $Execution$, the networkcloudlet's execution time is updated until the next scheduling event. If the stage is $Send$, the packet is constructed by the VM scheduler and submitted to the send-packet-queue of the VMs. After updating the execution stage of each networkcloudlet, VM scheduler will forward theses packets either to VMs on the same Host and to the switches (to forward to 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 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 non-blocked send approach is adopted such that a sender will not be blocked even though the corresponding receiver VM is not ready for 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 non-blocking message passing paradigm (such as $MPI\_Isend()$ and $MPI\_Irecv()$), which is a common practice in parallel applications.

For scheduling multi-tier applications, the currently implemented algorithm requires little modification since the nature of these applications is quite different from scientific applications. Multi-tier applications are more event based. Therefore, when a packet is received (Line 21), based on the packet 'type', the processing and data sent will vary. Therefore, in Figure 2 NetworkCloudlets has a circular array with three stages. NetworkCloudSim allows its users to configure and implement this event based logic according to their requirements.

VI. Datacenter 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 4. Using this example, one can model even more complex networks. Each VM sends packets from its $send\_packet\_queue$ to its host for further processing. The 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. Otherwise, the host creates a network packet and forwards it to the switch (generally called Edge Switch) NetworkHost is connected. The edge switch (Switch1) further decides whether the packet belong 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/switiching delays by sending an event to 'switch' itself. Based on the decision, the packet is either forwarded to other switches or to a connected host with a delay which is calculated based on 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.

VII. Performance Evaluation

To validate and understand the behavior of NetworkCloudSim, we conduct two 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 a use case study to understand the behavior of network components and the packet scheduling algorithms.

Algorithm 1 Cloudlet Scheduling and Execution on VM

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 which has submitted the networkcloudlet for execution on VM; $packet\_recv\_queue$ and $packet\_send\_queue$: queues having packets which are received and to be send by the VM

1: **for** Each NetworkCloudlet cl in $current\_execution\_queue$ **do**
2: **if** $cl.currstage = `Execution'$ **then**
3: Execute the networkcloudlet
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**

A. Comparison of Simulated and Real Execution Time

To validate the accuracy of simulation results of NetworkCloudSim, we compared the execution of a MPI application on a real infrastructure with simulated execution in NetworkCloudSim. For the experiments, we obtained traces of a controlled MPI program on a small scale real infrastructure using the *mpilog* tool using the methodology given by Miguel-Alonso et al. [13]. In the the MPI application, the main process generates several random numbers and then send the data to all other processes. The topology and configuration of the infrastructure are given in the Figure 5. Each host has 2 Xen VMs, each one with 2 cores and 1.5GB RAM. All the hosts are connected by a 100 megabits 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: a) varying number of communicating processes with 1000000 MPI_INT elements transferred from one process to another, and b) varying the amount of data transferred from the main process to other 7 processes. Figure 6(a) and 6(b) show the experimental results. The results from real execution is shown as "Measured" while 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 bandwidth is shared. This led to an increase in the total execution time. Figure 6(a) and 6(b) clearly show 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 process increases the simulation results match very closely to the measured ones. Therefore, we can conclude that NetworkCloudSim is effective in modelling the execution of parallel applications.

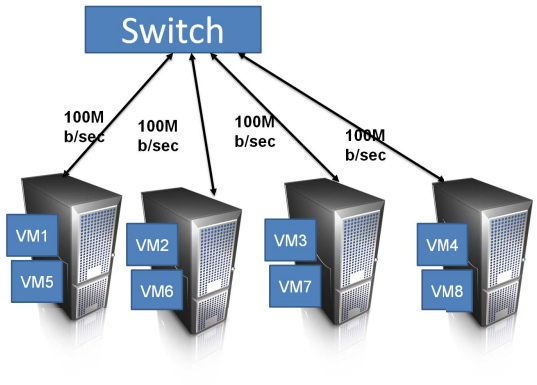
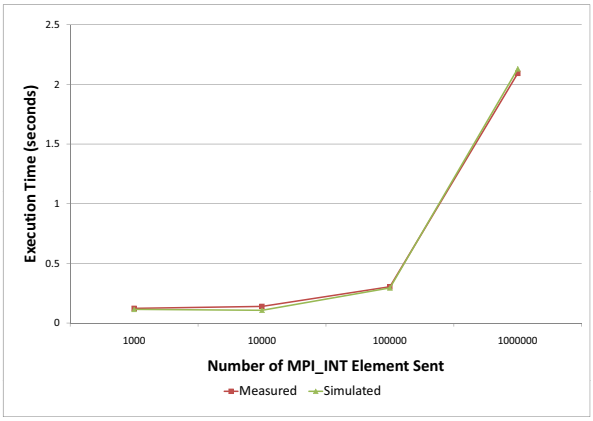


Figure 5: Experimental Datacenter Infrastructure

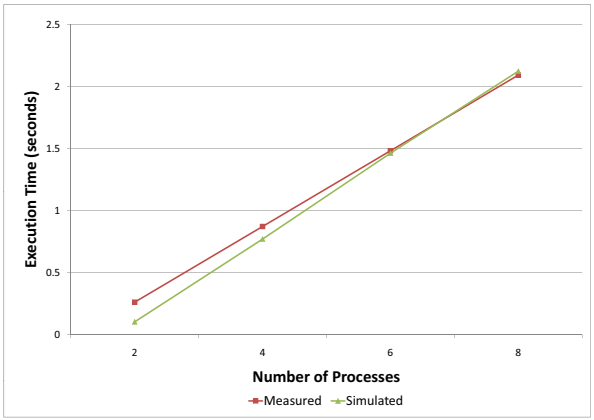
B. Evaluation of Task Assignment and Scheduling Policies

We have conducted a case study 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/second. Again, a very small scale data center with the configuration used previously is modeled. We compared firstly the effect of the resource allocation to each task of the application and secondly the effect of scheduling each task on the allocated VM. For the first set of results, we compared two scheduling policies:

1) **Random-NonOverlap:** In this scheduling policy, a VM executes the networkcloudlets in the first come first basis and therefore, other networkcloudlets are not



(a) Effect of Data Sent



(b) Effect of Communicating Processes

Figure 6: Measured and Simulated Execution Times of the MPI application

executed unless the currently executing one is finished.

