Using cloud computing to support higher education in networks and systems

Jianwei Liu, Yunhui Fu and Jim Martin*

School of Computing,
Clemson University,
Clemson, SC, 29634, USA
Email: jianwel@clemson.edu
Email: yfu@clemson.edu
Email: jmarty@clemson.edu
*Corresponding author

Abstract: Virtual machine (VM) technology is widely used in higher education to support pedagogy and research. However, without a standard platform for managing VM technology, VM-based infrastructure is likely to be acquired and deployed in an ad-hoc manner. The virtual computing lab (VCL) system is an open source cloud computing platform developed specifically to support higher education. The main goal of VCL is to make available dedicated, custom compute environments to users. We built a small VCL cloud and trialed its use to support several networking courses offered in the School of Computing at Clemson University during the 2013/2014 academic year. In this paper, we summarise the trial and provide results and conclusions. We found that well provisioned PCs might support up to ten VMs however when subject to peak loads, the system is not able to support accurate and reproducible network experiments.

Keywords: virtual computing lab; VCL; higher education; cloud computing.

Reference to this paper should be made as follows: Liu, J., Fu, Y. and Martin, J. (xxxx) 'Using cloud computing to support higher education in networks and systems', *Int. J. Cloud Computing*, Vol. X, No. Y, pp.xxx–xxx.

Biographical notes: Jianwei Liu is a PhD student in the School of Computing at Clemson Unversity. He received his Bachelors degree of Software Engineering, and Master of Science degree of Telecommunication and Information Systems from Xidian University in 2007 and 2010, respectively. His research interests include cloud computing, cloud-based network resource management, and M2M communications.

Yunhui Fu received his BS in Electrical Engineering from the Central South University at Changsha, China in 2000, and MS in Computer Science from the University of Taxas-Pan American in 2010. Since then, he has been in Clemson University to pursue his PhD. His research interests include improving he reliability, scalability of media communication over wired and wireless packet network.

Jim Martin is an Associate Professor in the School of Computing at Clemson University. His research interests include broadband access, wireless networks, internet protocols, and network performance analysis. His current research projects include heterogeneous wireless systems and DOCSIS 3.x cable access networks. He has received funding from NSF, NASA, the Department of

Justice, Cisco, IBM, CableLabs, and BMW. Dr Martin received his Ph.D. from North Carolina State University. Prior to joining Clemson, he was a consultant for Gartner, and prior to that, a software engineer for IBM.

This paper is a revised and expanded version of a paper entitled 'Using cloud computing to support higher education in networks and systems' presented at Proceedings of 2nd International IBM Cloud Academy Conference (IBM ICA), Atlanta, GA, April 19, 2014.

1 Introduction

A growing number of courses in universities require computer software as a learning tool. As the variety of software to be installed on campus computers grows, software licensing and management of open source software will continue to be a challenging problem. Security and failure-resilience of those systems is more critical than with personal computing, since they are used by potentially large number of users. Traditionally, groups of campus computers are imaged with a particular combination of system and application software. Instructors might not have the permission to change the image. While this approach can achieve the desired level of security and robustness, there are several disadvantages. First, it is not flexible. Instructors may require specific software running on specific operating system at specific period of time. System changes on campus computers even require the information technology department involved. Second, the approach is not efficient. It is common that specific software for courses to be only installed on a small portion of the departmental computers. Installing a large amount of software on all machines can lead to deteriorated performance and potential software conflicts.

Virtual machine (VM) technology is a promising solution to the above issues. Images of VMs can be tailored by instructors based on their unique course requirements. VMs can be isolated from other systems thereby supporting course scenarios where students require root permission. Allowing instructors (or students) to take snapshots of VMs enhances system robustness. However, as the level of capability of VM systems increases, the complexity and cost surrounding creation, storage, and management of VMs can become unmanageable.

VM technology is widely used in higher education to support pedagogy (Xu et al., 2012; Powell et al., 2007; Krishna et al., 2005; Du and Wang, 2008). Here at Clemson University, and likely at other universities, there is no standard mechanism in place to manage VMs. As a consequence, departments or individual faculty are likely to setup VM-based infrastructure to support education on an as-needed basis. Engineering programs teach domain specific problem solving methods using a wide variety of software systems that likely include clustered Linux environments, high performance computing, big data programming environments. These systems typically are based on combinations of licensed and open-source software. As the adoption of software systems in higher education grows, the management of the required infrastructure will continue to become more challenging. Cloud computing is a possible approach to meet the rapidly evolving computing requirements faced by higher education.

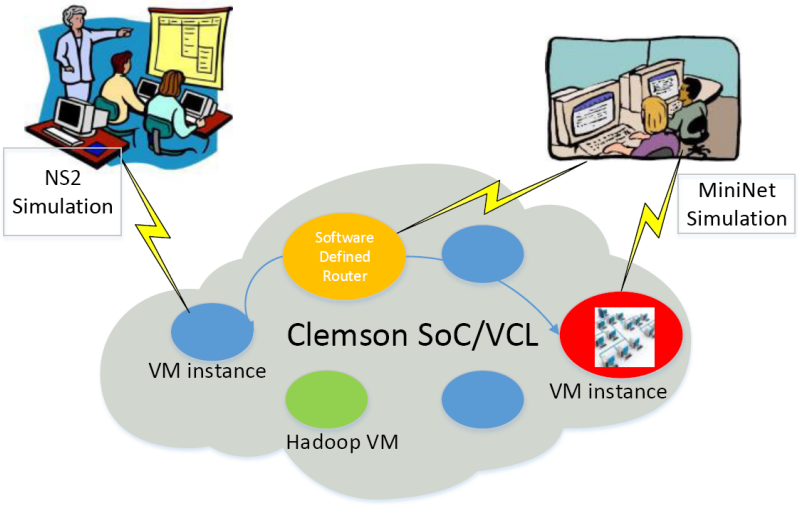
The technology and ideas surrounding cloud computing have evolved significantly over the last several years. At the highest level, cloud computing provides services to users over a network. The nature of the service, the target user community, and the economics that drive the use and adoption of the system all contribute to a diverse range of cloud computing systems. We focus on cloud computing to support higher education focusing on courses that involve networks or systems. These courses tend to be hands-on, lab oriented courses that involve writing and testing software in realistic system environments. Prior to VM technology, systems courses typically required a specific lab containing dedicated lab machines. It is likely that these courses impose the most challenging requirements on a potential cloud-based system that supports higher education.

In this paper, we report on our progress in evaluating cloud-based VM technology to support higher education. We used the virtual computing lab (Schaffer et al., 2009), a cloud computing platform that has been recently contributed by IBM and North Carolina State University as an Apache open source project (Apache VCL:, https://vcl:apache:org/). The goal of VCL is to make available an easily managed, cloud compute environment to users that could include students, faculty or staff in a University environment.

We built a small VCL cloud and trialed its use to support several networking courses in the 2013/2014 academic year. As shown in Figure 1, the cloud in the middle represents the Clemson School of Computing network with our VCL cloud in the core. Inside this cloud, the different ellipses signify various VM images that can be instanced for teaching purposes. Some might serve as software defined routers, others might serve to provide network emulation or simulation environments. The testbed is designed to support a small-scale trial of VCL. It involves two well provisioned compute nodes that were allowed to support up to ten VMs. We did not permit users to request 'bare-metal' machines. While the testbed can be used for other computer science courses, we focused our trial on a single course: an entry level graduate course on computer networking (specifically, CPSC8510). Our test bed currently does not integrate with OpenStack either. However, CCIT at Clemson is evaluating OpenStack to see he chance of overlaying VCL over OpenStack. We hope our trial can serve some useful information for them.

