Towards a User Deployable Service-oriented Autonomic Multi-cloud Overlay Infrastructure for Sky Computing

Courtney Powell[1], **Masaharu Munetomo**[1], **and Takashi Aizawa**[2]
[1]Information Initiative Center, Hokkaido University, Sapporo, Japan
[2]Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan

Abstract—*Cloud computing services at the SaaS level of the cloud computing model are easy to access and utilize. In contrast, the self-service approach taken at the IaaS level results in difficulties for non-savvy users. Compounding these difficulties is the fact that for resiliency, economic, and scalability reasons, utilization of IaaS resources across multiple cloud providers in a unified manner is deemed the best strategy. In this paper, we outline our proposal for a user-centric multi-cloud autonomic overlay infrastructure that can be deployed across existing cloud systems without the need for specialized hardware or action on the part of cloud providers. We also present the results of a simple genome sequencing experiment and bandwidth measurements conducted on a crude prototype implementation of the system*.

Keywords: virtual distributed ethernet (VDE), application defined networking (ADN), virtual overlay infrastructure, simple heterogeneous inter-cloud manager (SHINCLOM), cloud federation, CloudStack.

1 Introduction

Cloud computing has rapidly advanced into the public consciousness, with software as a service (SaaS) becoming a virtually indispensable part of everyday life due to easy access and utility. In cloud computing, everything is delivered as a service [1]. However, for users who require more than the relatively simple applications offered at this uppermost layer of the cloud computing model, things are not as simple because the lower levels utilize a self-service approach, in which users are expected to provision, manage, and maintain what they need by themselves. Deploying applications at the infrastructure as a service (IaaS) level is non-trivial as the distributed applications being deployed often comprise interdependent services that form a complex hierarchy across virtual machines (VMs), and which must be configured in a particular order [2]. Thus, simply accessing and utilizing IaaS resources on any one cloud platform requires a variety of technical know-how—such as knowledge of the relevant APIs to use and how to install, launch, configure, and maintain the desired applications and services.

Adding to the difficulties outlined above is the fact that mission critical business applications require that downtime be minimized and optimum performance maintained; yet, it has become patently clear that trusting essential applications to one cloud platform or provider is not a wise strategy due to the possibility of outages and scalability, flexibility, and economic issues. Further, at present, the frightening possibility of cloud lock-in [3] awaits unwary users as dynamically migrating applications and services from one cloud to another or launching and maintaining applications across cloud systems in order to avail oneself of better prices, levels of service, etc. is still very difficult.

Cloud federation is an emerging paradigm in which resources from multiple independent cloud providers are leveraged to create a virtual cloud system that overcomes some of the limitations of single-cloud systems (such as provisioning and scalability constraints), is resilient to failures, and provides high availability. As outlined by Petcu [4], the actual clouds can be federated in a number of ways: Horizontal federation, Inter-Clouds federation, Cross-Clouds federation, and Sky Computing [5].

In this paper, we outline our user-centric approach to cloud-federation, with which we ultimately aim to spare users the tedious, time-consuming, and error-prone process of manually installing, configuring, and monitoring multi-cloud applications and services at the IaaS level. Consequently, we are currently developing a user deployable autonomic multi-cloud overlay infrastructure comprising various applications, utilities, and services that users can easily deploy and utilize. In this sense, our objective is a form of Sky Computing as our aim is to compose the *existing user-level services* offered by cloud providers into a single virtual framework (in the form of a "single-cloud like image") that is independent of any one type of cloud platform and offering new services other than those provided by each individual cloud provider (i.e., greater than the sum of its individual parts). The actual infrastructure is being developed in a holistic way with the aid of a proposed layered autonomic multi-cloud model on which existing and future technologies can be integrated in such a way that how the various elements in our framework function and interoperate can be easily understood.

The remainder of this paper is organized as follows: In Section 2, we introduce and outline our proposed layered autonomic multi-cloud model. In Section 3, we discuss our preliminary implementation of a user deployable virtual overlay infrastructure and MPI clusters. We also discuss the

results of a simple genome sequencing experiment conducted on the clusters and bandwidth measurements done on the infrastructure. In Section 4, we outline future work to be done. Finally, we conclude this paper in Section 5.

