Experiences from Virtual Desktop Cloud Experiments in GENI

Prasad Calyam, Aishwarya Venkataraman, Alex Berryman, Marcio Faerman
Ohio Supercomputer Center/OARnet, The Ohio State University, Columbus, Ohio, 43012, USA;
Email: {*pcalyam*, *venkatar*, *berryman*}*@oar.net*, *mfaerman@osc.edu*

***Abstract*—Popular applications such as email, photo/video galleries, and file storage are increasingly being supported by cloud platforms in residential, academia and industry communities. The next frontier for these user communities will be to transition 'traditional desktops' that have dedicated hardware and software configurations into 'virtual desktop clouds' that are accessible via thin-clients. In this paper, we describe experiences from our research and development of virtual desktop cloud experiments in GENI. Our experimentation goal is to investigate and develop optimal resource allocation frameworks and performance benchmarking tools that can enable provisioning (i.e., resource sizing) and placement (i.e., resource mapping) of thin-client based virtual desktops at Internet-scale. We first motivate why virtual desktop cloud experiments cannot be done only at a table-top level, and why infrastructures such as GENI are essential. Next, we detail the methodology of our completed "provisioning" experiments, and our work-in-progress "placement" experiments in GENI that leverage multiple kinds of GENI resources such as aggregates, measurement services and experimenter workflow tools, as well as commercial software. Lastly, we present our vision on how our experiment slice setup and application development experiences, as well as outcomes can be leveraged in classroom labs, and 'living labs' that use GENI resources to foster training and wide-adoption of Future Internet applications.**

***Keywords*-Virtual Desktop Clouds, Cloud Experiment in GENI, Optimal resource Allocation, Classroom Courseware in GENI**

I. Introduction

There has been a rapid adoption of "cloud" platforms for online applications such as email, photo/video galleries and file storage in academia and industry. The next frontier for these user communities will be to transition their "traditional distributed desktops" that have dedicated hardware and software installations into "virtual desktop clouds" (VDCs) that are accessible via thin-clients. Moreover, in the not so distant future, we can envisage home users signing-up for virtual desktops (VDs) with a VD Cloud service provider (CSP) providing Desktop-as-a-Service (DaaS) as a utility. With such a utility service, a thin-client i.e., a set-top-box can be shipped to residential user to access a VD, similar to the model we have today for other common computing and communication needs such as VoIP (e.g., Vonage), and IPTV (e.g., Roku). This box can be connected to television monitors, or computer monitors, and multiple residential users can have their own unique login through this box to their personalized VDs.

The drivers for these transitions of traditional desktops to VDCs are obvious in terms of user convenience, consistent user-perceivable peak performance, and cost-savings: (i) desktop support in terms of hardware, operating system, application and security upgrades will be easier to manage, (ii) the number of underutilized distributed desktops unnecessarily consuming power will be reduced, and (iii) mobile users will have wider access to their desktop applications and data.

However, to allocate and manage VDC resources in a scalable and cost-effective manner at Internet-scale user loads, CSPs are faced with unique "provisioning" (i.e., resource sizing) and "placement" (i.e., resource mapping) challenges. User workload profiles in VDCs are bursty (e.g., during daily desktop startup, when user switches between text and graphic intensive applications), and thin-client user Quality of Experience (QoE) is highly sensitive to network health variations in the Internet. Unfortunately, existing works focus mainly on managing server-side resources based on utility functions of CPU and memory loads [1 - 4], and do not consider network health and thin-client user QoE. There is surprisingly sparse work [5 - 6] on resource adaptation coupled with measurement of network health and user QoE. Works such as [6] and [7] highlight the need to incorporate network health and user QoE factors into VDC resource allocation decisions.

It is self-evident that any cloud platform's capability to support large user workloads is a function of both the server-side desktop performance as well as the remote user-perceived QoE. In other words, *a CSP can provision adequate CPU and memory resources to a VD in the cloud, but if the thin-client protocol configuration or VD placement does not account for network health degradations and application context, the VD may become unusable for the user.*

Another real-world scenario that motivates our research focus is the fact that - *CSPs today do not have frameworks and tools that can estimate how many concurrent VD requests can be handled on a given set of system and network resources at a data center such that resource utilization is maximized, and at worst, the minimum user QoE is guaranteed as negotiated in service level agreements (SLAs).* Hence, resource allocations without combined utility-directed information of system loads, network health and thin-client user experience in VDC platforms inevitably results in costly guesswork and over-provisioning, even for as few as tens of users. Ultimately, such resource allocations tend to annoy users due to high service cost and unreliable QoE.

To overcome the above provisioning and placement challenges, our research and development of VDC experiments [8] in GENI is aimed at coupling "network-and-human awareness" within Internet-scale resource allocation frameworks and performance benchmarking tools for: (a) minimizing costly cloud resource over-provisioning, (b) avoiding guesswork in configuring thin-client protocols and VD placement, and (c) ultimately delivering optimum user QoE of virtualized desktop applications.

This material is based upon work supported by the National Science Foundation under award number CNS-1050225, VMware, and Dell. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Towards this purpose, we are pursuing three research themes:

1) We are developing a virtual desktop benchmarking toolkit viz., "VDBench" [9] for measuring user QoE and resource consumption characteristics of "atomic" (e.g., loading web-page content in an Internet browser) and "aggregate" (e.g., opening an worksheet, making calculations and plotting a graph) tasks executed via thin-client protocol configurations at different system load and network health conditions. For obtaining the measurements, VDBench uses a novel "marker packets" methodology to identify VD flows within a network, as well as to correlate thin-client user events with server-side resource performance events.
2) We are developing a utility-directed resource allocation model (U-RAM) [10] [11] that optimizes resource allocation and management (i.e., VD provisioning and placement) by leveraging utility-functions derived from VDBench toolkit measurements of server-side desktop performance, network performance, as well as the remote user-perceived (objective) QoE.
3) We are validating and tuning our U-RAM and VDBench toolkit in a distributed testbed under realistic user and system loads (i.e., characteristics of steady state, spikes/bursts, flash crowds) using multiple kinds of GENI resources [12], and are investigating adaptations that provide higher transparency and better control of virtual desktop delivery performance at thin-client user sites for a variety of desktop applications.