The paper is organised as follows. First, we provide necessary background information. Then we present related test beds and research work in Section 3. Next, we present the details of our VCL cloud in Section 4. In Section 5, we detail the customised images that we developed, following by some case studies of how we use VCL in our networking courses in Section 6. In Section 7, we summarise the lessons we have learned based on the trial. In the final section we identify next steps.

Figure 1 The concept of our cloud-based virtual lab (see online version for colours)



2 Background

In this section, we briefly overview VCL. We compare VCL with other cloud technologies that are available. The main goal of VCL is to make available dedicated, custom compute environments to users. The compute environments can range from something as simple as a VM running productivity software to a cluster of powerful physical servers running complex HPC simulations. North Carolina State University (NCSU) and other schools in North Carolina rely on VCL for managing compute resources for faculty, staff, and students (https://vcl:ncsu:edu/). VCL is used to provide compute platforms to faculty and students since 2004. It is now used by many universities and high schools across the USA and in Europe (Schaffer et al., 2009).

The VCL infrastructure includes a web server, a database, and a management node. Open source tools like Apache, MySQL, and extreme cluster/cloud administration toolkit (xCAT) are used for the above modules respectively. It requires all nodes to have two network interfaces, one that connects to the campus network and another that connects to a private network that is used to transfer images. A typical deployment of VCL would integrate with existing campus computing infrastructure (Peeler and Thompson, 2007).

VMs have been used in industry for many years. Advances in technology, the importance of computing to business, as well as economic factors have results in wide use of cloud computing in enterprise environment. OpenStack (Sefraoui et al., 2012) is an enterprise level VM management framework that is now widely used for data centre VM management. It can manage large number of server resources and VMs, with rich sets of features such as floating IP addresses and storing and managing files programmatically. Comparing with VCL, OpenStack has more features, works with large number of nodes, and supports almost all of the widely used VM formats. However, the amount of work in setting it up may intimidate small or medium sized organisation users away. So, in an

OpenStack enabled campus, the integration of VCL with OpenStack is an interesting direction to explore.

SaltStack (Hosmer, 2013) is a widely used configuration management tool that can largely reduce the time of orchestration for servers (Kavis, 2013). It solves the problem of "what if I want to configure 1,000 VMs to run the same service". Vagrant (Delaney, 2014) is another VM management tool that allows users to script and package the VM configuration and the provisioning setup. VM management tools and systems such as SaltStack and Vagrant focus on scalability whileVCLfocuses on resource management and access control.

The academic research community has developed slicing techniques to support research experimentation. A slice is defined as a set of basic resource allocation units, such as processing or network components, and a set of users that are allowed to access them in a controlled manner. Research projects including PlanetLab (Chun et al., 2003), Emulab (Hibler et al., 2008), and Global Environment for Network Innovations (GENI) (Berman et al., 2014) all involve large scale deployments of sliced-based test beds.

Clemson has a large computing environment referred to as the Palmetto cluster (CCIT, 2014). It currently comprises over 1,977 compute nodes (21,392 cores), plus 440 NVIDIA Tesla GPU accelerators. It ranked #8 among academic research clusters on June 2013 top 500 list, and benchmarked at nearly 400 TFlops using 195 nodes (3,120 Intel Sandy Bridge cores + 390 NVIDIA Tesla K20m accelerators). Users can use portable batch system (PBS) script to request computer resources that include the desired number of cores, the amount of memory and the interconnect type. Palmetto is designed to serve as a general purpose super-computing platform. Compute resources allocated to users are bare-metal machines (although we expect to see native VM support in the future). Therefore, users can not control the images of the computers and are generally not allowed to access them as root users.

The performance/quality of service (QoS) provided by Palmetto is based on a 'best effort' model as it lacks the necessary isolation mechanisms to prevent users from interfering with each other. If a user job consumes significant CPU resource at one node, or has significant memory leakage, the other tasks running on that node will be impacted. For a cloud used for networking teaching or experiment purpose, this difference will likely render different results for the same experimentation.

3 Related work

The use of cloud computing has been studied by the academic community. It has been noted that the flexibility of cloud computing economic models might benefit small and medium sized institutions to utilise cloud-based solutions rather than purchasing physical machines (Sultan, 2011). Stein et al. (2013) further argued that the budget in educational institutions is usually limited, and tried to utilise VCL-based cloud computing to assist pedagogy in K-12 schools. From their experience, VCL can provide computational resources for teaching and learning both cost-effectively and with the flexibility that education requires.

In supporting high education using cloud computing or emulation, Xu et al. (2012), Rollins (2011), Berman et al. (2014), Collicutt (2014) and Huang et al. (2014) have made contribution from various perspective using different approaches. Similar to our work,

Xu et al. (2012) provides a virtual laboratory platform to support networking courses. Based on the XEN hypervisor, the solution has the advantage of fine grained control of the software as it was developed by the authors. However, as stated in the paper, reliability is a crucial issue for a virtual lab system, and it is a problem for their system. This was actually an important design factor we considered when we designed our VCL platform.

Rollins (2011) is a similar paper on utilising latest technology to facilitate higher education. It provides an example on how wireless sensor network (WSN) can be introduced to undergraduate and introductory graduate students. Similar to our paper, the authors describe their application of a specific WSN platform and provide insights on how that approach can help students' learning process.

GENI (Berman et al., 2014) is an NSF sponsored effort to create a large scale test bed for network experimentation. GENI is aimed to provide a federated network supporting repeatable network experiments. It is increasingly used in classrooms as reported in the latest GEC 19 (Thomas, 2014). A wide range of tools are created to use GENI for educational purpose, like LabWiki (Jourjon et al., 2011) and integrated rule-oriented data system (iRODS) (Rajasekar et al., 2010). Users can load default or customised images to the VMs located at campuses across the USA, and use software defined network (SDN) to stitch them together for experimentations. Currently both inner-aggregate and inter-aggregate experiments are supported. Users can easily design the network topology through web-based tools like FLACK and Flukes (Xiong and Pan, 2013).

The authors of Berman et al. (2014) indicate that the motivation of GENI is to confront the challenge of 'internet ossification'. GENI wants to utilise sliceability, deep programmability, and geographical diversity to solve the ossification underlying the whole internet. In our context, the 'ossification' of network and computing resources within a campus is likely due to the lack of centralised network and computing management. Our work is clearly synergistic with GENI, the method of how we handle the 'ossification' problems differ in scope and objectives.

VCL is likely most appropriate for a small to mid-size 'private' cloud system. There are clear benefits to integrating large scale, federated, programmable network like GENI into education. However, concerns related to the required learning curve of GENI, as well as to the security, reliability and flexibility of a public cloud motivated our interest in VCL. Flexibility is of high importance. For example, with GENI, it can be difficult to access a GENI node using a remote desktop interface, such as with virtual network computing (VNC) (Richardson et al., 1998) even with the VNC server running on the image. With VCL, we simply revise the code to enable remote desktop for any VM.

Collicutt (2014) has introduced their experience and lessons learned in overlay VCL on top of OpenStack. Their test usage provides valuable knowledge on deploying VCL in a larger scale. They found several issues, such as the NAT problem, the golden image maintenance problem, etc.

