TeachCloud: a cloud computing educational toolkit

Yaser Jararweh, Zakarea Al-Shara, Moath Jarrah, Mazen Kharbutli, Mohammad N Alsaleh

TeachCloud: A Cloud Computing Educational Toolkit

Y. Jararweh* and Z. Alshara

Department of Computer Science,
Jordan University of Science and Technology, Jordan
E-mail:yijararweh@just.edu.jo
* Corresponding author
E-mail:zakarea.alshara@gmail.com

M. Jarrah and M. Kharbutli

Department of Computer Engineering,
Jordan University of Science and Technology, Jordan
E-mail:mjarrah@just.edu.jo
E-mail:kharbutli@just.edu.jo

M. N. Alsaleh

Department of Software and Information Systems
University of North Carolina at Charlotte
Charlotte, NC, USA
E-mail: malsaleh@uncc.edu

Abstract: Cloud computing is an evolving and fast-growing computing paradigm that has gained great interest from both industry and academia. Consequently, universities are actively integrating cloud computing into their IT curricula. One major challenge facing cloud computing instructors is the lack of a teaching tool to experiment with. This paper introduces *TeachCloud*, a modeling and simulation environment for cloud computing. TeachCloud can be used to experiment with different cloud components such as: processing elements, data centers, storage, networking, Service Level Agreement (SLA) constraints, web-based applications, Service Oriented Architecture (SOA), virtualization, management and automation, and Business Process Management (BPM). Also, TeachCloud introduces MapReduce processing model in order to handle embarrassingly parallel data processing problems. TeachCloud is an extension of CloudSim, a research-oriented simulator used for the development and validation in cloud computing.

Keywords: Teaching Cloud computing; CloudSim; Network topologies; MapReduce; SLA management; Rain Workload generator; BPM.

1 Introduction

Cloud computing is a new computing paradigm that is continuously evolving and spreading. Many experts believe it will become the dominant IT service delivery model by the end of the decade. As a result, universities worldwide are introducing cloud computing technologies in their curricula by updating existing courses or developing new ones. Cloud computing builds on a wide range of different computing technologies such as high-performance computing, distributed systems, virtualization, storage, networking, security, management and automation, Service-Oriented Architecture (SOA), Business Process Management (BPM), Service-Level Agreement (SLA), Quality of Service (QoS), etc. This complexity presents a major obstacle for the students to grasp and thoroughly understand cloud computing. The diversity of cloud computing related areas requires the students to put great efforts to understand each one of these areas alone in addition to integrating them in a single platform. This creates many challenges in teaching this rising technology for two main reasons: First, in order for a student to be able to grasp all aspects of cloud computing, the student must have adequate and sufficient background in the many areas listed above. Unfortunately, this is not always the case. For example, we found that while Computer Engineering students had sufficient background in the areas of high-performance computing, distributed systems, virtualization, storage, and networking, Computer Science students were lagging in these areas. On the other hand, Computer Science students had sufficient background in SOA, BPM, and management, areas Computer Engineering students were lagging in. Second, and more importantly, the fact that there are no teaching tools to cover the different aspects of cloud computing as a whole makes teaching more theoretical-oriented, and therefore, less effective. Although there exists teaching tools for most of the cloud components alone, no full-system tool exists yet.

Table 1 Challenges in Teaching Cloud Computing as Identified by Students

Challenge or Difficulty	Percentage Agreed
Lack of hands-on experience	93%
Lack of a comprehensive textbook	63%
Lack of help material on the Internet	57%
Insufficient background	17%
Vast amounts of different topics	17%

At Jordan University of Science and Technology, we were one of the first universities in the Middle East to introduce cloud computing concepts in our courses for both graduate and undergraduate students. Our first attempt to teach a cloud computing course was during the spring semester 2011 as an elective course for senior students. During the course, we identified several key challenges in teaching cloud computing that we list in Table 1. At the end of the course, we conducted a students' survey and asked the students to identify which challenges they thought were most important. The vast majority of the students (93%) agreed that the biggest challenge was lack of hands-on experience. Although a real cloud system can be used by the students to run experiments, the criticality and frangibility of the system poses many limitations and risks. A better solution is the use of a full cloud-system simulator tool with which students can play around and experiment risk-free. Other, but less important, challenges include lack of a single comprehensive textbook that covers all aspects of cloud computing, limited help material on the Internet, insufficient students' background, and the sheer amount of topics covered during a single semester.

Consequently, we moved forward to find an experimental environment to be used in teaching cloud computing. One available option was the CloudSim simulator from the University of Melbourne, Australia (Calheiros et al. (2011)). CloudSim is a very promising tool for teaching cloud computing. However, CloudSim shows several limitations and shortcomings: First, it lacks a Graphical User Interface (GUI) which is important to students as it makes it easier for them to use the tool. Second, it is built on top of a grid computing environment which puts limitations on the infrastructures it can simulate. Third, it only includes a basic and simplified network model and a limited workload generator. Fourth, CloudSim does not provide efficient processing for embarrassingly parallel data problems. Fifth, it lacks BPM and SLA components.

As a result of these limitations in CloudSim, we started to develop *TeachCloud* as a cloud computing educational toolkit for use in cloud computing courses. TeachCloud uses CloudSim as the basic design platform and introduces many new enhancements on top of it. These enhancements and extensions include:

- Developing a GUI for the toolkit.
- Adding the Rain cloud workload generator (Beitch et al. (2010)) to the CloudSim simulator.
- Integrating MapReduce framework into CloudSim to handle embarrassingly parallel data processing paradigms.
- Adding new modules related to SLA and BPM.
- Adding new cloud network models (VL2, BCube, Portland, and DCell) to represent the actual topologies that exist in real cloud environments.
- Introducing a monitoring outlet for most of the cloud system components.
- Adding an action module that enables students to reconfigure the cloud system and study the impact of such changes on the total system performance.

One use case scenario for TeachCloud will be presented in section 7. The rest of the paper is organized as follows: Section 2 introduces the CloudSim simulator and the TeachCloud toolkit interface. Section 3 discusses the Cloud workload generator and the Rain generator that TeachCloud supports. In section 4, the MapReduce data processing model is introduced. We discuss the SLA constraints and BPM in Section 5. Then in Section 6, we describe the network topologies that were integrated in the TeachCloud toolkit. Section 7 shows some simulation results that were generated using TeachCloud. And finally, Section 8 concludes the paper.