2) **Random-Overlap:** In this scheduling policy, VM starts executing the networkcloudlets in the front of waiting queue if the currently executing task is just waiting for the data (packet) to arrive from other peer tasks.

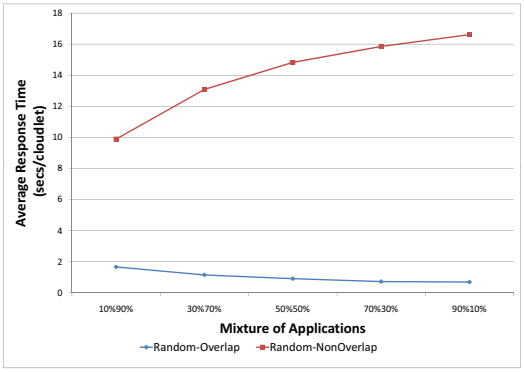
In the above policies, Random signifies that allocation of a VM to a task is done randomly. Figure 7(a) presents the first set of results where X-axis represent the ratio of the two types of applications (a%b%, i.e., a% of parallel applications and b% of parameter sweep applications). We can observe from the figure that the average response time (execution time+waiting time) of networkcloudlets for Random-Overlap is quite low in comparison to 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 we can observe from Figure 7(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 while it is decreasing in case of the Random-overlap scheduling policy. The reason for such a behavior in the case of the Random-NonOverlap scheduling policy is the increase of the communication delay with the increased proportion of applications having communicating tasks. In case of the Random-Overlap scheduling policy, this delay increase is neutralized by the overlapping of computation of one task and communication of another task.

In the second set of results, we compare the impact of resource allocation policy to understand how scheduling of communication tasks in different locations of datacenter impacts the response time. Figure 7(b) presents the results comparing two allocation policies Random-Overlap and RoundRobin-Overlap. In case of the RoundRobin-Overlap resource allocation and scheduling policy, tasks are assigned to VMs in a Round robin manner while for the other policy, tasks are randomly assigned to VMs. It can be noticed from Figure 7(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 Random-Overlap policy becomes higher than 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 above observations that the modelling of network is an essential part of the Cloud simulations.

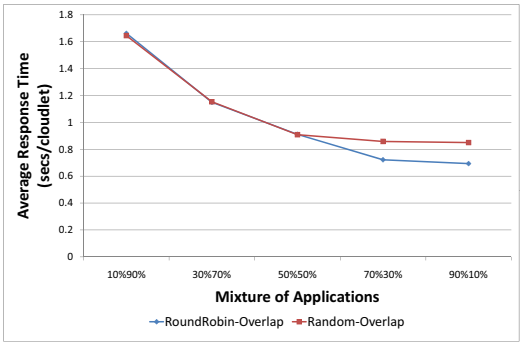
VIII. Conclusions

Use of simulation frameworks such as CloudSim is becoming increasingly popular in the Cloud computing community due to 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 present a simulation framework which supports the modelling of essential data center resources such as network and computational resources, and wide variety of application models such as parallel application, workflow and parametric sweep. Even a multi-tier web application can be modelled with little modifications. Our simulation framework is built on top of a widely used simulator, i.e., CloudSim.

We presented the main components of the NetworkCloudSim with their functionality and how different network topologies and different parallel applications can be modelled. 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



(a) Effect of Scheduling Processes (Cloudlet)



(b) Effect of Application Allocation

Figure 7: Case Study Results

and scheduling policies shows how NetworkCloudSim can help in building advance scheduling and resource allocation mechanisms for Clouds. 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 development of more power efficient resource management schemes rapidly 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 to packet level model. In future, we will integrate packet level network model in CloudSim so that users can simulate those Cloud applications which require precise network configurations.

ACKNOWLEDGMENTS

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.

REFERENCES

[1] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] J. Napper and P. Bientinesi, "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.

[3] J. Varia, *Cloud Computing: Principles and Paradigms*. Wiley Press, 2011, ch. 18: Best Practices in Architecting Cloud Applications in the AWS Cloud, pp. 459–490.

[4] R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[5] D. Kliazovich, P. Bouvry, and S. Khan, "Greencloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, 2010. [Online]. Available: http://dx.doi.org/10.1007/s11227-010-0504-1

[6] H. Casanova, "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.

[7] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and Computation: Practice and Experience*, vol. 14, no. 13-15, pp. 1175–1220, 2002.

[8] S. Lim, B. Sharma, G. Nam, E. Kim, and C. Das, "MDCSim: A multi-tier data center simulation, platform," in *Proceedings of IEEE International Conference on Cluster Computing, New Orleans, Louisiana, USA*, 2009.

[9] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, "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.

[10] J. Broberg and R. Buyya, *Grid Computing: Infrastructure, Service, and Applications*. CRC, 2009, ch. Flow Networking in Grid Simulations.

[11] H. Casanova, "Network modeling issues for grid application scheduling," *International Journal of Foundations of Computer Science*, vol. 16, no. 2, pp. 145–162, 2005.

[12] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 51–62, 2009.

[13] J. Miguel-Alonso, J. Navaridas, and F. Ridruejo, "Interconnection network simulation using traces of mpi applications," *International Journal of Parallel Programming*, vol. 37, no. 2, pp. 153–174, 2009.