The purpose of this paper is to inform the GENI community about our VDC experiments: background, progress, accomplishments, broader vision for sustainability in GENI and related open challenges. The remainder of the paper is organized as follows: In Section II, we motivate why VDC experiments cannot be done only at a table-top level in VMLab pilot infrastructure [13] at The Ohio State University, and why infrastructures such as GENI are essential to accomplish our research and development goals.

In Section III, we present an overview of our completed "provisioning" experiments, and our work-in-progress "placement" experiments that leverage multiple kinds of GENI resources that include: aggregates such as ProtoGENI, GENI Meso-scale backbone network, measurement services such as OnTimeMeasure and experimenter workflow tools such as Gush, as well as commercial software such as VMware View.

In Section IV, we detail our experiment slice setup/configuration, VDC application development as well as its deployment within GENI to enable other experimenters to potentially reproduce our experiments or at the least, use any best-practices we adopted in GENI.

In Section V, we present our vision on how our experiment experiences as well as outcomes can be leveraged in: (a) classroom labs such as the one being planned at Purdue University [20], and (b) "living labs" being envisioned as part of the US Ignite initiative [21]. These labs aim at leveraging GENI resources to foster training and wide-adoption of Future Internet applications.

Lastly, in Section VI, we conclude the paper by presenting a GENI outlook discussion on current capabilities and limitations in early experimenter support. In addition, we identify on-going activities in GENI that could make it easier for us and for future experimenters to setup and execute experiments similar to our VDC experiment in GENI.

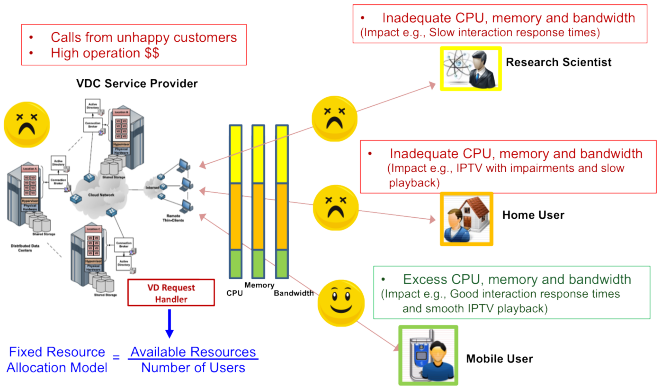

Fig. 1. VD Clouds Today - Overprovisioning and Guesswork in resource allocation with F-RAM

II. GENI Research Experiment Context

1) Research Problem: A "utility function" indicates how much of application performance in a VD can be increased with larger resource allocation [10] [14]. Beyond a point, VD application utility saturates and any additional resource allocation fails to further increase application performance. Today, VDCs allocate resources to VD requests without the awareness of system, network and human utility functions and thus tend to use a Fixed Resource Allocation Model (*F-RAM*), where all VDs are allocated equal amount of resources (e.g., every VD gets 2 GHz CPU, 2 GB RAM, 2 Mbps network bandwidth) as shown in Fig. 1. Such allocations overprovision and guess the required amount of resources needed for different VDs, and thus increase operation costs due to resource wastage and could fail to meet user SLAs. Consequently, some users are allocated more resources than their application utility saturation points (see mobile user in Fig. 1), and other users who need more resources are under provisioned (see home user or research scientist in Fig. 1).

In comparison, *U-RAM* uses awareness of utility functions of system, network and human components and profiles VD requests into distinct desktop pools in order to provision the appropriate amount of resources within each desktop pool. Thus, U-RAM succeeds in minimizing operation costs and reliably meets user SLAs as shown in Fig. 2. To realize the benefit of U-RAM, we can see from Fig. 2 that additional instrumentation and measurement with VDBench toolkit is needed on the thin-client side, and resource adaptation has to be based on profiling and feedback of thin-client and server performance. The feedback can also be used for user QoE measurement and bottleneck analyses as part of user "auto-support" services of CSPs, who need the ability to qualify and troubleshoot VD user QoE from the server-side (i.e., from within the cloud).

2) VMLab → GENI Transition Motivations: Our early experiments in developing VDBench toolkit and U-RAM were performed in a table-top manner in our "VMLab" pilot infrastructure [13] at The Ohio State University. VMLab is a shared testbed developed in collaboration with VMware, Dell and IBM in support of desktop virtualization experiments for research and education communities. As shown in left-half of Fig. 3, we used VMLab as a VDC data center that features: (a) popular user applications (Spreadsheet Calculator, Internet Browser, Media Player, Interactive Visualization), and (b) TCP/UDP based thin-client protocols (Microsoft RDP,