2 Autonomic Multi-cloud Model

We propose that autonomic multi-cloud applications and services be developed based on the conceptual model depicted in Fig. 1. This conceptual model allows for the design and development of applications and services in a cloud-agnostic way. Our aim is to leverage the many technologies and techniques currently available and map them onto the relevant layers of our model in a transparent manner by adapting, integrating, and (where necessary) modifying and extending them in such a way that the links among the technologies are easily analyzable and optimizable. The layers and the functions they perform in the model are discussed below.

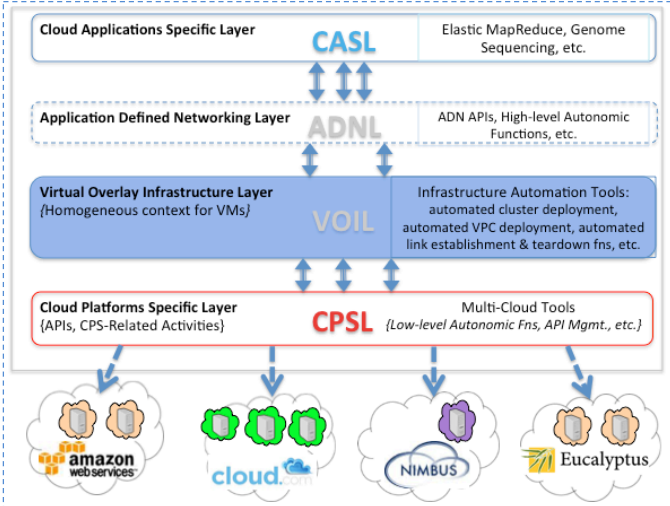

Fig. 1: Proposed layered autonomic multi-cloud model

2.1 Layer 1: Cloud Platforms Specific Layer (CPSL)

There are currently a large number of public and private cloud platforms, some better known than others. At present, these cloud platforms do not use a common set of APIs. The cloud platforms specific layer (CPSL) forms the foundation of our model as it translates the various calls from the upper layers into calls that can be initiated on the various platforms. Utilities such as LibCloud [6] and DeltaCloud [7] can be deployed at this layer to carry out these functions. Further, tools such as CloudInit.d [8] and Wrangler [9] can be extended and integrated to provide low-level autonomic functions such as monitoring and repair of VMs at this layer.

2.2 Layer 2: Virtual Overlay Infrastructure Layer (VOIL)

The virtual overlay infrastructure layer (VOIL) is a critical part of our proposed model. At this layer, the following activities and services are provided:

1. A seamless virtual overlay infrastructure that abstracts away the differences inherent in the various cloud platforms.
2. A homogeneous context for VMs from disparate clouds.
3. Infrastructure automation tools and services such as automated cluster and VPC deployment services; akin to the tool execution environment envisioned by Afgan et al. [10].

2.3 Layer 3: Application Defined Networking Layer (ADNL)

At the application defined networking layer (ADNL), the focus is on optimization and orchestration of the movement of data throughout the VOIL for each application. Application defined networking (ADN) [11] takes center stage at this layer. In contrast to software defined networking (SDN), which deals with the forwarding of individual packets throughout the network infrastructure, ADN works with data and gives applications the ability to dynamically adapt to their networking environment in order to optimize their performance. ADN is based on a feedback loop principle, which it leverages to monitor, analyze, and orchestrate infrastructure capacity and configuration in order to continuously adapt applications to facilitate optimum performance [11]. Thus, in our proposed model, ADN services monitor ADN-enabled applications in the cloud applications specific layer (CASL) and dynamically orchestrate (activate, deactivate, and tune) the services provided at the VOIL as it endeavors to achieve and maintain the performance targets of the associated application. For example, if the capacity links in the infrastructure are over-stretched and on the verge of causing imminent impairment to services and applications, ADN tools in the ADNL will be able to obviate this. At this layer, processes and agents such as those proposed in the GerNU project [12] could be modified and deployed to ensure optimum application performance and compliance with predefined service level agreements (SLAs).