MiniNet (Lantz et al., 2010) is an emulator that supports OpenFlow networks. It can efficiently emulate hundreds of nodes in a single laptop/server. Handigol et al. (2012) identifies the core requirements necessary for a system to produce realistic and reproducible results, including functional, timing and traffic realism. Our work evaluates VCL based on these requirements.

This tutorial (Te-Yuan Huang, 2014) provides a survey on using MiniNet to teach computer networking. Gupta et al. (2013) is another software system that serves as a

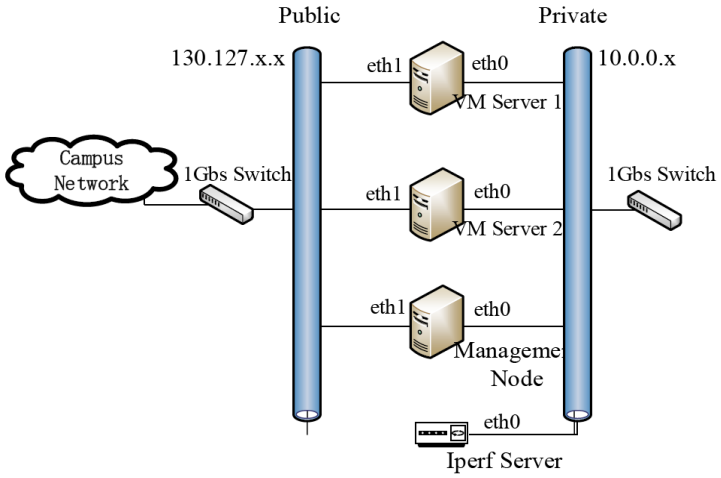
network emulator. They claimed faster emulation time with similar fidelity as MiniNet using event-based emulation.

Roy et al. (2014) introduces another test bed to further scale an OpenFlow-based emulation system. Finally, Antonenko and Smelyanskiy (2013) provides a cluster version of MiniNet that can scale MiniNet to emulate very large networks.

4 Deployment of VCL platform

We deployed the network with the topology showed in Figure 2. VM Server 1 and VM Server 2 are two bare-metal machines which serve as the VM host and support the VMs. Each machine is a Dell Optiplex 9010 which has 16 GB memory and an i7-3770 quadcore CPU. Due to the hardware limitation and homework need, we limited the resource of each VM to 500 MB memory. We provisioned the system to allow at most ten VMs to run concurrently on each physical machine. To make the system more failure-resilient, we deployed the VCL management node on a dedicated VM which ran on a third physical machine. Another machine was set up as an image back-up repository. This allowed easy system recovery and backups. All the VCL related modules including the MySQL database run on that machine. We also set up network file system (NFS) among those machines to share the VM images. Without NFS, we observed errors when importing updated images. The cleanest way to update all images quickly is to simply have VM servers use symbolic links to the images. This forces VCL to store images locally.

Figure 2 Topology of our VCL deployment (see online version for colours)



In some situations, we wanted the students to access the VMs using virtual network computing. This required a minor modification to the VCL source code. The problem occurs because there are two passwords involved. The first is the authentication required by ssh to access the image. The second is the password for VNC to connect to the image. We changed the VCL source code to unify the two passwords.

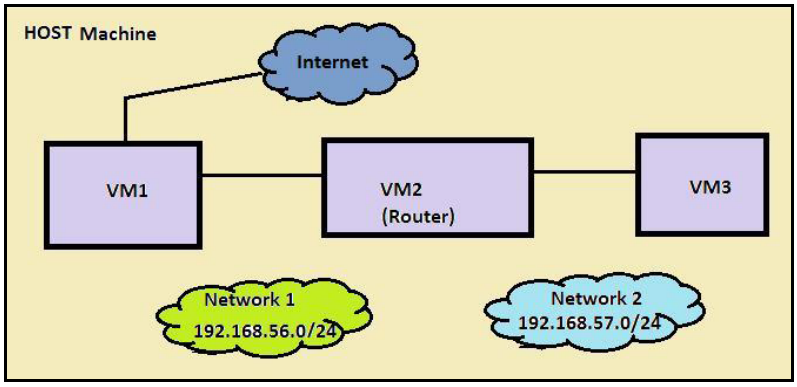
Due to the limited scale of the test bed, the students were placed into one of ten groups to cooperatively work on projects. Each group could reserve one VM at a time which limits the system to run at most ten VMs at the same time. To add flexibility, we allowed ten VMs per machine. VCL is responsible to the load balancing as it will select a machine that has more resource available and run an image in a VM on that machine.

5 Customised images for network courses

Based on the needs of different courses we designed a number of customised VM images. First, we created a base image using Cent OS 6.4. Then we made different variants of that system with features such as ns-2 (McCanne et al., 1997), MiniNet, and VNC support. We generated kernel-based virtual machine (KVM) `qcow2` images of those Operating Systems (OSs), and imported, or 'image captured' in VCL parlance, the images to VCL by the executable program `vcld`. While we have used the system for several classes, the majority of the following discussion is based on our experiences with an entry level graduate course in computer networking (referred to as CPSC8510). The course learning objectives included:

- basic understanding of computer networking principles
- basic understanding of TCP/IP
- able to develop client/server applications based on Unix sockets
- basic understanding of simulation and how to apply simulation analysis to network systems.

Figure 3 Topology for case Study I and III (see online version for colours)



Network experiment scenario: Each student group is given access to a simple three node system. This is illustrated in Figure 3. The students will learn basic network administration and networking skills. The group's account on the three virtual nodes will allow sudo access to basic administration commands that are necessary to setup the network environment and to monitor system configuration and performance. The students will develop or utilise existing open source network performance and security software on the test bed.

ns-2 simulation scenario: The open source ns-2 simulator is the de facto simulation tool used by the internet research community. It is a discrete-event simulator that faithfully reproduces a wide set of networking protocols and applications. The simulator used in the class will be patched with several modules for teaching purposes.

MiniNet scenario: As described in Lantz et al. (2010), MiniNet-HiFi is a container-based emulation tool that has been introduced in an effort to reproduce network experiments running real code on an emulated network using VM technology. Building on the knowledge obtained from the previous scenarios, we use MiniNet to emulate networks with and without SDN.

5.1 Case Study I

We designed the following lab to help students understand basic networking concepts and measurement tools. We planned to give every student three VMs. They should use two of them as a source and a destination of a networking test, and the third one can be configured as a software defined router. We list one of our assignments below as an example.

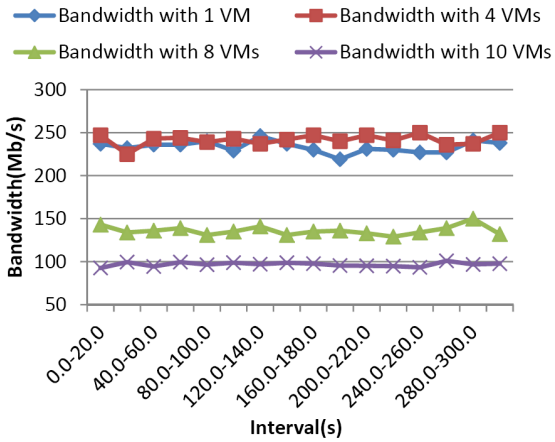
> Today is 6:30 am Monday morning, and Dunken Hines who is the Teaching Assistant assigned to proctor and grade an exam for the instructor, Dr. Rhinestone, while he is still away at a Computer Security Conference. The exam is a computer-based exam scheduled at 9:00 am, to be administered in Dr. Rhinestone's Lab 2 (one of his two labs). The server which is hosting the exam is situated in Lab 1. However, both Lab 1 and Lab 2 are on separate sub network. Lab 1 and Lab 2 are connected via a router.
>
> Hours before the exam, the server crashes. While Mr. Hines is attempting to re-install the server, the router connecting the labs also crashes. Now, Mr. Hines frantically calls Dr. Rhinestone and apprises him of the situation. Dr. Rhinestone suggests Hines to call the IT department for assistance. The IT department sends a network administrator (which is you!) to fix the router.
>
> You diagnose that the router has crashed as a result of a hardware problem.
>
> Since, a replacement for the router cannot be obtained in time, you suggest that a Linux-based host could be configured as a replacement.
>
> We model a simplified scenario of this very real use case in our virtual environment. For this assignment, you need to configure a private network like the one shows in Figure 3. The network should meet the following requirements:
>
> - VM1 should be the only VM that can access the internet
> - VM2 and VM3 should not access the internet.
>
> Once you have the Linux router configured, please run iperf between the VM's to assess the available bandwidth between the VMs.