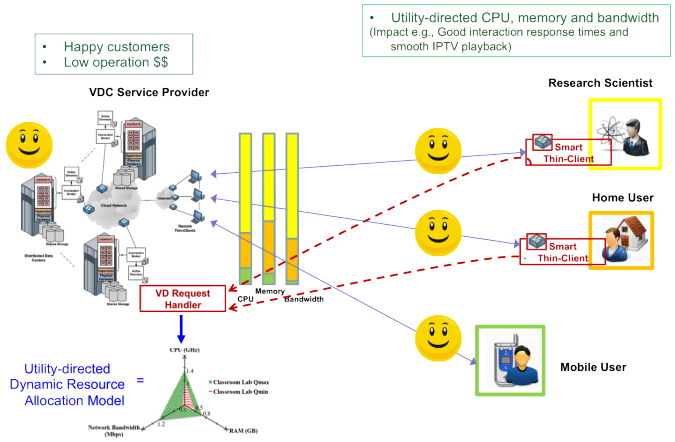

Fig. 2. VD Clouds in the Future - Intelligent resource allocation with U-RAM and smart thin-clients

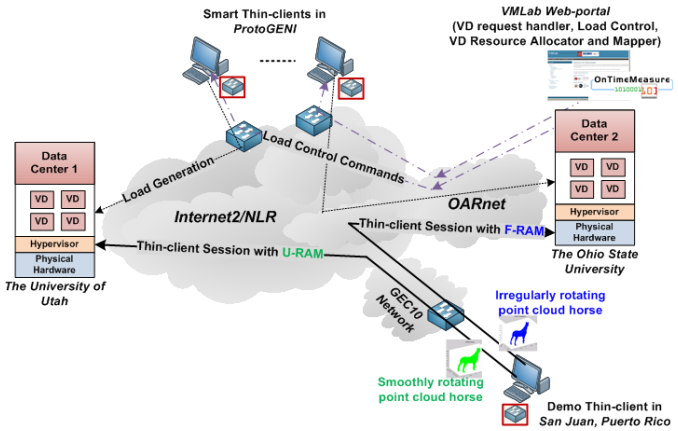

Fig. 4. GEC10 Demo of VDCloud Provisioning Experiment to compare U-RAM and F-RAM Schemes

HP RGS, Teradici PCoIP) under a variety of emulated user load and network health conditions. We obtained combined utility functions of exemplar user groups from the testbed [10] [11], which in turn were used to derive corresponding desktop pools in U-RAM. For argument sake, we considered three user groups: Engineering Site, Distance Learning Site and Campus Lab, each provisioned as a separate desktop pool within the VDC with a custom set of applications that use different amounts of CPU, memory and network bandwidth resources.

Although our preliminary experiments within VMLab showed promise, true validation of U-RAM and VDBench toolkit methodologies, and demonstration of their relevance - needed to be done in a multi-data center VDC setting with more realistic user, network and system loads, and with multiple geographically distributed thin-client sites as shown in the right-half of Fig. 3. We found that the GENI infrastructure which has emerged as the "federated cloud" to be ideal for transitioning and scaling up our VDC experiments. It provides state-of-the art system and networking resources, as well as a vibrant user community that is engaged in the creation, support and usage of next-generation system and networking technologies. It also provides wide-area network programmability that gives us the flexibility to conduct dynamic allocations and migrations in our VDC experiments. The computational power, networking bandwidth, network programmability, user opt-in mechanisms, instrumentation-and-measurement services, and experiment workflow tools in GENI are also appealing to run cloud-based research experiments in a repeatable, scalable and distributed manner.

III. Experiment Goals and Accomplishments

3) Provisioning Experiments: In our first set of experiments in GENI illustrated in Fig. 4, we validated the provisioning performance of U-RAM described in our related work in [10]. We were able to setup a VDC experiment in the GENI infrastructure using ProtoGENI/Emulab system resources [15], network resources spanning Internet2/NLR and OARnet, OnTimeMeasure instrumentation and measurement service [16], and Gush experimenter-workflow tool [17]. We evaluated the feasibility of including PlanetLab [18] resources into our experiment, but were unable to utilize the resources due to lack of stability in the nodes to host a data center environment, and also due to lack of graphics capabilities such as X11 server needed to initiate thin-client VD connections. Given that our VDC experiment had special needs (e.g., there has not been a precedent in ProtoGENI/Emulab with setting up ESXi hypervisor as part of a cloud experiment's data center), we faced challenges initially in our GENI slice setup. However, we have successfully over time - overcome most of the challenges with very helpful assistance from the ProtoGENI team (e.g., we now have a custom OS image installation procedure that is consistent with ProtoGENI operations and is repeatable).

We gave a live demonstration of our VDC experiment in GENI at the GEC10 Networking Reception. We setup a slice with 2 data centers, one at OSU VMLab and one at Utah Emulab and reserved several other nodes in a separate "user slice" to emulate thin-client VD connections. We set rate limits at the data centers to 10 Mbps network bandwidth using network emulators in order to setup a realistic environment for data center networking for approximately 15 VD users. Although we had setup the Utah Emulab data center earlier for a brief period, due to slice expiration and our initial challenges in making the hypervisor installation repeatable, we were unable to use the Utah Emulab data center in our live demonstration. As a result, we ended up using the OSU VMLab data center for the U-RAM and F-RAM performance comparisons.

On the user nodes reserved in GENI, we installed our VDC application scripts, and OnTimeMeasure. Our VDC application scripts were used to control the load generation of thin-client VD connections, and OnTimeMeasure was used along with VMware tools to collect and analyze performance measurements in the network and at host resources. The VMware tools provided an API for OnTimeMeasure to obtain measurements relating to CPU, memory, and number of VD connections at server-side. OnTimeMeasure was instrumented with the Gush experiment XML files to control sampling and query of measurements. Root Beacon of OnTimeMeasure was installed at the OSU VMLab data center, and several Node Beacons of OnTimeMeasure were installed at the thin-client VD nodes to measure end-to-end network path measurements of available bandwidth, latency and loss.