2.4 Layer 4: Cloud Applications Specific Layer (CASL)

At the cloud applications specific layer (CASL), it is envisioned that applications will be able to operate without consideration for the disparate cloud platforms on which they are running. Thus, in the CASL *(pun intended)*, applications should be able to operate with impunity in pursuit of their various performance targets.

3 Prototype Implementation and Results

As a first step towards the realization of our user deployable autonomic multi-cloud overlay infrastructure based on the proposed autonomic multi-cloud model, we developed both a user-deployable virtual multi-cloud overlay infrastructure using virtual distributed Ethernet (VDE) [13] and an automated cluster deployment tool. We discuss these developments and our preliminary results below.

3.1 Overlay infrastructure implementation

We implemented the virtual infrastructure depicted in Fig. 2 using VDE as the virtual networking utility and Python Fabric [14], which provides a basic suite of operations for executing local and remote shell commands and uploading/downloading files, for system configuration. VDE has been used to implement virtual network laboratories such as Marionnet [15] and to build private networks for clusters of nodes in the Eucalyptus private cloud platform [16]. It is an open-source OSI Layer 2 virtual network software tool that can run on both virtual and physical machines and is able to forward, dispatch, and route plain Ethernet packets. VDE has the same structure as modern Ethernet networks and its main components are managed switches, wires (any tool capable of transferring a stream connection, e.g., SSH), and plugs (programs connected to a switch to convert all the traffic to a standard stream connection). A VDE cable comprises a wire with a plug at either end and is used to interconnect two VDE switches.

To implement the virtual overlay infrastructure depicted in Fig. 2, we deployed a VDE switch on the tap0 network interface of each VM (and physical computer) and connected the switches using SSH. As most cloud platforms use Class A private addressing (i.e., 10.x.x.x) on their eth0 network interfaces and small private networks use Class C private addressing (i.e., 192.168.x.x), to avoid confusion, we opted to use Class B private addressing (i.e., 172.16.x.x to 172.31.x.x) in our overlay infrastructure.

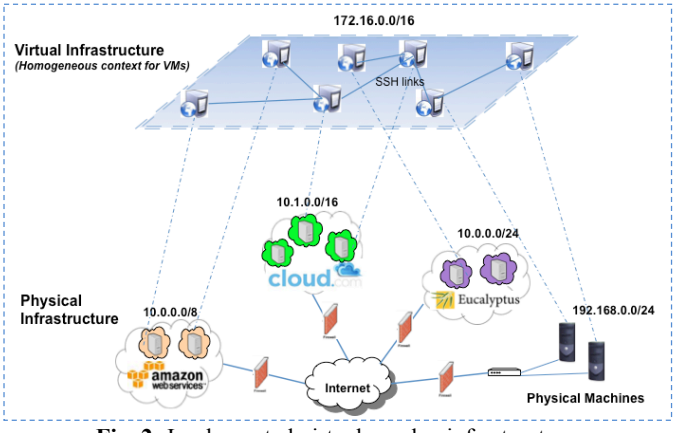

Fig. 2: Implemented virtual overlay infrastructure

3.2 Automated MPI cluster deployment

To enable automated cluster deployments on the infrastructure, we created a tool in Python that deploys and configures master and slave nodes using prebuilt CloudStack [31] templates made from Ubuntu 12.10 64-bit servers, with MPICH2 [32] installed.

3.2.1 Experimental cluster configurations

To get an idea of the performance of our multi-cloud cluster, we compared the performance of a small six-machine cluster for the three configurations illustrated in Figs. 3 to 5. For Configuration 1 (Fig. 3), the cluster was deployed on a conventional single-site CloudStack network (i.e., without the virtual overlay infrastructure). For Configuration 2 (Fig. 4), the cluster was deployed on the virtual overlay infrastructure across two CloudStack networks. For Configuration 3 (Fig. 5), the cluster was deployed on the virtual overlay infrastructure, this time comprising two CloudStack networks, two AWS [33] regions, and a physical machine.