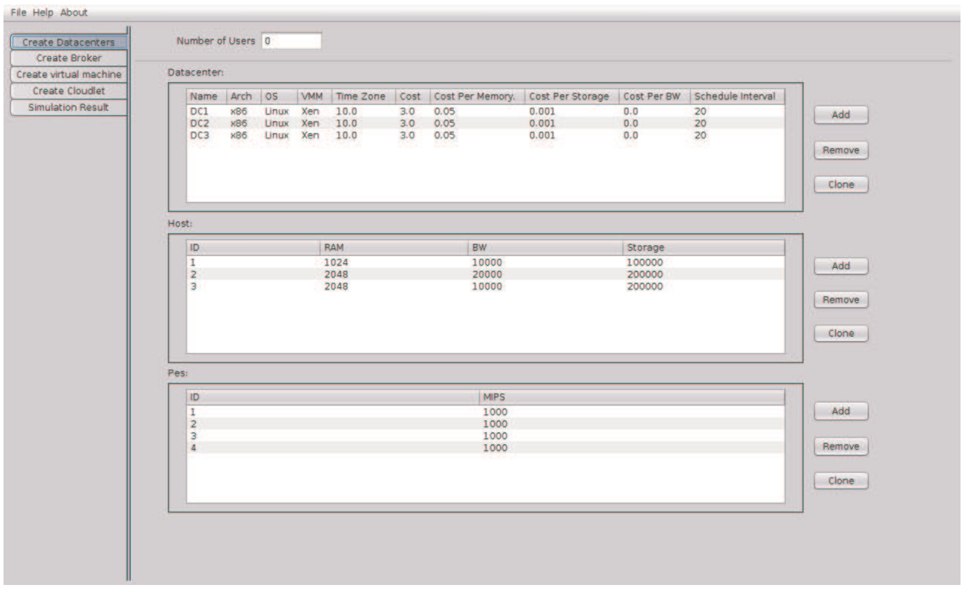
2 CloudSim: The underlying framework for TeachCloud

CloudSim is a cloud computing modeling and simulation tool that was developed at the University of Melbourne, Australia (Calheiros et al. (2011)). It aims to provide cloud computing researchers with a comprehensive experimental tool to conduct cloud computing-related research. It supports the modeling and simulation of various cloud computing components, including power management, performance, data centers, computing nodes, resource provisioning, and virtual machines provisioning (Kim et al. (2009)).

CloudSim is a layered design framework written in Java and was initially built on top of SimJava and GridSim (Buyya et al. (2009)). SimJava is a discrete event simulator that has been widely used. GridSim is a grid computing simulator that uses SimJava library. However, SimJava has several scalability limitations that led the developers of CloudSim to implement a new discrete event management framework which is the CloudSim core simulation engine ((Calheiros et al. (2011)). The following is a brief description of the different cloud computing features and components and how CloudSim models them:

- *Data centers* represent the core infrastructure in a cloud system and include the hardware and software stack. In defining the data centers, the modeler needs to identify the number of hosts in each data center.
- *Hosts* are modeled using the class *Host* which models a physical resource such as a computer or a server. A data center can have multiple hosts.
- *Cloud federation* governs multiple geographically distributed cloud systems (private and/or public) that are inter-connected through a network topology.
- *Cloud tasks* are represented by *Cloudlets* which are the cloud-based application services. In this class, computational metrics are used to model the complexity of applications.
- *Brokering* is performed through the use of the *DatacenterBroker* class. The researcher establishes the contracts between the cloud users and the service providers.
- *Storage in the grid* is modeled using the *SANStorage* class.
- *Virtual machines (VM)* are modeled in the class *VirtualMachine*.
- *Coordination* is performed using the *CloudCoordinator* class which models the communication between different cloud coordinators and cloud brokers. In addition, it monitors the internal state of each data center that is connected to it.
- *Bandwidth provisioning services* are modeled using the *BWProvisioner* class.
- *Memory provisioning* and the policies for allocating memory to VMs are modeled using the *MemoryProvisioner* class.
- *Virtual machine provisioning* is performed using the *VMProvisioner* class which allocates VMs to the hosts.
- *VM allocation policies* are maintained by the class *VMMAllocationPolicy* which models the policies for allocating processing power to VMs. Allocation can be space-shared or time-shared.
- *Power consumption* is modeled in the *PowerModel* class which models data centers power consumption.

To the best of our knowledge, CloudSim was never used as a teaching tool. This is mainly related to the difficulties in using it by students. Nevertheless, the capabilities of CloudSim are very promising for cloud computing teaching. In this work, an extension is provided in order to make it convenient for teaching. Hence, a Graphical User Interface along with several new key features were added. The GUI allows students to easily create the main components in a cloud system. For example, as shown in Figure 1, a student can create data centers and define their parameters such as the number of hosts on each data center. Figures 2, 3, and 4 demonstrate the creation of Cloudlets, virtual machines, and brokers, respectively. The new features that were added as part of TeachCloud are described in Sections 3, 5, and 6.



Number of Users: 0

Datacenter:

Name	Arch	OS	VMM	Time Zone	Cost	Cost Per Memory.	Cost Per Storage	Cost Per BW	Schedule Interval
DC1	x86	Linux	Xen	10.0	3.0	0.05	0.001	0.0	20
DC2	x86	Linux	Xen	10.0	3.0	0.05	0.001	0.0	20
DC3	x86	Linux	Xen	10.0	3.0	0.05	0.001	0.0	20

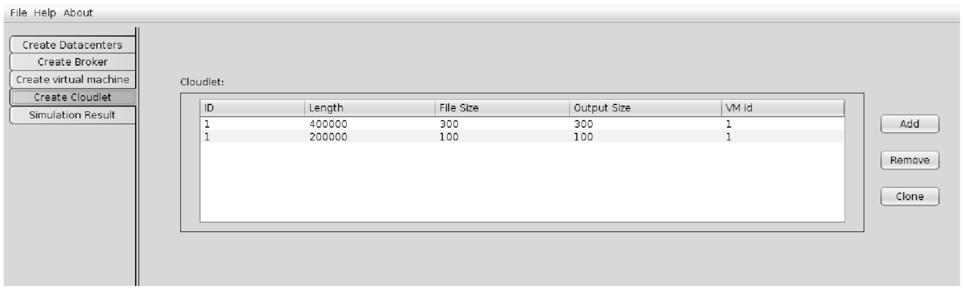
Host:

ID	RAM	BW	Storage
1	1024	10000	100000
2	2048	20000	200000
3	2048	10000	200000

Pes:

ID	MIPS
1	1000
2	1000
3	1000
4	1000

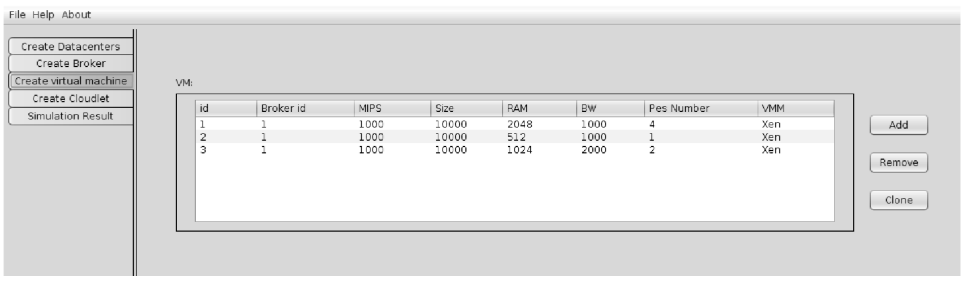
Figure 1 Creating data centers in TeachCloud



Cloudlet:

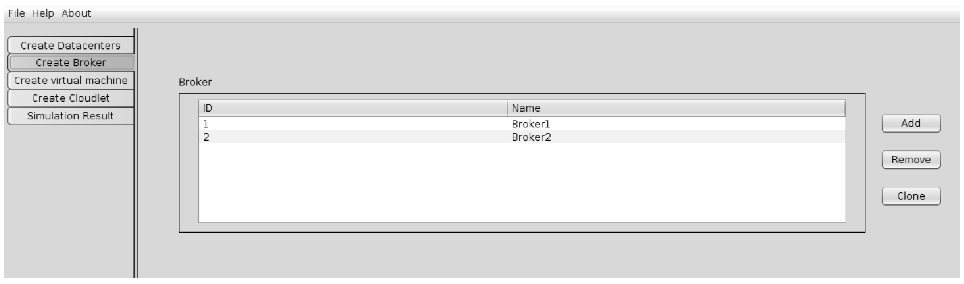
ID	Length	File Size	Output Size	VM id
1	400000	300	300	1
1	200000	100	100	1

Figure 2 Creating Cloudlets in TeachCloud



id	Broker id	MIPS	Size	RAM	BW	Pes Number	VMM
1	1	1000	10000	2048	1000	4	Xen
2	1	1000	10000	512	1000	1	Xen
3	1	1000	10000	1024	2000	2	Xen

Figure 3 Creating virtual machines in TeachCloud



ID	Name
1	Broker1
2	Broker2

Figure 4 Creating cloud brokers in TeachCloud

3 Woload generation and the Rain workload generator

Cloud computing (CC) systems promise to handle and fulfill different customers' workload requirements. CC applications and services vary widely ranging from data intensive (i.e. big-data) applications to financial applications and web services. CC workloads differ in their size, execution times, memory footprints, and QoS requirements, among other factors. Moreover, such diverse workloads are very dynamic and show different patterns of execution phases during an application's life span. Hence, CC systems are required to dynamically adapt their operational environment on the fly to cope with these workload dynamics. Workloads characterization is very crucial in a CC system in order to achieve proper resources provisioning, cost estimation, and SLA considerations. CC application developers and hosting companies need to predict changes in the workload patterns to be able to adjust promptly when variations in the patterns occur. Moreover, a detection of any possible system bottlenecks is needed before actually porting the application to the real system. Cloud applications updates after deployment may also cause problems in the actual cloud system. These updates may change the workload patterns and dynamics. These changes will affect the cloud system resources allocation and QoS measurements. As a result, a mechanism to test cloud applications before their actual deployment or updates is of the utmost necessity.

Based on the aforementioned facts, an accurate and comprehensive workload modeling environment is an essential component in any CC simulation tool. Current workload modeling environments for systems such as grid computing do not fit or match the workloads found in real CC systems. CloudSim provides a basic cloud workload modeling structure that fails to capture many of the characteristics of current CC workloads. Hence, CloudSim users need to extend available modules to generate the required patterns. CloudSim models the workload as an extension to the Cloudlet entity and introduces the Utilization Model which is an abstract class that needs to be extended by users to implement the workload patterns related to the application (Calheiros et al. (2011)). The Utilization Model provides methods and variables in order to specify resources requirements of applications during the deployment phase. The input to this method is a discrete time parameter, while the output is a percentage of computational resources that is required by the Cloudlet. CloudSim workload modeling requires CloudSim users to generate the required workload patterns by overwriting the $getUtilization()$ method. The main drawback of this approach stems from the fact that the user himself is required to accurately represent the application workload patterns with no guarantees of their correctness. Consequently, any related measurements based on this workload may be misleading to the user. Also, this approach will incur an extra overhead on the user (i.e. the student in our case). Moreover, it suffers from the lack of flexibility and scalability.

TeachCloud provides advanced workload modeling capabilities by introducing the Rain workload generator framework from the University of California at Berkeley (Beitch et al. (2010)). The Rain framework presents accurate and comprehensive CC workload characteristics. Rain is an open source workload generator that is available to the cloud computing community. This motivates us to utilize the advanced capabilities of Rain to achieve a meaningful load generator that is able to handle different CC applications. TeachCloud replaces the basic workload generator framework available in CloudSim by integrating the Rain workload generator as demonstrated in figure 5. The integration process is simplified by the fact that both CloudSim and Rain were written in Java. The three main workload characteristics that Rain provides are: 1) Variations in workload intensity (e.g. small, medium, and large amounts). 2) Variations in the operations mix performed by the system components (e.g. data intensive vs. computing intensive and tasks dependencies). 3) Variations in the data access patterns and frequency, also known as data hot spots. Predicting data hot spots enables efficient access to them which reduces access time and overhead. It is obvious that the aforementioned workload variations are closely related to each other and need a careful handling by any successful workload generator.