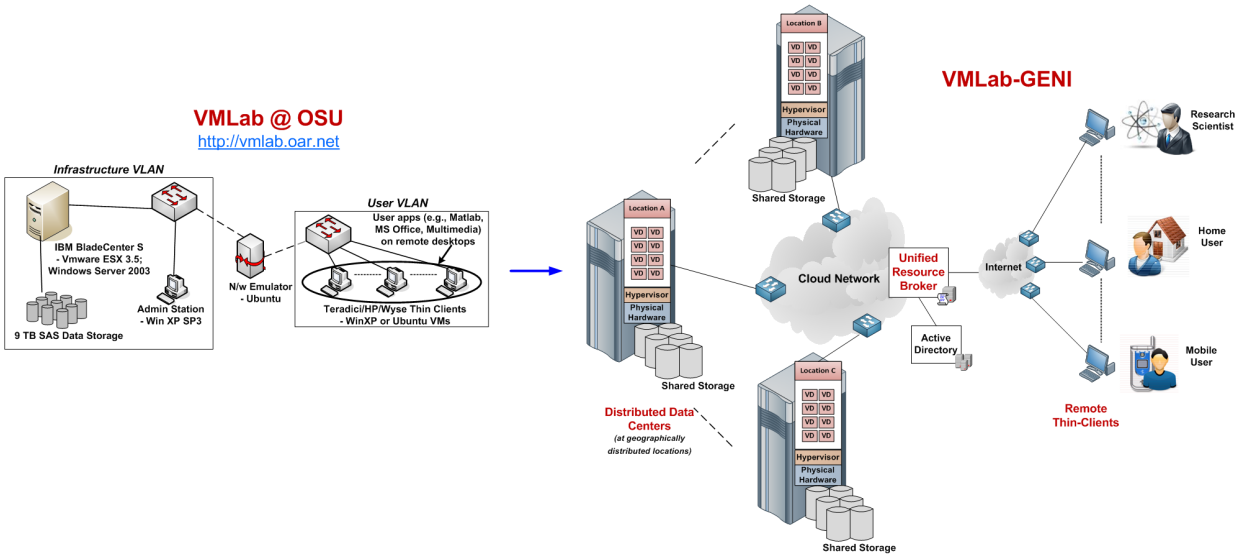

Fig. 3. VMLab Table-top Experiments transition to VMLab-GENI Cloud Experiments

A "cloud broker" [1] was developed that had a web-portal user interface to live demonstrate increasing system and network loads at experiment's data centers through generation of thin-client connections belonging to different user desktop pools. The user slice included a demo site at the GEC10 Puerto Rico conference venue, which initiated two separate thin-client connections - one to check U-RAM performance and one to check F-RAM performance. We used Matlab-based animation of a horse point-cloud as the VD application to demonstrate that U-RAM provides "improve performance" and "increased scalability" in comparison to F-RAM. We characterized satisfactory user QoE performance as the "smooth" rotation, and unsatisfactory user QoE performance as the "irregular" rotation - of the point-cloud horse animation. More details of the above experiment setup and results can be found in [12].

4) Placement Experiments: In our second set of experiments in GENI illustrated in Fig. 5, we are working towards validating the performance of coupling U-RAM provisioning with subsequent "placement" strategies across distributed data centers, as described in our related work in [11]. The placement decisions in VDCs are influenced by session latency, load balancing and operation cost constraints. In addition, placement decisions need to be changed over time for: (a) proactive defragmentation of resources for improved performance and scalability, and (b) reactive VD migrations for increased resilience and sustained availability. The proactive defragmentation of resources is performed using global optimization schemes to overcome the "resource fragmentation problem" in VDCs that occurs due to placements being done opportunistically to reduce user wait times for initial VD access. We refer to opportunistic placements as those that are performed using local schemes that use high-level information about resource status in data centers. Over time, resource fragmentation due to careless packing of VDs on resources and due to changing application workloads - leads to the "tetris effect" [19] that decreases scalability (VDs/core) and performance (user QoE), hence the VDC Net-Utility. In contrast, the reactive VD migrations are triggered by cyber-attack or planned maintenance events, and should be performed in a manner that does not drastically affect the VDC Net-Utility. However, it is important to note that - not all VD migrations suggested by proactive or reactive schemes generate positive benefit in VDC Net-utility, since VD migration is an expensive and disruptive process. Hence, we model the cost-of-migration and normalize it to utility of VDs, and migrate only the VDs ("positive pairs") that generate positive Net-benefit in the VDC.

[1]The objective of a cloud broker in practice is to handle online VD requests and make decisions regarding the provisioning (i.e., resource sizing) placement and (i.e., resource mapping) of resources. Thereby, it ensures user QoE is satisfied while maintaining scalability of the VDC in terms of the overall number of VDs handled.

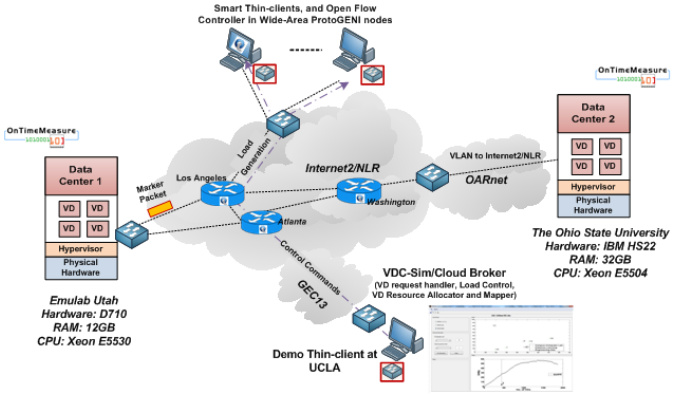

Fig. 5. "Work-in-progress" VDCloud Provisioning and Placement Experiment to validate U-RAM coupled with Cost-aware Migration