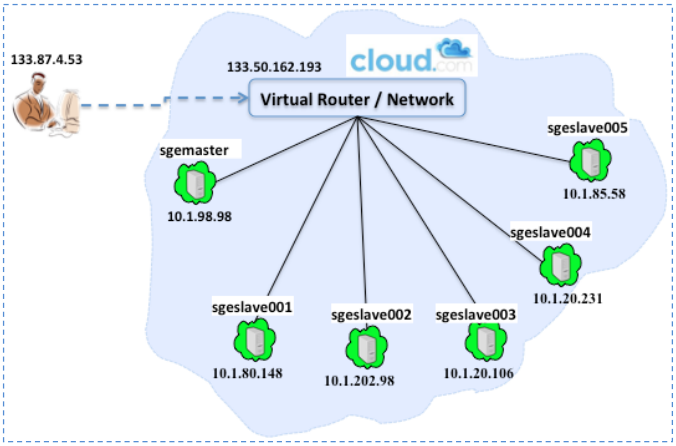

Fig. 3: Conventional single-site CloudStack cluster—No VDE *(Configuration 1)*

3.2.2 Simple genome sequencing experimental results

Executing the ClustalW-MPI alignment sequence tool [17] on three files across the cluster in each of the three configurations gave us the results shown in Table 1. For the largest file, the transfer time for Configuration 2 was almost twice that of Configuration 1, while that of Configuration 3 was almost 30 times that of Configuration 1. For the smallest file, however, Configurations 2 and 3 had very close transfer times. Large time differences between Configuration 1 and Configuration 3 were also evident in the sequence alignment times. We believe that the large time differentials are due to the fact that Configuration 3 comprised three nodes that were connected to the master via slow interconnection (Internet) links. To test the speed of the links, we conducted bandwidth measurements, which we discuss in Section 3.3.