3.1 Rain workload generator architecture

Rain is a unique statistics-based workload generator that overcomes many of the other workload generators limitations. It provides the user with ability to easily reuse, configure, and schedule cloud applications workloads. The cloud applications variations in Rain are achieved efficiently by using empirical probability distributions. Rain architecture handles the variations in the load intensity and the mix of operations by using load scheduling mechanisms. On the other hand, changes in the mix of operations and the realization of access patterns and data hot spots are done using application-specific request generators (Beitch et al. (2010)). Moreover, Rain architecture diverges from current workload

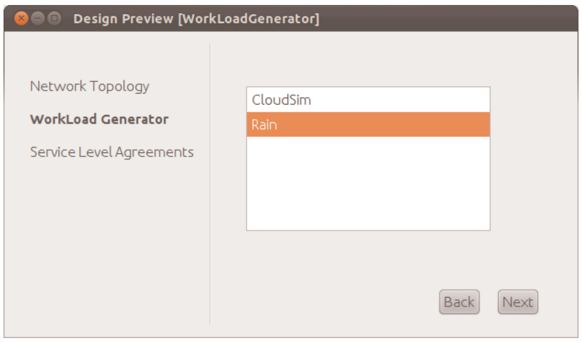


Figure 5 Rain workload generator as part of the TeachCloud tool

generators in the way workload requests are handled. While current generators couple request generation with its execution, Rain decouples request generation from request execution which adds flexibility to requests management. Another limitation of current generators is related to the fact that the thread that generates the request is also responsible for its execution; this is known as thread-affinity problem. Thread-affinity problem impacts the performance of the generator as one thread cannot generate new requests until it completes the current one. Rain solves this problem by changing the logic of thread and request relation. Any request generated by a certain thread can be executed by any other thread. This solution provides flexibility, scalability and improves the generator performance.

Rain architecture consists of the following components (Beitch et al. (2010)):
1) The *Scenario* component is responsible for the experiment configuration parameters (e.g. maximum number of users, experiment duration, etc.).
2) The *Generator* component uses the configuration from the Scenario component to create requests and operations that will be used by the other components.
3) The *Scoreboard* component is used to present experiment results and summary.
4) The *Threading* component contains the available threads for executing the tasks produced by the request generator component.
5) The *Benchmark* component is responsible for running the entire experiment. It interacts with other components by loading the Scenario that needs to be handled by the threading components. It also initializes the Scoreboard and presents the results at the end of the experiment.

3.2 Rain workload generation scenario component

A new workload will be initialized by using the scenario where each entity, i.e. user, will be assigned to a certain generator and assigned to a thread. The thread will request a new operation from the generator. When the thread finishes executing the operations assigned by the generator, it will generate a summary (e.g. status, execution results, etc) and write its details on the scoreboard. All these steps will be controlled by the benchmark component. TeachCloud users will use the workload generator in many ways. First, students can experiment with different workload patterns and mix of operations while using a fixed system configuration (e.g. VMs, network, etc.). This will help in understanding how workloads impact CC system components utilization and its impact on the SLA. Second, students can conduct experiments with different resources provisioning algorithms and assess how these algorithms react to workload variations. Third, students will be able to study the impacts of different workload patterns in the CC system components (VMs, network topology, etc.). Fourth, students can apply an intensive workload scenario to detect any bottlenecks in the system.

4 MapReduce Model

MapReduce (Dean and Ghemawat (2004)) has been widely used as a strong parallel data processing model. It is a programming model that has efficiently solved problems of large datasets using large clusters of machines. MapReduce is used in distributed grep, distributed sort, web-link graph reversal, web-access log stats, document clustering, machine learning, etc. (Xiao and Xiao (2011)). Cloud computing is a suitable environment for processing and analyzing terabytes of data through utilization of many resources connected through a topology (Armbrust et al. (2009)). Most cloud providers such as Amazon EC2, Microsoft Azure, Google, Yahoo and Facebook, adopted MapReduce in their computing environments.

Because on the fact that it is difficult to perform benchmarking experiments for MapReduce in real infrastructures, TeachCloud gives a simulation solution to achieve that. Users can use TeachCloud toolkit to perform experiments under non-static conditions (e.g. availability and workload pattern) in a controllable environment where tests can be re-executed.

4.1 Design and implementation of MapReduce

This subsection provides finer details about the fundamental classes of MapReduce. The class design diagram for MapReduce and its correlation with CloudSim is shown in Figure 6. The list of classes are:

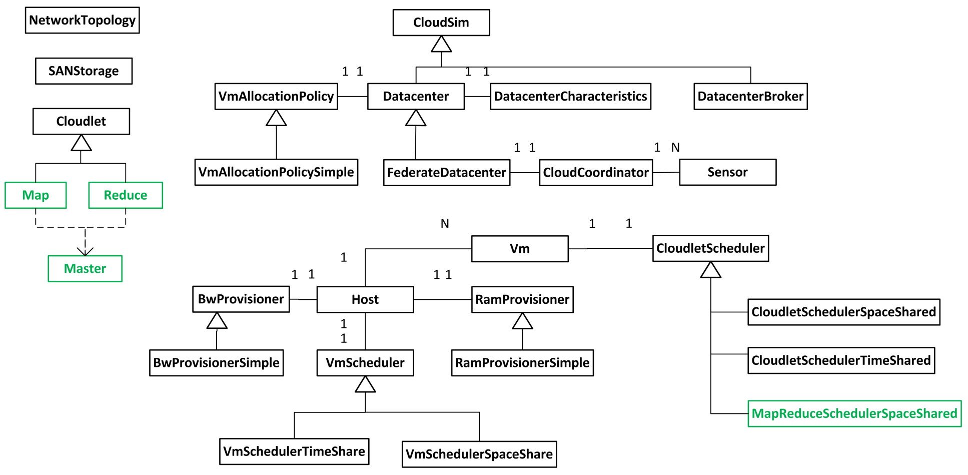


Figure 6 MapReduce class design diagram

1. Master: This class contains instances of all mappers and reducers that belong to the same user. Also, it contains information about the status and data locations during the execution of the mappers and reducers.

2. Map: It models the mapper's information and behaviors such as: input dataset size, output dataset size, and utilization model.