IV. Slice Topology Setup and Configuration

We now describe our latest experiment slice topology setup and configuration that we are developing to validate our U-RAM coupled with cost-aware migration scheme. There are four major components of our VDC experiment: (i) data centers, (ii) thin-client sites, (iii) programmable network, and (iv) cloud broker application. We have a 2 data center configuration comprising of OSU VMLab and Utah Emulab resources with IP connectivity (Layer 3) on the Internet. The "wide-area ProtoGENI" (WAPG) nodes located at meso-scale campuses

(e.g., Stanford, Georgia Tech, Clemson, Wisconsin, BBN) will serve as thin-client sites. One of the WAPG nodes will be used to host our VDC experiments OpenFlow controller. The traffic from the thin-client sites are being made to flow within VLANs (Layer 2) through OpenFlow-enabled network segments in the GENI meso-scale backbone [22] network (operated by Internet2/NLR) before reaching the data centers.

The slice topology has multiple network paths with long and short geographical distances between the thin-client sites and the data centers. As a result, we will be able to investigate the use of OpenFlow for dynamic link bandwidth sizing and the load balancing of multiple thin-clients connecting to the data centers on diverse paths. We were advised to ensure that we develop our OpenFlow implementations such that they could be done in the GENI infrastructure routers/switches at the hardware-level, as opposed to at the software-level, where routing performance can be very slow. Currently, most actions in OpenFlow implementations that pertain to Layer 2 headers can be performed at hardware-level in the GENI infrastructure routers/switches, however manipulating IP headers are mostly done at the software-level.

Hence, the triggering of OpenFlow implementations to identify, route and tear-down flows in our experiment will be based on VDBench "marker packets" (shown in Fig. 5) that will be sent from the data centers to the thin-client sites as part of non-IP traffic sessions. The marker packets contain: (a) Layer 2 information of the VDs in the data centers, (b) information regarding the thin-client protocol being used, and (c) information pertaining to the user group (i.e., Engineering Site, Distance Learning Site and Campus Lab) of the VD session. For emulating the OpenFlow implementation, we recreated our slice topology in MiniNet [23] and validated our OpenFlow controller software with simulated traffic.

We are planning to demonstrate our latest experiment progress at GEC13 using a demo thin-client as shown in Fig. 5. The demo thin-client will be used to show application performance in virtual desktops under varying request loads and fault-conditions. In conjunction, we are planning to demonstrate our VDC simulator viz., "VDC-Sim" that also acts as a cloud broker application for: VD request load control using adjustable sliders, resource provisioning (i.e., U-RAM) and placement (i.e., local and global cost-aware optimizations) scheme selections, as well as for introducing fault-levels (e.g., cyber-attacks, planned maintenance). We have already seen promising results running simulation scenarios within VDC-Sim, and we hope to validate our VDC-Sim with our experiment demo at GEC13.

In Appendix A through C, step-by-step instructions on our experiment slice setup/configuration as well as its deployment within GENI are presented. Our hope is that these instructions will enable other experimenters to potentially reproduce our experiments or at the least, use any best-practices we adopted.

V. Sustainability and Broader Impact

A. Experiment Evolution Requirements

We believe that our VDC research experiment experiences and outcomes could serve as a basis for education and innovation use cases, which leverage GENI resources to foster and cross-pollinate: training, entrepreneurial enterprise outgrowths and wide user-adoption - of Future Internet applications. We hope to evolve our experiment in a manner that it can be sustainable in the GENI community, and can attract actual users for a future VDC instance leveraging GENI capabilities. Also, we hope that we can convincingly demonstrate the relevance of our schemes so that CSPs and industry vendors of VDC related technologies such as VMware and Dell can eventually adopt our schemes that are validated in GENI. The evolution requirements that we plan to address relate to critical attributes such as: *reproducibility*, *modularity*, *programmability* and *intuitiveness* (or ease of use). Fortunately, these are the evolution requirements of the various GENI components (e.g., infrastructure, software services) as well, that are inherent in our VDC experiment. Ultimately, we plan to develop a "portable VDC GENI experiment suite" that encompasses our VDC-Sim, experiment methodology, VDBench toolkit, cloud broker application and best practices in GENI. In the following two sub-sections, we describe our early efforts to improve sustainability and influence broader impacts of our VDC experiment within the GENI community, and in turn motivate the development of our VDC experiment suite.

B. Classroom Lab Use Case

We are collaborating with Purdue University to develop classroom courseware (details of the proposed courseware can be found in [20]) that exposes students to concepts relating to computer and network virtualization, which have been realized in practice within the GENI infrastructure. Students will have the opportunity to obtain hands-on experience by modifying our VDC experiment suite in GENI and will be able to compare resource allocation schemes and performance of thin-client protocols for diverse virtualized desktop applications. They will be able to use the VDBench and OnTimeMeasure tools to obtain network, system and user QoE measurements, and will be able to reconfigure GENI slices to optimize resource allocations. Further, as part of a more advanced course or for extra credit, students could extend the VDC experiment suite for more in-depth evaluations of thin-client protocol optimization (e.g., aggressive pipelining, forward error correction, tunneling) or even write new OpenFlow controllers within the cloud broker application in the VDC experiment suite.