We planned to use one VM as a software-defined router. However, in practice, we found the limitation of VCL in network topology prevented us from doing so. Thus, we changed our experiment to use two VMs connected directly to finish the desired networking experiments like `iperf`. To answer the question of whether VCL can provide consistent/repeatable result, we did the following experiments to verify the property of VCL in isolating the networking resource usage. We first started one VM and created a

high speed `iperf` from that VM to our shared server VM (which is not on the two VCL VM servers). Then, we repeat but have four, eight and then ten different student VMs running the `iperf` to another VM running on a remote machine.

We plot the evolutions of the four scenario's throughputs in Figure 4. As we can see, the two experiments with one and four VMs have very similar results. However, as we further increase the amount of VMs, we can see that the bandwidth tested from `iperf` decreased, which is very different from the underload situations. While our experiments suggest the system provides network labs to produce repeatable experiences, if the system is loaded differently we will get different results.

Figure 4 Comparison of bandwidth test under different VM loads (see online version for colours)



5.2 Case Study II

The network simulator is a widely used simulation tool designed for network experimentation. We use the tool in several of our networking courses as it can help students understand networking concepts presented in class. We identify the following sample ns-2 assignment.

> You are to write a procedure that use Chapter VIII of the ns tutorial (http://www:isi:edu/nsnam/ns/tutorial/nsscript4:html#second) which records the throughput of the TCP flow and the throughput of the UDP flow, starting a time 0.0 seconds, until time 5.0 seconds, with a sliding window of size 0.5 seconds. Using xgraph, plot the throughput obtained by both flows on the same graph. Submit your script and the plot. Important Remark: We modified the simulator so that Agent/TCPSink has a 'bytes_' variable that can be read by TCL scripts to determine the amount of data received, in a fashion similar to Agent/LossMonitor.

As an open source software, ns-2 has many customised versions and extensions. However, some extensions can only work with certain versions of ns-2. At the same time, certain ns-2 version is better to be compiled under certain versions of operating systems to avoid compilation errors. Before, we just prepare one version of ns-2 in some VM at our lab. When another one is needed, we create another VM, and build an image using another operating system or compilation environment. We believe this situation is true

for large number of modern software for teaching. One possible solution is to prepare different versions of Linux for various ns-2 versions, and then let students download the correct image as a starting point. However, this solution still lacks the cloud nature, and therefore suffers from scalability. The load for large numbers of students downloading large VM image files will be a challenge to servers in our lab. Also, students sometimes complain about problems they met running the downloaded image in their local environment (e.g., network settings). With VCL, we find it is natural to manage different versions of ns-2 different Linux images, and let the student reserve the image they need. We no longer need to create VMs on an as-needed basis, which increases the reusability of the images. Students are also happier with this, since they can focus more on the homework assignment itself, rather than the environment needed by the assignment.

5.3 Case Study III

We designed the following homework assignment for the students to get familiar with MiniNet.

> You should set up a network like the Case Study I, but only make VM1 a real VM. The other nodes should be virtual nodes in a MiniNet VM. First, connect the network properly and redo the 'iperf' experiment in Case Study I. Second, configure a DHCP sever on the VM2, and set up an OpenFlow controller in the MiniNet VM. Set up the controller properly so that only VM1 can receive the DHCP broadcasting from VM2.

This homework assignment design helps us to circumvent the limitation of VCL we mentioned in Case Study I.

6 Evaluation and results

We conducted more experiments to explore the characteristics of VCL in supporting the three scenarios above.

For the first scenario, we already did the experiment in Figure 4. However, we were not sure of why the system behaves like that, or where the bottleneck of the system is. Thus, we further conduct the following experiment.

We repeat the procedure as described for Figure 4. Briefly speaking, we gradually increased the number of VMs, and made every VM instance perform an `iperf` to another VMserver. The numbers of VMs we tried are {1, 4, 8, 10}.We also kept track of which sever every VM instance is assigned to for better understanding the results, as well as the loadbalancing characteristics of VCL. Also, we recorded the CPU and network card statistics while running the experiments using the tool `collectl` (http://sourceforge:net/projects/collectl/).

Table 1 shows the VM assignment for the ten VM scenarios. VM Server 2 is allocated more VMs than VM Server 1. One possible reason is that the numbers of VMs linked to VM Server 1 and VM Server 2 are different in our system.

We collected and show the average CPU utilisation rates of both VM servers, and number of cores used.

Table 1 VM instance assignment to VM servers

VM#	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
# of VM in VM Server 1 (Iperf)	0	0	0	0	1	2	2	3	3	3
# of VM in VM Server 2 (Iperf)	1	2	3	4	4	4	5	5	6	7
# of VM in VM Server 1 (ns-2)	0	0	1	1	1	2	2	3	3	3
# of VM in VM Server 2 (ns-2)	1	2	2	3	4	4	4	5	6	7
# of VM in VM Server 1 (MiniNet)	0	0	1	2	2	2	2	2	2	2
# of VM in VM Server 2 (MiniNet)	1	2	2	2	3	4	5	6	7	8

The bandwidth test using `iperf` in the VM shows a throughput of only 250 Mbits/s. This is true even with only one VM running the test. We further look into the statistics. Figure 5 shows the average CPU utilisation rate ofVMServer 2. Figure 6 shows the average CPU utilisation rate on different cores with the standard deviations as error bars. Sever 1 has similar behaviour when given similar load. It shows that the CPU utilisation rate increases to a maximum of 60% on two cores for a single run. Further, the average CPU utilisation rate is still below 70% even with ten VMs running. This suggests that the network is the bottleneck rather than the CPU. We increased the Linux buffer size for sending and receiving packets. But the results remained the same. We ran `iperf` from the test from the physical server that the VM resides on to the same destination and observed a throughput of 930 Mbits/s. We conjecture that the VM interface has limitation on the bandwidth. We further found it is because quick emulator (QEMU) defaults to the RTL8139 network interface card (NIC) model, which has limitation on bandwidth (Novich, 2013). We tested with another model named virtio, and we could reach around 930 Mbits/s even in the VM. However, we believe this default bandwidth limitation is useful in our scenario to provision network performance isolation, and we continued our experiments with this setting we used during our trial.