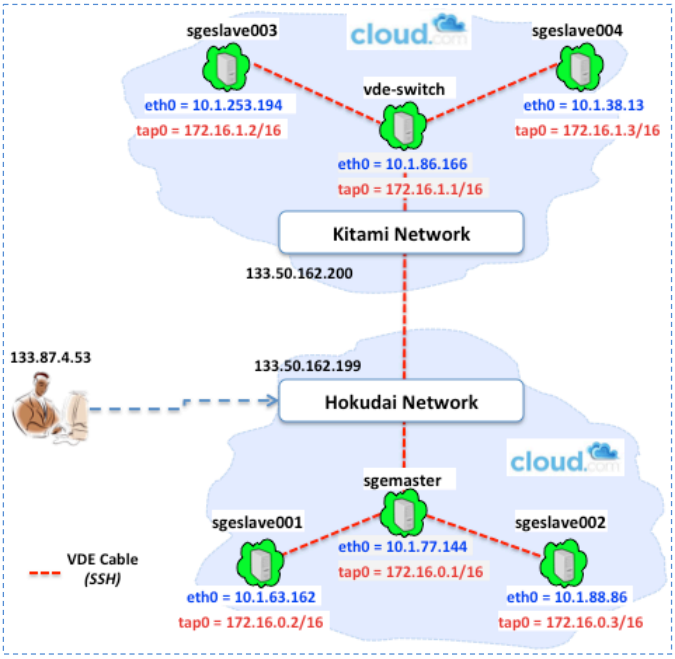

Fig. 4: Cluster across two CloudStack networks—Using VDE *(Configuration 2)*

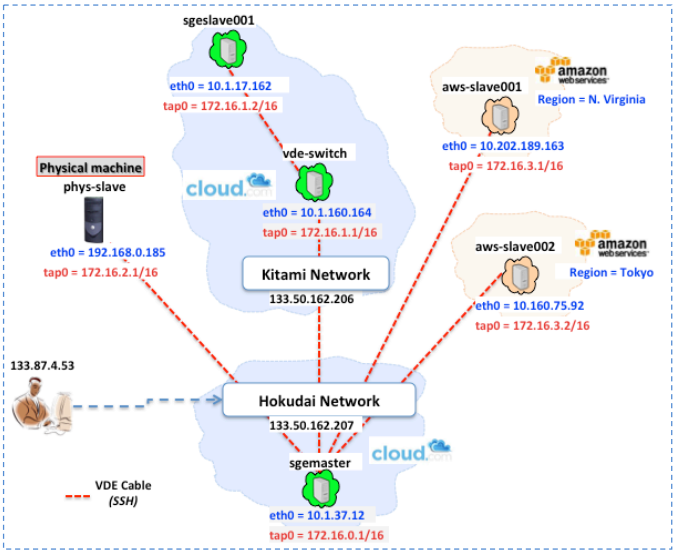

Fig. 5: Cluster across two CloudStack networks, two AWS regions, and one physical machine—Using VDE *(Configuration 3)*

Table 1: Simple genome sequencing experimental results

Cluster configuration	File number	File size	Number of groups	Time to transfer to slave nodes (s)	Time for pairwise sequence alignments (s)	Time for multiple sequence alignments (s)
Configuration 1 (Single site—No VDE)	1	4 KB	4	1.19×10^{-4}	0.011	0.077
	2	37 KB	21	3.29×10^{-4}	2.491	2.306
	3	124.3 MB	1170	23.76	n/a	n/a
Configuration 2 (Two CS networks—VDE)	1	4 KB	4	24.59×10^{-4}	0.023	0.335
	2	37 KB	21	47.65×10^{-4}	2.554	3.941
	3	124.3 MB	1170	43.23	n/a	n/a
Configuration 3 (Two CS networks, 2 AWS regions, 1 physical machine—VDE)	1	4 KB	4	21.98×10^{-4}	0.709	16.502
	2	37 KB	21	81.29×10^{-4}	4.046	71.507
	3	124.3 MB	1170	642.20	n/a	n/a

3.3 Bandwidth measurements

Using Iperf [18] with its default TCP window size of 8 KB, we conducted bandwidth tests between sgemaster and each of the slave nodes. The network configuration used is depicted in Fig. 6. The resultant bandwidth determined for the link between sgemaster and each slave node is shown in Fig. 7. Using the bandwidth obtained on eth0 as the standard (i.e., 207 Mbps), we computed the loss in bandwidth across the virtual infrastructure. The results obtained indicate the following:

1. The use of the virtual infrastructure results in a 6% decrease in bandwidth.
2. The second virtual router (Kitami Network) introduces an additional 6% decrease in bandwidth, resulting in a total bandwidth decrease of 12% from sgemaster to vde-switch.
3. Using node vde-switch as a proxy for the master to communicate with the slaves across the second network results in an overall loss in bandwidth of 43%. (This implies that a direct connection from the switch on each slave to the switch on the master is better.)
4. The links created across the Internet are (as expected) much slower than those inside the data center. This fact is reflected in the 99% and 92% bandwidth losses between sgemaster and euca-slave (in the Eucalyptus Community

Cloud (ECC) [34]) and sgemaster and aws-slave (AWS Tokyo region), respectively.