C. "Living Lab" Use Case

We are collaborating with cities of Dublin, OH and Chattanooga, TN to develop "living labs" being envisioned as part of the US Ignite initiative [21]. Both these cities have made significant investments in broadband access and data center infrastructures, and are seeking to deploy novel Gigabit applications such as our VDC experiment in GENI for their local communities, especially for high-tech small businesses. The cities are working towards supporting GENI-enabled racks in their data centers and are willing to provide "*cheap virtual desktops with custom applications*" for small businesses to use on-demand. As part of our extended VDC experiments in GENI, our plan is to conduct pilot studies that will enable small businesses in these cities to use virtual desktops that generate business value. As a consequence of the studies: (a) the cities will realize the challenges (e.g., policy, technology feasibility) of providing the service to small businesses, (b) we will have the opportunity to modify our VDC experiment suite to cater real user needs, (c) small business users will be able to assess benefits from such a city-supported service, and (d) a business model could be developed for making the city-supported service a commercially viable enterprise.

VI. Conclusions

As an early experimenter in GENI, we were expecting to gain several benefits as well as face several challenges in working with the GENI infrastructure and development community. We found that the GENI community is well setup to provide experimenter support for our VDC experimentation through friendly point-of-contacts in the various GENI projects. In addition, we found that the GENI initiative greatly removes the burden for us to procure physical resources, and has several tools to setup and perform routine tasks involved in running experiments. The GENI Engineering Conference sessions, demonstrations at networking receptions, tutorials, wikis, mailing-lists and cluster conference calls are very helpful to learn about GENI-wide activities, discuss experiment ideas and find resolutions to any experiment-related problems.

In terms of challenges, we found in our early days of setting up our experiments that there were sparse resources with high amounts of CPU and RAM for setting up multiple geographically distributed VDC data centers. In addition, given the limited number of available distributed nodes that can act as thin-client sites, we have not been able to run as many thin-client sessions in our experiments as we had originally hoped. Also, given the shared nature of the infrastructure, experimenters are not encouraged to have long-standing reservations of non-sharable resources (such as ProtoGENI resources) in GENI slices. Further, given that the GENI community is rapidly enhancing infrastructure and software, finding appropriate documentation on our own was a challenge, and there is a significant learning curve for a student in setting up advanced experiments. However, given the facts that: (a) GENI racks and other meso-scale deployments are ramping up, (b) tools to help experimenters are being improved and integrated into GENI on a daily-basis, and (c) there is a growing trend in the documentation examples that experimenters can use to build on - the outlook for future experimenters in terms of capabilities and ease-of-use in GENI is promising.

The ultimate realization of GENI will be when we will be able to perform large-scale VDC experiments that possess the dynamic nature of the Internet in terms of: (a) user behavior/cyber-attacks/cross-traffic, (b) actual user accessible VDC services, and (c) classroom labs that use VDC services in GENI. At that point, we can expect that words in the GENI glossary such as "slice" will have become mainstream vocabulary in the society.

References

[1] D. Gmach, S. Krompass, A. Scholz, M. Wimmer, A. Kemper, "Adaptive Quality of Service Management for Enterprise Services", *ACM Transactions on the Web*, Vol. 2, No. 8, Pages 1-46, 2008.

[2] P. Padala, K. Shin, et. al., "Adaptive Control of Virtualized Resources in Utility Computing Environments", *Proc. of ACM SIGOPS/EuroSys*, 2007.

[3] B. Urgaonkar, P. Shenoy, et. al., "Agile Dynamic Provisioning of Multi-Tier Internet Applications", *ACM Transactions on Autonomous and Adaptive Systems*, Vol. 3, No. 1, Pages 1-39, 2008.

[4] H. Van, F. Tran, J. Menaud, "Autonomic Virtual Resource Management for Service Hosting Platforms", *Proc. of ICSE Workshop on Software Engineering Challenges of Cloud Computing*, 2009.

[5] K. Beaty, A. Kochut, H. Shaikh, "Desktop to Cloud Transformation Planning", *Proc. of IEEE IPDPS*, 2009.

[6] N. Zeldovich, R. Chandra, "Interactive Performance Measurement with VNCplay", *Proc. of USENIX Annual Technical Conference*, 2005.

[7] J. Rhee, A. Kochut, K. Beaty, "DeskBench: Flexible Virtual Desktop Benchmarking Toolkit", *Proc. of Integrated Management (IM)*, 2009.

[8] "Virtual Desktop Cloud" GENI Experiment Wiki - http://groups.geni.net/geni/wiki/FirstGenCalyam

[9] A. Berryman, P. Calyam, A. Lai, M. Honigford, "VDBench: A Benchmarking Toolkit for Thin-client based Virtual Desktop Environments", *IEEE Conference on Cloud Computing Technology and Science (CloudCom)*, 2010.

[10] P. Calyam, R. Patali, A. Berryman, A. Lai, R. Ramnath, "Utility-directed Resource Allocation in Virtual Desktop Clouds", *Elsevier Computer Networks Journal (COMNET)*, 2011.

[11] M. Sridharan, P. Calyam, A. Venkataraman, A. Berryman, "Defragmentation of Resources in Virtual Desktop Clouds for Cost-Aware Utility-Optimal Allocation", *IEEE Conference on Utility and Cloud Computing (UCC)*, 2011.

[12] P. Calyam, M. Sridharan, Y. Xu, K. Zhu, A. Berryman, R. Patali, A.Venkataraman, "Enabling Performance Intelligence for Application Adaptation in the Future Internet", *Journal of Communications and Networks (JCN)*, 2011.

[13] P. Calyam, A. Berryman, A. Lai, R. Ramnath, "VMLab Testbed for Desktop Virtualization to Support Research and Education", *MERIT Desktop Virtualization Summit, East Lansing, MI*, 2010 - http://vmlab.oar.net.

[14] R. Rajkumar, C. Lee, J. Lehoczky, D. Slewlorek, "Resource Allocation Model for QoS Management", *Proc. of IEEE Real-Time Systems Symposium*, 1997.