3. Reduce: It models the reducer's information and behaviors such as: input dataset size, output results size, and utilization model.

4. MapReduceSchedulerSpaceShared: This class extends an abstract class CloudletScheduler. It represents the behavior's policy between the mapper and reducer. For example, it determines the share policy of processing power among mappers and reducers in a virtual machine.

In the MapReduce algorithm, the input data is partitioned and distributed to a set of sub-problems which in turn can be partitioned further into smaller sub-problems. This allows the computing grid or cluster to process small size problems in parallel. The output is constructed by collecting all the results back from the sub-problems in a fashion that yields a correct result for the initial problem. The illustration in Figure 7 shows the flow of a MapReduce operation (Dean and Ghemawat (2004)). When a user program simulates MapReduce tasks, the following sequence of actions occur:

1. The workload in the user experiment is read and parsed to initiate the list of map and reduce instances. Each instance has parameters that are used to simulate the MapReduce model. The parameters include: ID, input data size, output data size and utilization models.

2. The master node assigns the instances created in step 1 to computing nodes in the cluster. The master also keeps track of mappers and reducers information such as: datacenter id, host id, virtual machine id, and status (created, ready, waiting, running, success, failed, canceled). Then each mapper or reducer is submitted to the cloud computing environment using the utilization models with an idle state called ready state.

3. After the submission of the mappers (initially in ready state), the simulation of MapReduce starts while taking into account each map's status and utilization model.

4. When a mapper finishes processing, it stores the data set result in a specific location and informs the master of the address.

5. When all mappers finish processing, the master sends signals to all reducers to start working on the data sets that were produced.

6. During MapReduce simulation, TeachCloud toolkit continuously collects data to be displayed for statistical analysis when the simulation is over.

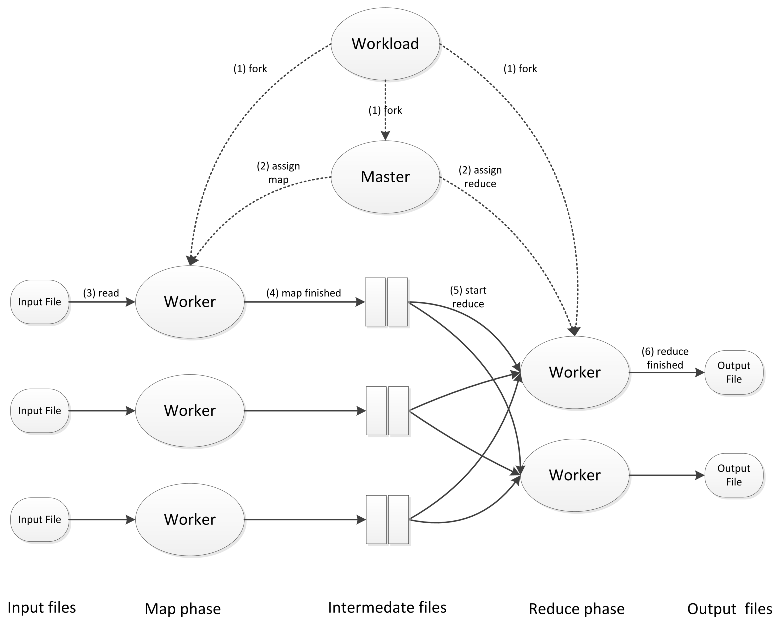


Figure 7 Flow of MapReduce operation

5 Service Level Agreements and Business Process Management

Traditionally, business aspects have received little attention in computing curricula. As a result, injecting the business flavor in cloud computing courses is no easy task. Service Level Agreement (SLA) and Business Process Management (BPM) are considered among the basic pillars of CC today (Faniyi and Bahsoon (2011)). Based on our teaching experience, students show much concern regarding SLA and BPM, and how they are related to CC. In TeachCloud we made efforts to introduce SLA to the students in a simple and effective manner. Starting from the basic definition of SLA as a contract between the service provider and the service user, which defines in measurable terms the service quality that the user expects to receive. TeachCloud provides a set of these measurable terms that the user can change while studying their impact on other system components. These measurable terms include, but are not limited to: number of users who can use the system simultaneously (e.g. 1000 active user), service availability (e.g. 99.9%), service cost, service outage handling terms, business continuity and disaster recovery, network performance, security measures, service metering tools available to the user, user compensation in case of SLA violation, and customer support by the service provider. To make things more realistic, we connect the SLA terms with the application sensitivity to the availability of the service, as some of the applications cannot handle availability of less than 100%. TeachCloud will enable the students to study the effect of the SLA terms on the cloud system, service cost with different SLA configurations and different QoS requirements. Further, TeachCloud will integrate the Web Service Level Agreement (WSLA) framework that is proposed to handle SLA monitoring and enforcement (Patel et al. (2009)). Figure 8 shows the SLA component of TeachCloud.

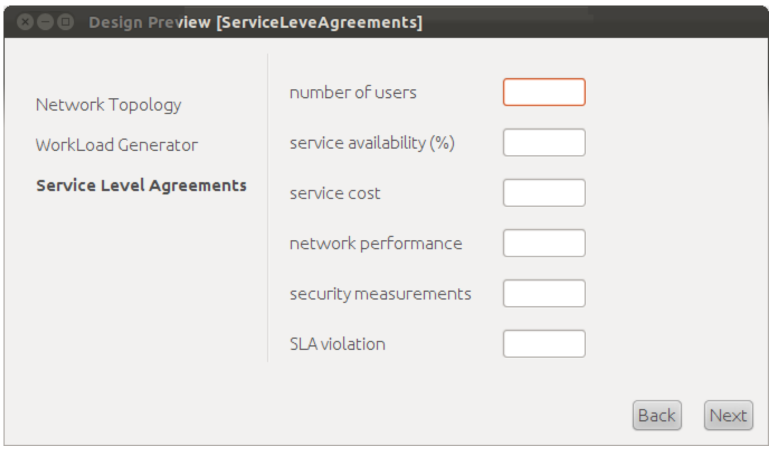


Figure 8 SLA parameters in TeachCloud

Furthermore and as a part of the next phase in TeachCloud development, BPM modeling will be introduced. This will help the students in understanding business-related issues in a cloud system and how CC supports customers' business success. Further, the students will be able to perform feasibility studies for running applications on the cloud system versus on a private computing system based on Capital Expenditure (CapEx) and Operating Expenditure (OpEx).