Figure 5 The average CPU utilisation rate of Sever 2 (see online version for colours)

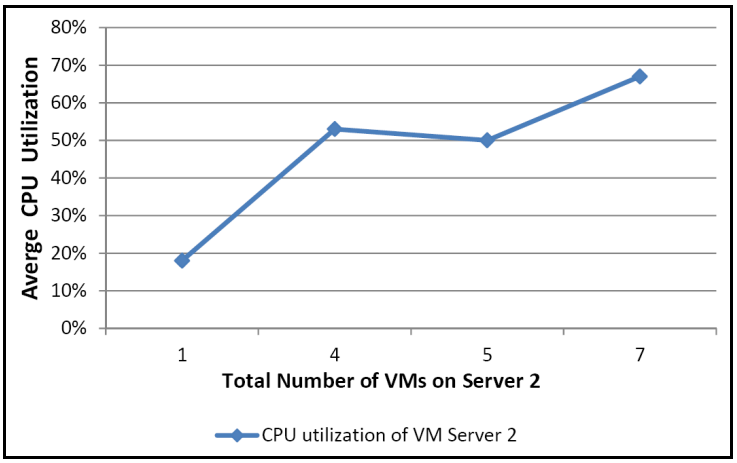
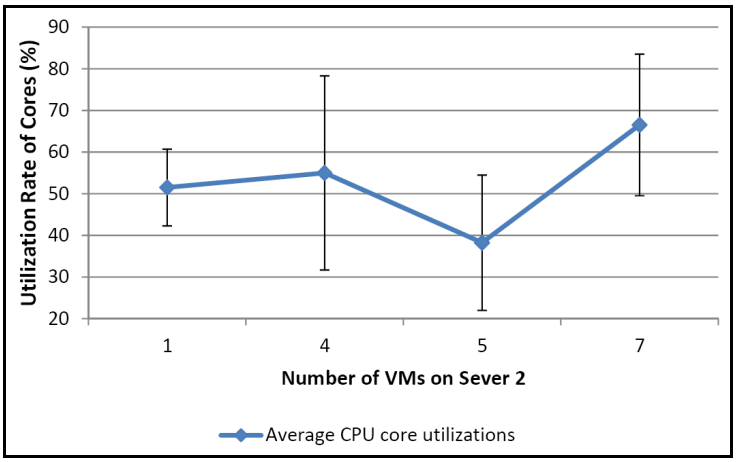
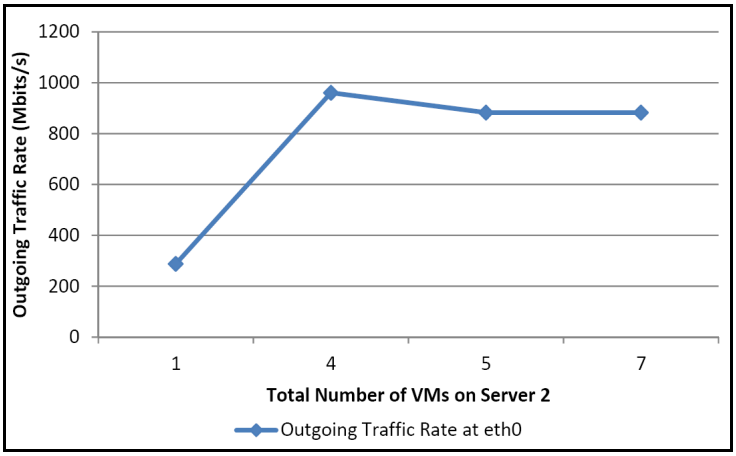


Figure 6 Average utilisation rate on different cores (see online version for colours)



In Figure 4, we see that the bandwidth results of one and four VMs are very close, since a setting of four VMs is still within the limit of the network card and switch capacity. However, with eight VMs, the bottleneck is no longer the virtual interfaces, but the actual interface on the host machine. Figure 7 clearly shows that the network reaches its capacity when four VMs are running on the same VM server. If each consumes 250 Mbps of bandwidth, the aggregate fully utilises the physical network.

Figure 7 Outgoing traffic rate of VM Server 2 (see online version for colours)



Another finding of our experiments is that, to reduce the overhead of recreating new VM, VCL tends to perform lazy-shutdown of unused VM instances (the instance will reboot into a clean copy, but the VM is always there). If the other user requested the same VM, VCL would just clean it up and provide the VM as a refurbished one. This is good for the loading time user experience. However, it will make the reasoning of the VCL load balancing scheme difficult. Also, it prevents us from analysing the memory usage of both

VM servers in different scenarios, though this is already a hard problem due to Linux's memory caching.

Figure 8 CPU utilisation rates of both VM servers running ns-2 (see online version for colours)

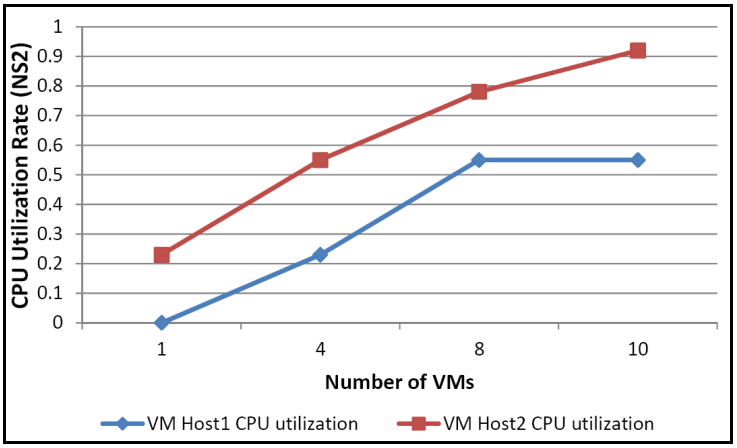


Figure 8 shows the CPU utilisation rates of both VM servers when running {1, 4, 8, 10} ns-2 simulations. Every simulation is a wireless simulation with five nodes running for 2,500 s. Figure 9 shows the simulation time needed for a wireless simulation. The baseline running time is 32 s as we can see from running the first result of VM Server 2. The error bars in the figures are based on the standard deviation of the results from different VM instances. As we can see in Figure 9, the running time increases as the number of VMs grows, and the standard deviation from different VMs also increases. The CPU utilisation rates show why they increase. We can see that for both VM servers, CPU utilisation rates are increasing. VM Sever 2 increases faster because more VMs are assigned to it. For VM Server 2, it had taken almost the capacity of the whole CPU when ten VMs were running the simulations simultaneously. The increase of standard deviation means higher loads on different cores tends to make the running time less predictable. Figure 10 shows that CPU utilisation rates of both VM servers increase as the number of VM instances on it increase. The CPU utilisation rates of the two VM servers are very close when there were four VM instances, since the loads are evenly distributed then as we see from Table 1. The result of this experiment and the hardware specification we used as mentioned in Section 4 can also serve as a guidance to estimation of hardware needs for certain computation power needed by VMs.

We tried to solve the problem we met in case study I using MiniNet. MiniNet provides a set of predefined scripts that build commonly studied topologies such as linear and trees. Using the linear topology from study 1, we performed experiments involving 1, 4, and 8 VM instances. Specifically we issue the MiniNet command 'sudo mn –switch ovsk –test iperf –topo linear, 12'. This command creates a linear topology with 12 nodes linking in a line. Iperf `iperf` is used to test the end-to-end throughput of the network. Figure 11 plots the mean and standard deviation of the iperf test result for each experiment. We can see that the bandwidth tested decreases as the number of VMs increases. This shows that the isolation among different VMs in VCL is not perfect. The results running at different time with different sever loads may vary. This can also

provide us an idea on how much we can trust the results of high-CPU-cost applications like the MiniNet simulation. Also, the increase of standard deviation tells us that VCL does not guarantee equal computational power to every VM, and the difference increases as the number of VMs goes up.