[15] ProtoGENI Aggregate in GENI - http://www.protogeni.net

[16] OnTimeMeasure Instrumentation and Measurement Service in GENI - http://groups.geni.net/geni/wiki/OnTimeMeasure

[17] Gush Experimenter Control Tool in GENI - http://groups.geni.net/geni/wiki/GushProto

[18] PlanetLab Aggregate in GENI - http://groups.geni.net/geni/wiki/PlanetLab

[19] A. Hillier, "Server Capacity Defrag: Maximizing Infrastructure Efficiency through Strategic Workload Placements", *CiRBA Whitepaper*, 2010.

[20] S. Fahmy, "Programming Assignments for Graduate Students using GENI", *GENI Courseware Whitepaper*, 2011. http://groups.geni.net/geni/attachment/wiki/ScalableMonitoring/lab.pdf

[21] NSF/OSTP US Ignite Initiative - http://us-ignite.org

[22] TangoGENI Operations - http://groups.geni.net/geni/wiki/TangoGENI

[23] Mininet Software-defined Network Emulation Platform - http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Mininet

Appendix A
Data centers

A. Resource Reservation

Using the below NS file, we requested a slice in Emulab with a Dell D710 hardware node and a compatible OS:

```
set ns [new Simulator]
source tb_compat.tcl
set esxi [$ns node]
tb-make-soft-vtype N {d710}
tb-set-hardware $esxi N
tb-set-node-os $esxi FEDORA10-STD
$ns rtproto Static
$ns run
```

B. Image Bootup

The ESXi Kickstart files obtained from the ESXi ISO were deployed on to a web server (http://⟨IPAddress⟩/kickstart/)

1) A gPXE config 'default' file was created to describe the URL location of the kernel files that the web server accesses.

```
default 1
prompt 1
menu title VMware VMvisor Boot Menu
timeout 50
label 1
kernel http://<IP Address>/kickstart/esxi/mboot.c32
append http://<IP Address>/kickstart/esxi/vmkboot.gz ks=http
    ://<IP Address>/kickstart/ks.cfg --- http://<IP Address
    >/kickstart/esxi/vmkernel.gz --- http://<IP Address>/
    kickstart/esxi/sys.vgz --- http://<IP Address>/
    kickstart/esxi/cim.vgz --- http://<IP Address>/
    kickstart/esxi/ienviron.vgz --- http://<IP Address>/
    kickstart/esxi/install.vgz
label 0
localboot 0x80
```

C. Server Customization

1) A kickstart config file 'ks.cfg' was created that was used to customize the ESXi's servers at both the OSU VMLab and the Utah Emulab data centers.
   The following fields were defined:
   a) root password
   b) disk format issue
   c) web server URL of kernel files
   d) network settings

```
# Accept the VMware End User License Agreement
vmaccepteula

# Set the root password for the DCUI and Tech Support
    Mode
rootpw root-password

# Choose the first discovered disk to install onto
autopart --firstdisk --overwritevmfs

# The installation media is in the CD-ROM drive
install url http://<IP Address>/kickstart/esxi

# Set the network to DHCP on the first network
    adapater
network --bootproto=dhcp

##%*post --unsupported --interpreter=busybox*
##reboot
```

   At the OSU VMLab data center, the below steps were done by our system administrator. At the Utah Emulab data center, we contacted the Emulab administrator for the steps listed below.
2) gpxelinux.0 and 'default' files were given to the Emulab/VMLab administrator to place on their TFTP server.
   a) The administrator manually set the DB state for that node to use gPXE
   b) The node was then restarted so that it booted using gPXE
   c) The scripts that were previously configured automated the install process. Upon completion of the install, the ESXi node stops at a screen saying 'Press any key to reboot'
   d) The Administrator modified the boot order for the node so that it was able to boot off of the local hard drive, and not the gPXE. The administrator also set boot partition to "0" so that it booted through the MBR. Failure to do this will cause the installation process to continually loop.
   e) TCP/UDP port 902 on the Emulab firewall had to be opened for this node.
3) At this point, we were able to connect to the Utah Emulab/OSU VMLab ESXi server with the VMware vSphere Client.

D. Deploying Desktop Pools

The layer 2 connectivity that was described in Section IV brings many benefits in addition to the programability that OpenFlow provides. Since both data centers will be on the same VLAN, the multiple data center setup can be simplified to a single IP subnet. This in turn simplifies our DHCP, DNS, and Active Directory (AD) configuration across both data centers. We can use the existing DHCP, DNS, and AD in OSU VMLab to support the Utah Emulab data center. The only redundant component we need is the connection broker i.e., VMware View Manager that tunnels a VDC request to the virtual desktop. Once we have a connection broker in place, we can deploy our desktop pools in Utah Emulab using the same methodology we used in OSU VMLab. The steps to create pools in a data center are listed below:

```
1. Login to the Connection server web-portal http://<IP
    Address>/admin
2. Go to Inventory -> Pools and Click Add
3. Select Automated Pool
4. Select Dedicated Pool and Enable Automated Assignment
5. Select View Composer Linked Clones
6. Enter an ID and give Display name as <usergroup>
7. Pool settings, View Composer Disk settings can be left to
     default settings
8. In the provisional settings, give name convention as <
    usergroup_> and pool size ~10
9. In the Default image, select an appropriate
    Windows_XP_template and appropriate snapshot.
10. Select appropriate virtual machine folder, and select
    host and cluster where you want to deploy the pool.
11. Repeat these for each of the <usergroups: engineering
    site, distance learning site, campus lab>
```

APPENDIX B
THIN-CLIENT SITES

We developed a custom package (available upon request) that we installed before starting the setup of the thin-clients and the cloud broker application. The package contains the scripts necessary for configuring thin-clients and the cloud broker application.