6 Network topologies in TeachCloud

Data centers in a cloud computing infrastructure normally contain thousands of physical servers connected through switches, routers or other network devices. Hence, the proper design of the data center network plays a critical role in the cloud environment since it directly affects the performance and the throughput of cloud applications. VMFlow (Mann et al. (2011)), for example, is a framework for placement and migration of VMs to reduce cost. It shows that the network topology along with network traffic demands should be taken into consideration in order to meet the objective of network power reduction. As shown in Figure 9, conventional data centers are organized in a hierarchy of three layers: core, aggregation and servers. The core layer consists of the core routers that connect the aggregation switches to the Internet. In the servers layer, the servers are organized in racks and connected to access switches (normally called Top-of-Rack switches) which are in turn connected to the aggregation switches. The aggregation layer may provide some functions such as domain service, location service, server load balancing, and more (Zhang et al. (2010)). This simple design suffers from some limitations as stated in (Greenberg et al. (2009)). First, while going up in the hierarchy, it becomes very hard technically and financially to sustain high bandwidth which results in preventing idle servers from being assigned to overloaded services which affects the data center performance. Second, the protocols that are normally used in the core and aggregation layers limit the utilization of the links to less than 50% of the maximum utilization. Third, the performance of communication depends on distance and the addressing scheme of the hierarchy which causes the services to scale up to nearby servers for rapid response and performance. That requires some unused capacity to be reserved for a single service for future scale up and not to be shared with others.

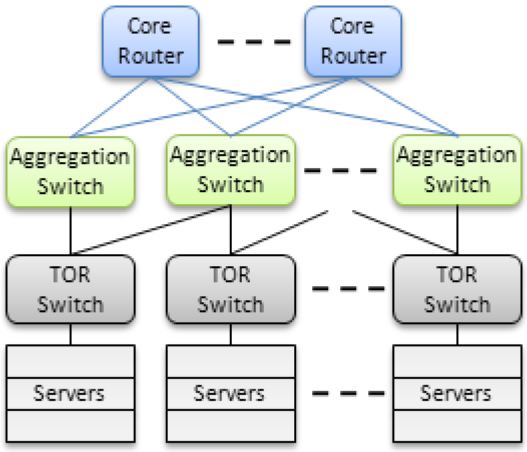


Figure 9 Conventional data centers hierarchy

The diversity of the applications that can be deployed on cloud environments makes it difficult to use real cloud infrastructures such as Amazon EC2 (Amazon EC2 (2012)), Google App Engine (AmazonEC2 (2012)) or Microsoft Azure (Chappell (2008)) for performance, throughput and cost evaluation because each application may have different configuration and deployment requirements. It is extremely hard to reconfigure such real infrastructures during the applications development and over the multiple test runs. Moreover some of the parameters are not even configurable by the developer. This motivates the use of cloud simulation tools that provide a controllable environment for developers to test and tune their application services before deploying them on real clouds.

CloudSim was proposed to overcome the limitations of existing distributed grid and network simulators since none of them offers the environment that can be directly used for modeling CC. CloudSim supports modeling large scale cloud computing environments and data centers along with many novel features such as simulating network connections between the different entities in the system. NetworkCloudSim was developed to simulate the behavior of parallel and distributed applications in cloud computing environments (Garg and Buyya (2011)). NetworkCloudSim offers the basic entities and classes to simulate different network topologies in a cloud environment such as Switch (RootSwitch, AggregateSwitch and EdgeSwitch), NetworkDatacenter and NetworkDatacenterBroker as show in Figure 10.

However, NetworkCloudSim does not support the common network topologies in cloud computing such as VL2 (Greenberg et al. (2009)), BCube (Guo et al. (2009)), Portland (Mysore et al. (2009)), and DCell (Guo et al. (2008)). Hence,

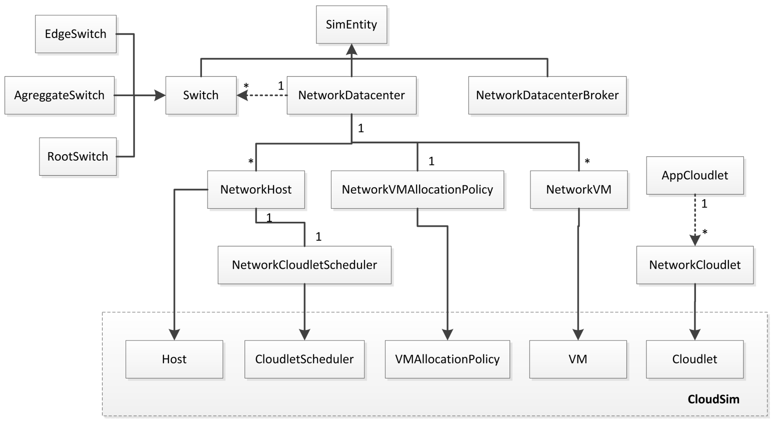


Figure 10 Class diagram of NetworkCloudSim

these topologies were integrated into our TeachCloud tool (Jararweh et al. (2012)). Also, TeachCloud supports a GUI to drag and drop entities, define their properties, and establish the connections between them which results in a customized network topology. In our implementation, each network topology has a structure similar to the one shown in Figure 11, which describes the structure of a VL2 topology. The workload of the simulation which describes the users tasks is converted to a list of NetworkCloudlets and AppCloudlets. Each NetworkCloudlet has information regarding the required bandwidth and the destination resources. The NetworkCloudlet traverses through switches from RootSwitch down to AgreggateSwitches until it reaches the corresponding EdgeSwitch which has appropriate destination resources. Each EdgeSwitch is connected to a number of servers running virtual machines. The VM would be the host for the NetworkCloudlet processing.