Figure 9 Simulation time of ns-2 programs on both VM servers (see online version for colours)

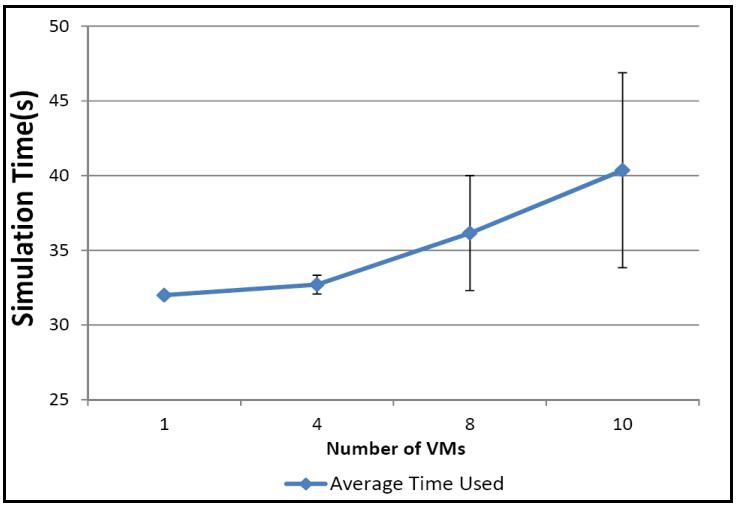


Figure 10 CPU utilisation rates of both VM servers running iperf test inside MiniNet (see online version for colours)

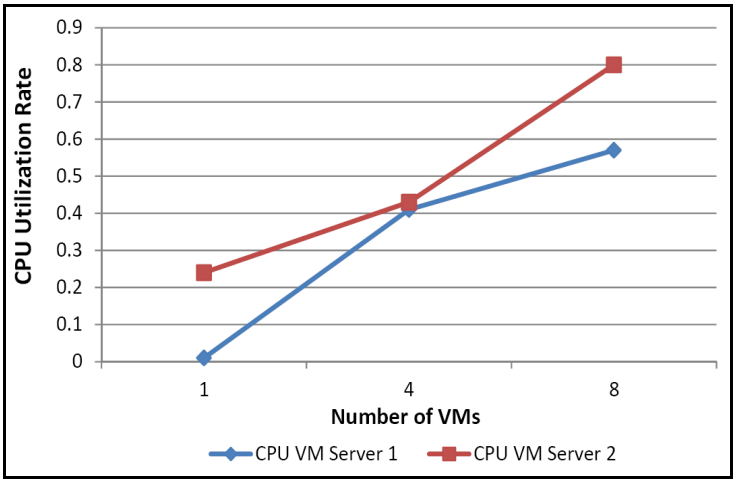
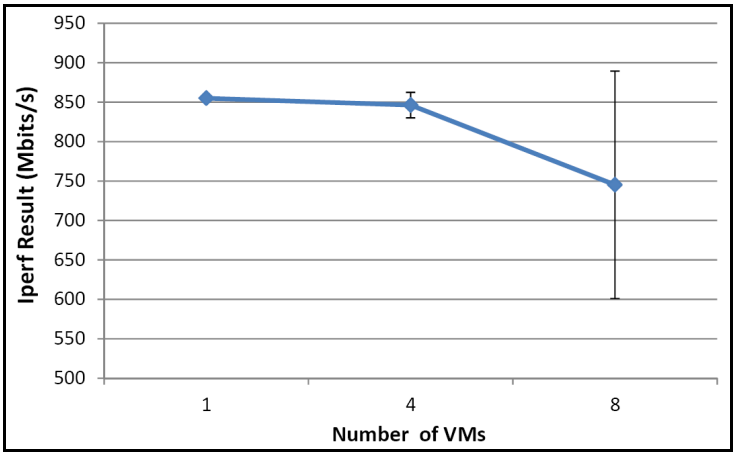


Figure 11 The bandwidth result tested from the iperf test inside MiniNet (see online version for colours)



We also did another test in MiniNet using the command 'sudo mn –switch ovsk –controller ref –topo tree,depth=2,fanout=9 –test pingall'. This starts a tree topology of depth 2 and degree of 9, and run ping between each pair of nodes. It i's interesting to see that with only one VM running this test, there was no dropping of ping packets. But, with more VMs running this test simultaneously, we noticed significant packet drops. Figure 12 shows the drop rates with the standard deviation drawn as error bars.We think the main reason is that the ping test has timeout setting. If the CPU utilisation rate increases, the delay introduced may time out the ping packets. As we can see in Figure 13, both VM servers' CPU utilisation rate increase as the number VMs goes up. Figure 14 tells us the running time of the ping tests as we added more VM instances running this test.

Figure 12 Droprate of the pingall test inside MiniNet (see online version for colours)

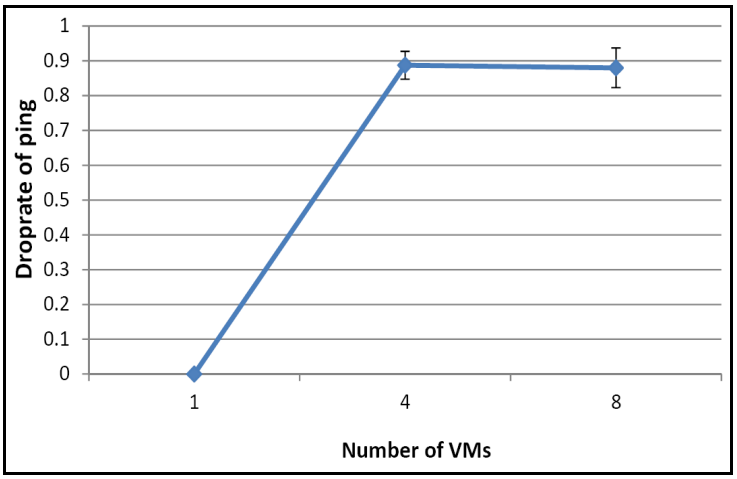


Figure 13 CPU utilisation rates of both VM servers running pingall test inside MiniNet (see online version for colours)

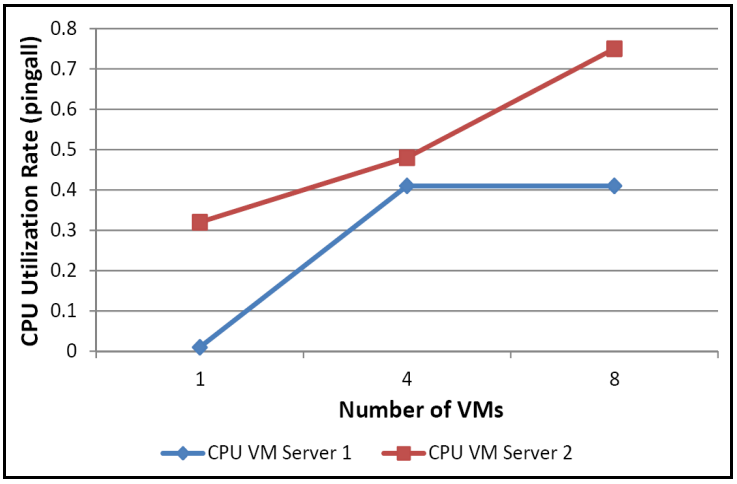
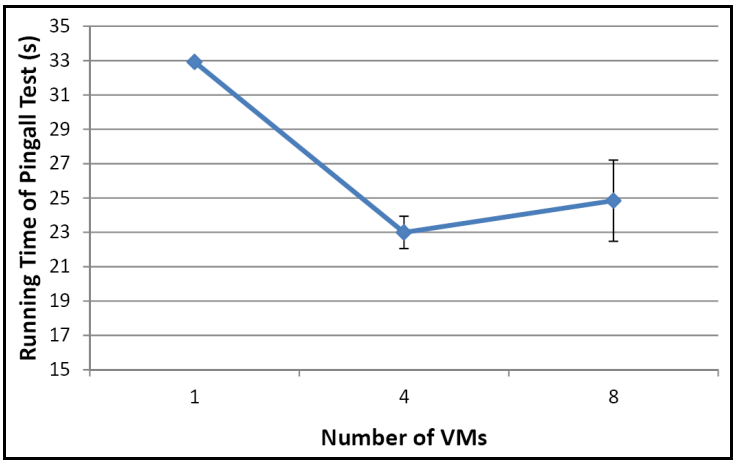