A. Resource Reservation

We created an "user slice" in GENI and reserved several nodes that were used as thin-clients. This step requires a ProtoGENI user account. The sample .ns file below requests for 5 nodes with FEDORA-8 .

```
set ns [new Simulator]
source tb_compat.tcl
set node1 [$ns node]
set node2 [$ns node]
set node3 [$ns node]
set node4 [$ns node]
set node5 [$ns node]
set lan0 [$ns make-lan "$node1 $node2 $node3 $node4 $node5"
    100Mb 0ms]
tb-set-node-os $node1 FEDORA8-STD
tb-set-node-os $node2 FEDORA8-STD
tb-set-node-os $node3 FEDORA8-STD
tb-set-node-os $node4 FEDORA8-STD
tb-set-node-os $node5 FEDORA8-STD
$ns rtproto Static
$ns run
```

The above .ns file creates a LAN with 5 ProtoGENI nodes.

B. Configuring Thin-clients

The nodes were installed with software to enable: (i) viewing the performance of allocated VDs from the data center (via VMware View Client), and (ii) users to connect via thin-clients (via VNC Viewer) to the data centers to generate request loads. We also created automated scripts that stored username and password and installed them on all the reserved nodes.

1) To enable auto-login, users need to authenticate with the new ProtoGENI nodes by logging into all the new ProtoGENI nodes. This step can be avoided by adding the experiment domain to known host list in ~/.ssh/config. For example, the below step will ensure that any host from emulab.net is added to the list of known hosts.

```
echo -e "Host *.emulab.net\n\tStrictHostKeyChecking no
    \n" >> ~/.ssh/config
```

2) To install the required software, we ran the bash script listed below which automatically installed all the required scripts on all the thin-client nodes in listofnodes.

a) login credentials for ProtoGENI node were entered in RemoteLogin.pl file, lines 15 and 16.

```
my $user="<your-ProtoGENI-username>";
my $pass="<your-ProtoGENI-password>";
```

b) Automation of installation procedure:

```
./RemoteLogin.pl file-with-list-of-all-nodes ./
    emulab.sh
```

To do above step manually, login to each thin-client node and install software individually as listed below:

i) Download VMware View Client. For this, login to one of your ProtoGENI nodes and execute:

```
wget http://vmware-view-open-client.
    googlecode.com/files/VMware-view-open-
    client-4.5.0-297975.x86_64.rpm
```

ii) Copy the contents of the nodeSetup folder on your web-server to the home directory of your ProtoGENI node.
Next, manually login to each of the nodes and execute the emulab.sh script to install the pre-requisites. Also it is required to create a new password for vncserver. This would be required while connecting to the data center through VNC viewer.

iii) rDesktop setup: rDesktop setup is done by running scripts corresponding to the usergroup and the resource allocation scheme preferred. The contents of the script are shown below:

```
#!/usr/bin/perl -w
use strict;
system "sudo killall Xvnc;rm -rf .vmware";
eval {`vncserver :10`};
sleep 15;
print "starting vmview..";
eval {`vmware-view -s http://<IP Address>:80
    -d vm.lab -n "Engineering" -u
    Engineering -p engineering -q --display
    =:10> view.log 2>&1 &`};
print "Done.";
```

In the second last-line, following parameters are configured:

```
-s: view-connection broker address
-d: domain name for your VHDs network
-n: resource pool name
-u: usergroup name
-p: password for the usergroup
-q: silent mode
```

The required credentials change based on the usergroup.

APPENDIX C
CLOUD BROKER APPLICATION

The cloud broker application for the resource allocation scheme was controlled from a web-portal in our VDC provisioning experiments. Experimenters could choose a resource allocation scheme and move sliders to simulate VD request loads. Starting the experiment by clicking on "Run" button would then invoke a series of scripts in the VDCloud-GENI.tar.gz package. For our VDC placement experiments, we are developing a MATLAB-based user interface as our cloud broker application. The setup steps listed below are common for both the web-portal and the MATLAB based applications.

1) Installing Required Perl Modules
For the VDC provisioning experiment, install Net::SSH::Perl before running the scripts. For the VDC placement experiment, below is the list of modules that were installed:

```
Net::SSH::Perl
Math::Matrix
List::Util
Config::Tiny
Config::IniFiles
```

2) Editing the nodes-list file:
We listed all the nodes used in the experiment in AllNodesURAM, AllNodesFRAM, vanillaNodesURAM and vanillaNodesFRAM to enable scripts to access the status of these nodes. "status" indicates if the node is available or if it is already assigned to an usergroup as determined by the cloud broker.

```
status,your-ProtoGENI-node-name
```

Initially, status is set to available, the status thereon is maintained as per experiment state. Example:

```
available,node1.VMLab.geni.emulab.net
available,node2.VMLab.geni.emulab.net
available,node3.VMLab.geni.emulab.net
available,node4.VMLab.geni.emulab.net
available,node5.VMLab.geni.emulab.net
```

3) The resourceAtDataCenters file contains the resources available at the data centers used for the experiment. The values in this file could be set to be less than the actual resources in order to constrict certain resource consumption behavior. An example set of values is shown below:

```
L1,"CPU",16,"Network",50,"RAM",32
```

The userGroupUtilityFunction file contains the usergroup characteristics from 'VDBench' measurements. An example set of values is shown below:

```
Campus Computer Lab,"CPU",0.5,1.4,"RAM",0.2,0.8,"
    Network",0.1,1.2
Engineering Site,"CPU",0.7,1.5,"RAM",0.350,1.2,"
    Network",0.2,1.5
Distance Learning,"CPU",0.3,1.5,"RAM",0.2,1.1,"Network
    ",2,4
```