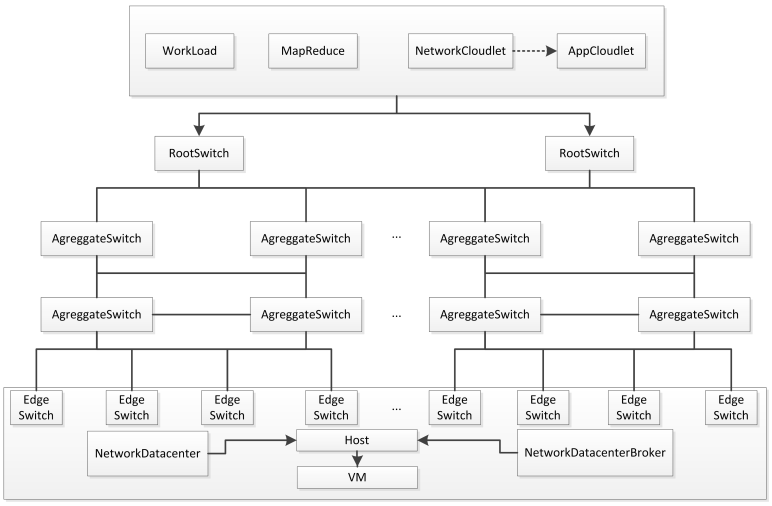


Figure 11 VL2 network structure in TeachCloud simulator

In addition, TeachCloud supports a graphical interface that allows users to build and implement customized network topologies. Figure 12 shows the modules that are supported in TeachCloud GUI. As shown in the figure, there are two methods to build a network topology, the first is to load a previously-saved topology from a file and display it in the Visual Designer Canvas. The second alternative is to use a Palette and Properties tool to design the network into Visual Designer Canvas, the palettes contain different objects that can be dragged and dropped into the Visual Designer Canvas. Each object has a list of attributes (e.g. bandwidth and delay) that can be filled in a properties window. After the network topology is completed, the GUI contains Simulation Control buttons that can be used to start and control the simulation. Once the simulation is finished, the results are displayed as charts of different types.

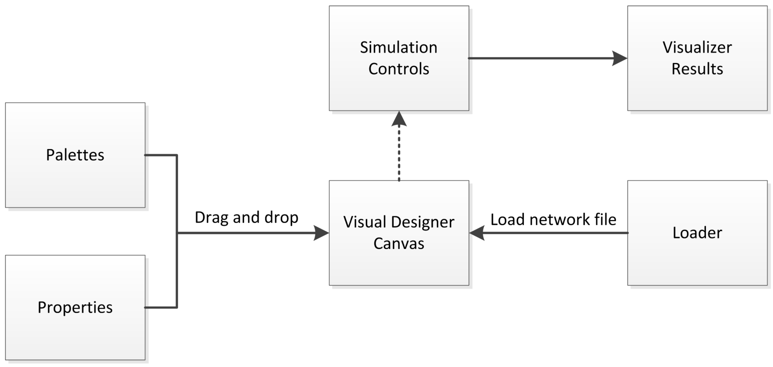


Figure 12 TeachCloud network GUI builder model

7 Simulation and experiments

In order for TeachCloud users to comprehend the different aspects of cloud computing environments, they can simulate experiments by specifying some infrastructure configurations. For example, the following scenario assumes that a user with ID=0 has a cloud environment that consists of two data centers (datacenter_1 and datacenter_2). Each data center has the parameters shown in Table 2. Moreover, each data center has four identical physical nodes (hosts) each with the properties shown in Table 3. Two virtual machines (VM) are executing on each host which results in eight VMs for each data center. The virtual machines share the resources of the host equally. In addition, in this scenario, VL2 network topology is used with the specifications shown in Table 4. The workload that was used to carry out this simulation is a Rain workload. Figure 13 shows the utilization and power consumption results of the simulation produced by TeachCloud.

To demonstrate the capabilities of TeachCloud in simulating different configurations, Figure 14 shows the experimental results produced by TeachCloud for another cloud system configuration. The configuration is identical to the pervious one except that there is only one virtual machine per host, the task

Table 2 Data center configuration

ISA	x86
Operating System	Linux
Virtual Machine Monitor (VMM)	Xen
Time Zone	current time
Cost per Processing Second	0.03 cents
Cost per Memory Unit	0.05 cents
Cost per Storage Unit	0.001 cents
Cost per Bandwidth Unit	0.01 cents
Virtual Machine Allocation Policy	Allocate VM to the host with lowest utilization

Table 3 Host properties

Host ID	A unique ID automatically generated in the range 0-3
Storage Capacity	1 TB
Number of CPUs	2
Cores per CPU	4
MIPS for each core	1024
Memory capacity	2 GB
Bandwidth	10 Mbps
Virtual Machine Scheduler	Space shared

Table 4 VL2 network topology structure

Switching levels	3
Types of switches	One root switch, two aggregate switches, two edge switches
Delays	Root switch = 0.00285 ms, aggregate switch = 0.00245 ms, edge switch = 0.00157 ms
Bandwidth	Root switch = 100 Mb, aggregate switch = 100 Mb, edge switch = 40 Mb

scheduler is time shared, and a customized network topology with one level of switching is used.

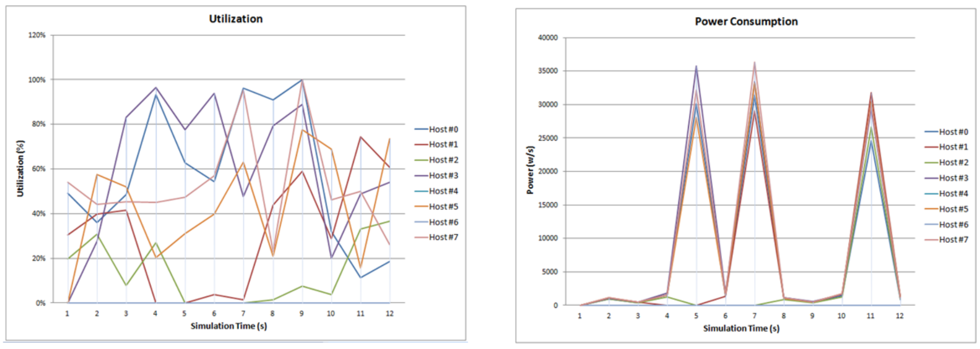


Figure 13 Host utilization and power consumption with VL2 network topology

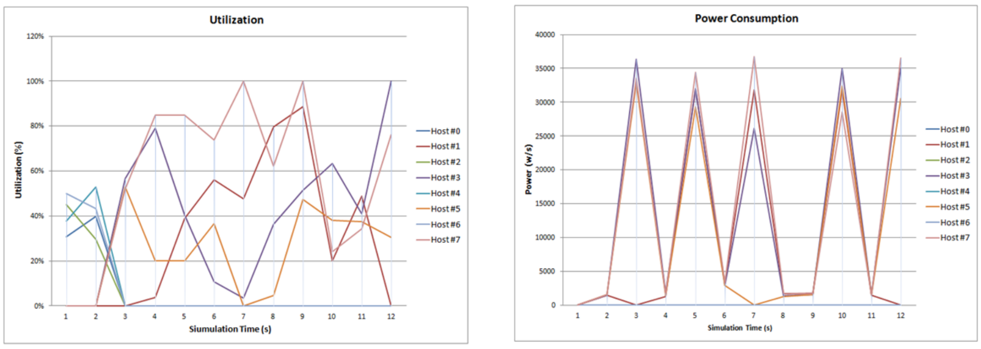


Figure 14 Host utilization and power consumption after configuration modification