Figure 14 Running time of pingall test inside MiniNet (see online version for colours)



We used the statistics collection feature in the VCL web interface to keep track of the usage of our system. And, at the end the 2013 Fall semester, we collected the following usage statistics and figures. As shown by the statistics, we had 171 reservations and 24924 hours reserved in total. We can see the number of reservations to different images in Figure 15.

Figure 15 The usage statistics of different images provide in CPSC8510 (see online version for colours)

	Reservations	Unique Users	Hours Used	< 2 min load time	>= 2 min load time	Failures
CentOS 6.4 i386 Base:	39	9	6194	34	2	2 (5%)
CentOS 6.4 i386 Base_NS2:	29	8	8050	25	1	1 (3%)
CentOS 6.4 i386 Gnome and VNC:	54	9	4979	45	6	1 (1%)
CentOS 6.4 x86_64 Base:	31	3	3245	30	0	9 (29%)
CentOS 6.4 x86_64 Gnome and VNC:	27	4	2453	23	3	3 (11%)

As we can see, the 'Cent OS 6.4 i386 Base' image and the 'Cent OS 6.4 i386 Base_NS2' images are used the most in hours. This is true because most of our homework depends only on those basic images. We can also see that most of the images can be loaded in less than two minutes, which indicates that the VNC support we added is useful/required in some labs. We can also see some failures of those images. That is because we had short period of server problem in the middle of term, which prevented the student from reserving. We hope to ameliorate this problem by updating our hardware in cooperation with CCIT at Clemson.

Figure 16 displays the usage of our VCL virtual lab over time. The y-axis is the number of active reservations, while the x-axis is time. From Figure 16, we can see that the reservation is concentrated in some days which we think is consistent with the timing of homework assignments. Figure 17 shows the average hourly usage of our VCL VMs during the semester. As we can see, most of the activities are between 10:00 am and 8:00 pm.

Figure 16 VM usage over the 2013 fall semester (see online version for colours)

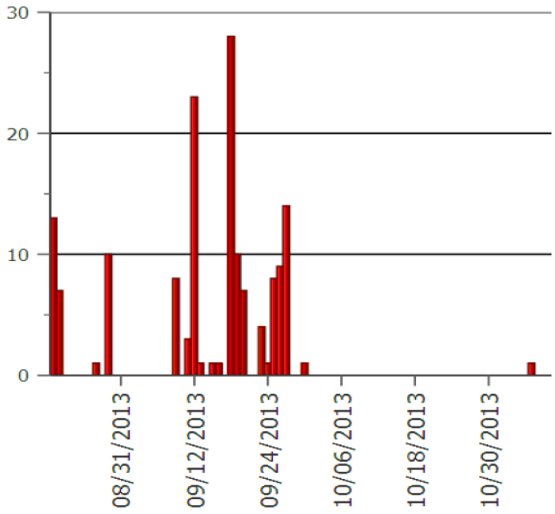
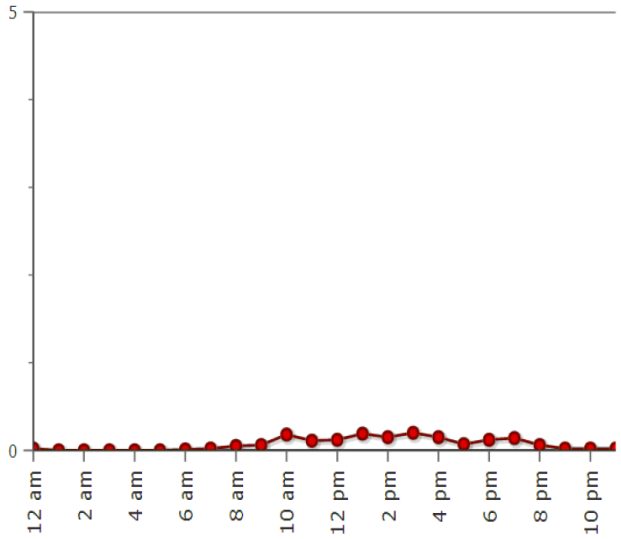


Figure 17 Hourly usage of our VCL virtual lab (see online version for colours)



7 Lessons learned

In Table 2, we compare VCL with other VM management tools. From the table, we can see that KVM manager does not provide basic capabilities such as user/user group management and root permission to students. Public cloud such as PlanetLab and GENI utilises slicing techniques to provide similar capabilities to VCL. However, use of public research cloud presents reliability, security and flexibility issues.

Table 2 Comparison of different approaches for virtual labs

Requirements	*PlanetLab/GENI*	*KVM/KVM manager*	*OpenStack/ SaltStack*	*VCL*
Support 10,000 images	Yes	Yes	Yes	No
Allow faculty to create/save image	Yes	Partially	Yes	Yes
Allow student to create/save image	Yes	No	Possible	Possible
Store data persistently	Possible	Yes	Possible	Possible
Root permission to students	Yes	No	Yes	Yes
User/user group management	Yes	No	Yes	Yes
Software defined networking	Yes	No	Yes	No
Easy for instructors to deploy	Yes	No	No	Yes

We summarise the lessons we learned from our trial:

1 In our tests, we found that the system might be CPU bound or it might be network bound depending on the workload. Our system was designed to limit the maximum number of active VMs to ten or fewer. At this peak load, the system could not maintain realistic and reproducible experiments. However, for our network classes, which tend to be small, VCL is extremely helpful as it allows us to build (just once) any number of specialised Linux distributions that are packages and customised for the specific course. The system provides the instructor basic tools so resources are fairly shared across the student population.

2 The students overwhelmingly found the hands-on exercises helpful. The next semester, we will modify the image used to the nature of our CPSC8510 course requires the system to accurately emulate. In the future we plan on leveraging the testbed in a larger scale.

3 The nature of graduate level networking courses requires a system (live, emulated, or simulated) to provide results that are accurate and that are reproducible. We found live experiments using VMs to have different outcomes depending on work loads.We found that a 'light weight' emulator such as MiniNet also suffers from performance issues as the number of active VMs on the system exceeds 4. Finally, we found that ns-2 does address the limitations in doing live or emulated functions to scale. This is primarily due to the separation of 'simulation time' from real time.

4 The cost of deploying and operating a VM system such as VCL is quite high. Since peak usage induced by courses might be low, a larger deployment involving many courses would amortise costs.