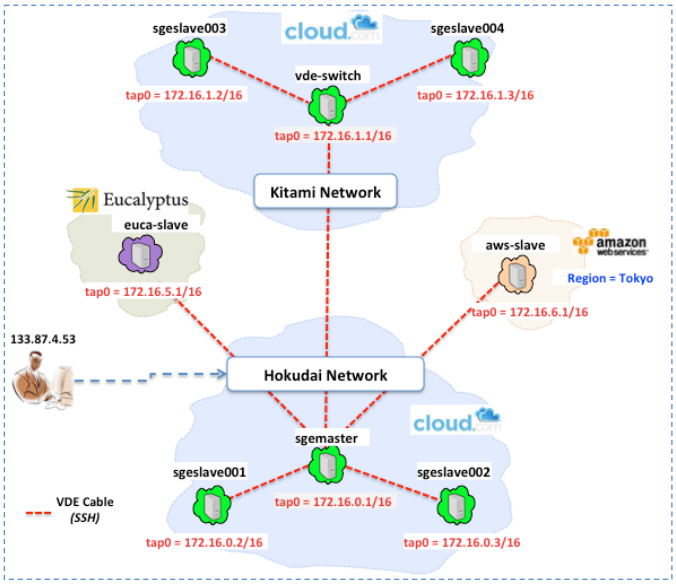

Fig. 6: Configuration used in bandwidth tests

4 Discussions and Future Work

It is obvious that in order to implement the kind of infrastructure proposed above the bandwidth disparities reported in this paper will have to be taken into account and techniques implemented to lessen their impact on the performance of the applications implemented on the virtual overlay infrastructure. The implementation of point-to-point dedicated links such as SINET [19] could be a solution for some inter-cloud links. However, as the idea is to facilitate connectivity across any available cloud system to which a user has access, other means of optimizing the user-level connections among cloud systems across the Internet are also imperative. For some applications, establishment of a primary link and the provision of a secondary link/route may be necessary. This can easily be done with VDE, as any switch on any node can be connected to a multiplicity of other nodes; thereby, providing the ability for the creation of alternate links/route. In fact, we plan to explore the performance impact of various topologies, such as star-ring hybrid and star-hypercube hybrid.

Even though our preliminary investigation indicates only a 6% fall in bandwidth due to the use of VDE, further investigations need to be done to see how the actual overlay infrastructure itself affects the performance of applications. Further, we intend to evaluate various other networking technologies to determine how well they cope with the rigors of a system such as this, and whether they can be combined and/or extended. Among these are ViNe [20], which uses user-level virtual routers (VMs on which ViNe processing software is installed and configured) to dictate overlay network communication by acting as gateways for nodes that do not run the ViNe software; VNET/U [21], which, in addition to facilitating overlay networking, may be able to provide ADN-related services such as monitoring of application communication and computation behavior [22]; and N2N [35], which uses P2P principles and the concept of edge nodes and super nodes (which relay packets across NAT boundaries for edge nodes) to facilitate overlay networks. Integration of the high-performance message passing protocol Open-MX [23] and user mode Linux related technologies such as VNX [24], Netkit [25], and Cloonix [26] also present interesting possibilities.