8 Conclusion and future work

This paper presented TeachCloud, a comprehensive, easy-to-use, and efficient cloud computing modeling and simulation toolkit. TeachCloud fills a large gap in teaching cloud computing caused by the lack of such a comprehensive and easy-to-use tool, in addition to the high-risks and costs of allowing students to experiment using a real cloud system. TeachCloud provides a rich, yet simple, GUI to build cloud infrastructure and present results and charts. TeachCloud allows a user to customize all aspects in a cloud infrastructure from the host processing nodes to the network topology. In addition, MapReduce model is integerated in TeachCloud to allow the processing of large datasets. It also allows users to integrate the SLA and other business aspects into the tool. Furthermore, it includes an extensive workload generator capable of representing real world cloud applications accurately. Moreover, the modularity in TeachCloud's design allows users to integrate new components or extend existing ones easily and effectively.

TeachCloud makes it easy for users to comprehend the different cloud system components and their roles in the whole system. Users can modify the different components and their parameters, run simulations, and analyze results. As future work, our goal is to formulate practical exercises using TeachCloud which instructors can use as guidelines in the teaching process. TeachCloud will be available as an open source toolkit for academic use along with the practical exercises.

References

Zhang, Qi and Cheng, Lu and Boutaba, Raouf (2010) 'Cloud computing: state-of-the-art and research challenges', *Journal of Internet Services and Applications*, Vol. 1, pp.7–18.

Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D. A., Patel, P. and Sengupta, S. (2009) 'VL2: a scalable and flexible data center network', *SIGCOMM Comput. Commun. Rev.*, Vol. 39, pp.51–62.

Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y. and Lu, S. (2008) 'Dcell: a scalable and fault-tolerant network structure for data centers', *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, Seattle, WA, USA, pp.75–86.

Buyya, R., Ranjan, R. and Calheiros, R.N. (2009) 'Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities', *Proceedings of the international conference on high performance computing and cimulation (HPCS), 2009*, Leipzig, Germany, pp.1–11.

Dean, J. and Ghemawat, S. (2004) 'Mapreduce: simplified data processing on large cluster', *Proceedings of the 6th conference on Symposium on Opearting Systems Design and Implementation, 2004*, USENIX Association, Berkeley, CA, USA.

Garg S. K. and Buyya R. (2011) 'NetworkCloudSim: modelling parallel applications in cloud simulations', *Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing (UCC 2011)*, Melbourne, Australia.

Xiao, Z. and Xiao, Y. (2011) 'Accountable MapReduce in cloud computing', *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2011*, pp.1082–1087.

Jararweh, Y., Alshara, Z., Jarrah, M., Kharbutli, M., and Alsaleh, M.N. (2012) 'TeachCloud: Cloud Computing Educational Toolkit', *Proceedings of the 1st International IBM Cloud Academy Conference, ICA CON 2012, 2012.* North Carolina, USA, April 2012.

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I. and others (2009) 'Above the clouds: A berkeley view of cloud computing', *EECS Department, University of California, Berkeley, Tech. Rep.* UCB/EECS-2009-28, 2009.

Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y. and Lu, S. (2009) 'BCube: a high performance, server-centric network architecture for modular data centers', *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, Barcelona, Spain, pp.63–74.

Faniyi, F. and Bahsoon, R. (2011) 'Engineering Proprioception in SLA Management for Cloud Architectures', *Proceedings of the 9th Working IEEE/IFIP Conference on Software Architecture (WICSA), 2011*, Boulder, CO, USA, pp.336–340.

Kim, K. H., Beloglazov, A. and Buyya, R. (2009) 'Power-aware Provisioning of Cloud Resources for Real-time Services'. *Proceedings of the 7th international workshop on middleware for Grids, Clouds and e-Science.* Urbana Champaign, Illinois, USA, 2009.

Patel, P., Ranabahu, A. and Sheth, A. (2011) 'Service Level Agreement in Cloud Computing', *In Cloud Workshops at OOPSLA09, 2009*

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F. and Buyya, R. (2011) 'CloudSim: a toolkit for modeling and simulation of Cloud computing environments and evaluation of resource provisioning algorithms', *Software: Practice and Experience*, Vol. 41, pp.23–50.

AmazoneEC2 (2012) 'Amazon Elastic Compute Cloud (Amazon EC2)', *http://http://aws.amazon.com/ec2/.*

GoogleApp (2012) 'Google App Engine', *http://appengine.google.com.*

Mysore, R. N., Pamboris, A., Farrington, N., Huang, N., Miri, P., Radhakrishnan, S., Subramanya, V. and Vahdat, A. (2009) 'PortLand: a scalable fault-tolerant layer 2 data center network fabric', *SIGCOMM Comput. Commun. Rev.*, Vol. 39, pp.39–50.

Chappell, D. (2008) 'Introducing the Azure Services Platform', *White Paper, sponsored by Microsoft Corporation.*

Beitch, A., Liu, B., Yung, T., Griffith, R., Fox, A. and Patterson, D. A. (2010) 'Rain: A Workload Generation Toolkit for Cloud Computing Applications', *Technical Report UCB/EECS-2010-14.*

Mann, V., Kumar, A., Dutta, P. and Kalyanaraman, S. (2011) ‘VMFlow: Leveraging VM Mobility to Reduce Network Power Costs in Data Centers’, *Lecture Notes in Computer Science (NETWORKING 2011)*, Vol. 6640, pp.198-211.