8 Conclusions

In this paper, we explored the use of an open source cloud-based VM management system (VCL) for use in computer network courses at Clemson University. We conclude that a properly provisioned VCL deployment can meet at least some of the needs. Since VCL was not originally designed to support virtual networks, we found an alternative to be to run an emulator (we used MiniNet) as a VM. Also, we did find that at low workloads, the system faithfully reproduces network activities. However, on the positive side, VCL is light weight which makes it useful for small deployments such as on a department level.

In future work, we plan to develop a set of VM image that provides a Hadoop data analysis system. Hadoop (Shvachko et al., 2010) is a distributed computing framework for implementing the Google MapReduce algorithm in a scalable fashion. It enables users to store and process big data and analyse it in ways not previously possible with standard SQL-based approaches. To create realistic scenarios, we will develop a Hadoop test bed that provides aspects of very large scale problem scenarios. We would integrate this into a big data course offered at Clemson University.

References

Antonenko, V. and Smelyanskiy, R. (2013) 'Global network modelling based on mininet approach', in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13*, pp.145–146, ACM, New York, NY, USA.

Apache VCL [online] https://vcl:apache.org/.

Berman, M., Chase, J.S., Landweber, L., Nakao, A., Ott, M., Raychaudhuri, D., Ricci, R. and Seskar, I. (2014) 'GENI: a federated testbed for innovative network experiments', *Computer Networks*, Vol. 61, Special Issue on Future Internet Testbeds, C Part I, pp.5–23.

Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M. and Bowman, M. (2003) 'Planetlab: an overlay testbed for broad-coverage services', *ACM SIGCOMM Computer Communication Review*, Vol. 33, No. 3, pp.3–12.

Clemson Computing and Information Technology (CCIT) (2014) *Overview of Palmetto*.

Collectl [online] http://sourceforge:net/projects/collectl/.

Collicutt, C.M. (2014) 'Using apache VCL and OpenStack to provide a virtual computing lab', *IBM ICA*.

Delaney, R. (2014) 'Vagrant', *Linux J.*, Vol. 244.

Du, W. and Wang, R. (2008) 'Seed: a suite of instructional laboratories for computer security education', *Journal on Educational Resources in Computing (JERIC)*, Vol. 8, No. 1, pp.3.

Gupta, M., Sommers, J. and Barford, P. (2013) 'Fast, accurate simulation for SDN prototyping', in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13*, pp.31–36, ACM, New York, NY, USA.

Handigol, N., Heller, B., Jeyakumar, V., Lantz, B. and McKeown, N. (2012) 'Reproducible network experiments using container-based emulation', in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, pp.253–264, ACM.

Hibler, M., Ricci, R., Stoller, L., Duerig, J., Guruprasad, S., Stack, T., Webb, K. and Lepreau, J. (2008) 'Large-scale virtualization in the emulab network testbed', in *USENIX Annual Technical Conference*, pp.113–128.

Hosmer, B. (2013) 'Using salt stack and vagrant for drupal development', *Linux J.*, No. 227.

Huang, T-Y., Jeyakumar, V. and Lantz, B. (2014) 'Teaching computer networking with mininet' [online] https://github:com/mininet/mininet/wiki/SIGCOMM-2014-Tutorial-Teaching-Computer-Networking-with-Mininet (accessed 15 April 2015).

Jourjon, G., Rakotoarivelo, T., Dwertmann, C. and Ott, M. (2011) 'International conference on computational science (ICCS) labwiki: an executable paper platform for experiment-based research', *Procedia Computer Science*, Vol. 4, pp.697–706, *Proceedings of the International Conference on Computational Science (ICCS)*.

Kavis, M. (2013) 'Saltstack is gaining momentum', *The Virtualization Practice* [online] http://www.virtualizationpractice.com/saltstack-is-gaining-momentum-22170/ (accessed 15 April 2015).

Krishna, K., Sun, W., Rana, P., Li, T. and Sekar, R. (2005) 'V-netlab: a cost-effective platform to support course projects in computer security', in *Proceedings of 9th Colloquium for Information Systems Security Education.*

Lantz, B., Heller, B. and McKeown, N. (2010) 'A network in a laptop: rapid prototyping for software-defined networks', in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, p.19, ACM.

McCanne, S., Floyd, S., Fall, K., Varadhan, K. et al. (1997) *Network Simulator ns-2* [online] [online] http://www-nrg.ee.lbl.gov/ns (accessed 15 March 2015).

Novich, L. (2013) *Red Hat Enterprise Linux 6 Virtualization Administration Guide* [online]. https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization_Administration_Guide/sect-Virtualization-Troubleshooting-KVM_networking_performance.html (accessed 15 March 2015).

Peeler, A. and Thompson, J. (2007) 'VCL tutorial', *IBM University Days 2007*, D.M.V.

Powell, V.J., Davis, C.T., Johnson, R.S., Wu, P.Y., Turchek, J.C. and Parker, I.W. (2007) 'Vlabnet: the integrated design of hands-on learning in information security and networking', in *Proceedings of the 4th annual conference on Information security curriculum development*, p.9, ACM.

Rajasekar, A., Moore, R., Hou, C-y., Lee, C.A., Marciano, R., de Torcy, A., Wan, M., Schroeder, W., Chen, S-Y., Gilbert, L. et al. (2010) 'Irods primer: integrated rule-oriented data system', *Synthesis Lectures on Information Concepts, Retrieval, and Services*, Vol. 2, No. 1, pp.1–143.

Richardson, T., Stafford-Fraser, Q., Wood, K. and Hopper, A. (1998) 'Virtual network computing', *Internet Computing, IEEE*, Vol. 2, No. 1, pp.33–38.

Rollins, S. (2011) 'Introducing networking and distributed systems concepts in an undergraduate-accessible wireless sensor networks course', in *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, pp.405–410, ACM.

Roy, A.R., Bari, M.F., Zhani, M.F., Ahmed, R. and Boutaba, R. (2014) 'Dot: distributed openflow testbed', in *Proceedings of the ACM Conference on SIGCOMM, SIGCOMM '14*, pp.367–368, ACM, New York, NY, USA.

Schaffer, H., Averitt, S., Hoit, M., Peeler, A., Sills, E. and Vouk, M. (2009) 'NCSU's virtual computing lab: a cloud computing solution', *Computer*, Vol. 42, No. 7, pp.94–97.

Sefraoui, O., Aissaoui, M. and Eleuldj, M. (2012) 'OpenStack: toward an open-source solution for cloud computing', *International Journal of Computer Applications*, Vol. 55, No. 3, pp.38–42.

Shvachko, K., Kuang, H., Radia, S. and Chansler, R. (2010) 'The hadoop distributed file system', in *Mass Storage Systems and Technologies (MSST), IEEE 26th Symposium on*, pp.1–10, IEEE.

Stein, S., Ware, J., Laboy, J. and Schaffer, H.E. (2013) 'Improving K-12 pedagogy via a cloud designed for education', *International Journal of Information Management*, Vol. 33, No. 1, pp.235–241.

Sultan, N.A. (2011) 'Reaching for the "cloud": how SMEs can manage', *Int. J. Inf. Manag.*, Vol. 31, No. 3, pp.272–278.

Thomas, N.R.V. (2014) 'GENI In classroom', *GEC 19*.

Xiong, K. and Pan, Y. (2013) 'Understanding protogeni in networking courses for research and education', in *Research and Educational Experiment Workshop (GREE), 2nd GENI*, pp.119–123, IEEE.

Xu, L., Huang, D. and Tsai, W-T. (2012) 'V-lab: a cloud-based virtual laboratory platform for hands-on networking courses', in *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE '12*, pp.256–261, ACM, New York, NY, USA.