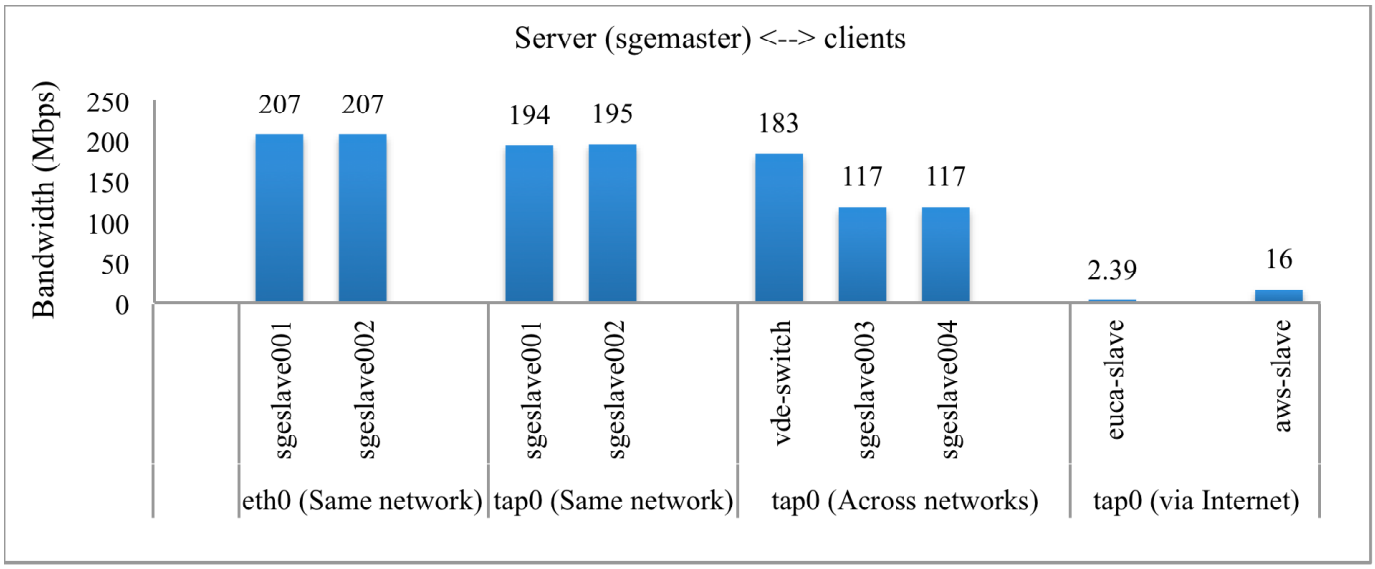

Fig. 7: Bandwidth from sgemaster to each client for the configuration in Fig. 6

Smith [27] enumerated a number of problems that may beset applications in single-site and federated clouds, such as the possibility of traffic impairment when multiple providers are utilized in order to reduce risks and difficulty avoiding network latency and bottlenecks. Our ultimate aim is to obviate such problems by applying the appropriate technology (or optimized mix of technologies) at each layer of our model. In [28], Friedman touched on the increasing trend of applications integrating across clouds, and states that *"This presents an impedance mismatch where a highly distributed and concurrent application must talk across a narrow 1-1 link with another highly distributed and concurrent application."* Some actual application challenges in the scientific arena, including the fact that application failures are prevalent in federated clouds, are also outlined by Deelman et al. [2]. We believe that the creation of an overlay infrastructure such as ours, in which applications are treated as first-class entities (i.e., ADN) and thus have the ability to autonomously act to fulfill specified requirements, can obviate this impending impedance mismatch.

The infrastructure proposed in this paper is being developed as part of the Simple Heterogeneous INter-CLoud Manager (SHINCLOM) project at Hokkaido University, Japan. In this project, we have already implemented a web interface that provides simple functions such as registration of cloud credentials in the content management system Drupal [29], which facilitates the development of the various sections of a cloud management system as modules that can be easily integrated [30].

5 Conclusion

In this paper, we outlined our proposal for the development of a user-centric multi-cloud overlay infrastructure based on a proposed layered autonomic multi-cloud model. Experimental measurements conducted on a crude prototype implementation of the system indicate that the system is feasible but more work needs to be done to improve the bandwidth of the inter-cloud links across the Internet.

6 Acknowledgements

This work utilizes the Hokkaido University Academic Cloud, Information Initiative Center, Hokkaido University, Sapporo, Japan and is supported in part by the CSI fund, National Institute of Informatics, JAPAN.

7 References

[1] P. Vicat-Blanc. "Harmony in the cloud." http://www.sdnzone.com/topics/software-defined-network/articles/332443-harmony-the-cloud.htm, April 1, 2013. Last accessed: April 25, 2013.

[2] E. Deelman, G. Juve, and G.B. Berriman. "Using clouds for science, is it just kicking the can down the road?" CLOSER 2012, pp. 127-134.

[3] J. McKendrick. "Cloud computing's vendor lock-in problem: Why the industry is taking a step backward." http://www.forbes.com/sites/joemckendrick/2011/11/20/cloud-computings-vendor-lock-in-problem-why-the-industry-is-taking-a-step-backwards/, November 20, 2011. Last accessed: April 25, 2013.

[4] D. Petcu. "Portability and interoperability between clouds: Challenges and case study." Towards a Service-Based Internet, LNCS, Springer, vol. 6994, pp. 62–74, 2011.

[5] K. Keahey, M. Tsugawa, A. Matsunaga, and J. A. Fortes. "Sky computing." IEEE Internet Computing, vol. 13, no. 5, pp. 43–51, 2009.

[6] Apache LibCloud: A unified interface to the cloud. http://libcloud.apache.org/. Last accessed: April 25, 2013.

[7] DeltaCloud. http://deltacloud.apache.org/. Last accessed: April 25, 2013.

[8] J. Bresnahan, T. Freeman, D. LaBissoniere, and K. Keahey. "Managing appliance launches in infrastructure clouds." Teragrid Conference, 2011.

[9] G. Juve and E. Deelman. "Automating application deployment in infrastructure clouds." CloudCom 2011.

[10] E. Afgan, K. Skala, D. Davidovic, T. Lipic, and I. Sovic. "CloudMan as a tool execution framework for the cloud." MIPRO 2012, pp. 437–441, May 21–25, 2012.

[11] Lyatiss whitepaper. "Application defined networking: The missing link for optimal cloud application performance and agility." www.becreative.ca/lyatiss/docs/Whitepaper-ADN.pdf, 2013. Last accessed: April 25, 2013.

[12] H.P. Borges, B.R. Schulze, J.N. Souza, and A.R. Mury. "Automatic services instantiation based on a process specification." Journal of network and computer applications, 2012

[13] R. Davoli. "VDE: Virtual distributed ethernet." TRIDENTCOM'05, pp. 213–220, 2005.

[14] Python Fabric. http://docs.fabfile.org/. Last accessed: April 25, 2013.

[15] J.-V. Loddo and L. Saiu. "How to implement a virtual network laboratory in six months and be happy." In ACM SIGPLAN Workshop on ML. ACM Press, 2007.

[16] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. "The Eucalyptus

open-source cloud-computing system." CCGRID'09, Shanghai, China, pp.124-131, May 2009.

[17] K.-B. Li. "ClustalW-MPI: ClustalW analysis using distributed and parallel computing." Bioinformatics, pp. 585-586, 2003.

[18] Iperf. http://openmaniak.com/iperf.php. Last accessed: April 25, 2013.

[19] SINET. http://www.sinet.ad.jp/index_en.html?lang=english. Last accessed: April 25, 2013.

[20] M. Tsugawa, A. Matsunaga, and J. Fortes. "User-level virtual network support for sky computing." e-Science'09, pp. 72–79, 2009.

[21] A. Sundararaj and P. Dinda. "Towards virtual networks for virtual machine grid computing." Proc. 3rd USENIX Virtual Machine Research And Technology Symposium (VM 2004), May 2004.

[22] A. Gupta and P.A. Dinda. "Inferring the topology and traffic load of parallel programs running in a virtual machine environment." Proc. 10th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP), June 2004.

[23] Open-MX: Myrinet express over generic ethernet hardware. http://open-mx.gforge.inria.fr/. Last accessed: April 25, 2013.

[24] VNX: Virtual Networks over linuX. http://web.dit.upm.es/vnxwiki/index.php/Main_Page. Last accessed: April 25, 2013.

[25] Netkit: The poor man's system to experiment computer networking. http://wiki.netkit.org/index.php/Main_Page. Last accessed: April 25, 2013.

[26] Cloonix: Dynamical topology virtual networks. http://clownix.net/. Last accessed: April 25, 2013.

[27] M. Smith. "Network and application performance in cloud." http://erpcloudnews.com/2013/04/network-and-application-performance-in-cloud/. Last accessed: April 25, 2013.

[28] J. Friedman. "Patterns for programming in the clouds." http://www.cs.york.ac.uk/library/proj_files/2010/EngDInd/julz/litreview-final-31may.pdf. Last accessed: April 25, 2013.

[29] Drupal. http://drupal.org/. Last accessed: April 25, 2013.

[30] Y. Naoi. "Clanavi: How to manage your cloud by Drupal." Bay Area Drupal Camp 2010, November 13, 2010.

[31] CloudStack open source computing. http://cloudstack.apache.org/. Last accessed: May 31, 2013.

[32] MPICH: High-performance portable MPI. http://www.mpich.org/. Last accessed: May 31, 2013.

[33] Amazon web services. http://aws.amazon.com/. Last accessed: May 31, 2013.

[34] Eucalyptus community cloud (ECC). http://www.eucalyptus.com/eucalyptus-cloud/community-cloud. Last accessed: May 31, 2013.

[35] N2N: A layer two peer-to-peer VPN. http://www.ntop.org/products/n2n/. Last accessed: May 31, 2013.