Towards a Ubiquitous Cloud Computing Infrastructure

Jacobus Van der Merwe, K.K. Ramakrishnan, Michael Fairchild, Ashley Flavel,
Joe Houle, H. Andres Lagar-Cavilla, John Mulligan
AT&T

***Abstract*—In this extended abstract we explore the architectural components of a *Cloud Control Architecture* with the aid of a number of cloud computing use cases. We specifically consider cloudbursting and follow-the-sun and focus on the mechanisms and user/provider interactions that would make these scenarios real. We are particularly concerned with the coordination of cloud and networking resources and mechanisms that would be applicable to cloud providers that are also network service providers.**

I. Introduction

Cloud computing is rapidly gaining acceptance as a highly flexible means of managing resources. This is certainly true in what has become the cloud computing mainstay of providing resources for Internet facing services, but is also true within enterprise network environments where so-called ”private-cloud” infrastructures are increasingly being utilized. Such ubiquity of cloud resources hold the promise of enabling new service models, where resources are seamlessly utilized at the time and location that are best suited to the needs of the current workload, while at the same time optimizing business objectives such as minimizing cost and maintaining service quality levels.

In this paper we explore the architectural components required to realize such a *ubiquitous cloud computing infrastructure*. The architecture that transpires includes: A *service abstraction layer* through which users access cloud services and that captures the service logic of such services. An *intelligence layer* which provides information regarding the infrastructure as a whole to inform decisions concerning the placement, migration and consolidation of resources. Acting on user requests and information from the intelligence layer is an *orchestration layer* which utilizes control and data plane *mechanisms* to coordinate the manipulation of resources in a seamless manner.

While not all cloud service providers are also network service providers, our specific focus in this paper is the case where cloud and network services are bundled together to provide unique cloud related services. It is desirable for storage and compute resources in the cloud to be seamlessly connected to an enterprise, so that these appear as if they are secure, local resources within the enterprise local area network (LAN). Transparent LAN services over the wide area, using a virtual private LAN services (VPLS) that uses IP/MPLS as the underlying connectivity over the WAN is intended to provide a multipoint-to-multipoint Ethernet service between multiple enterprise sites in geographically dispersed locations. It enables sites in different points on a WAN to appear to be on the same Ethernet-based Local Area Network. Thus cloud computing across data centers (i.e., hybrid clouds between the enterprise and the provider, or pooling of resources across cloud sites) allows both the enterprise and the cloud provider to gain advantages through statistical multiplexing/sharing of computing and storage resources.

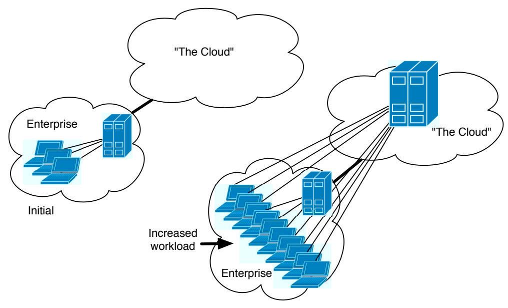

Fig. 1. Conceptual view of “cloudbursting”.

We attempt to ground our exploration in reality by considering a number of specific use cases. Specifically in Section II we consider two scenarios involving the interaction between enterprise datacenters and resources in a cloud provider datacenter. In Section III we generalize our observations to present a Cloud Control Architecture. We describe the control and data plane mechanisms in more detail in Section IV before concluding.

II. Scenarios

The two use cases we explore in this paper are the so called *cloudbursting* and *follow-the-sun* scenarios depicted respectively in Figures 1 and 2. Cloudbursting in essence means that resources in an enterprise network are augmented with resources in “the cloud” during times when insufficient resources are available in the enterprise network. Follow-the-sun, on the other hand, suggests that the cloud resources utilized to fulfill a certain function, dynamically move around the globe to where that function is to be performed at a particular time of day. The apparent simplicity of these concepts belies both the underlying challenges to make it a reality as well as the true potential of a ubiquitous cloud environment. We discuss these use cases in more detail below and specifically consider the following questions:

- What types of workload can utilize the different scenarios?

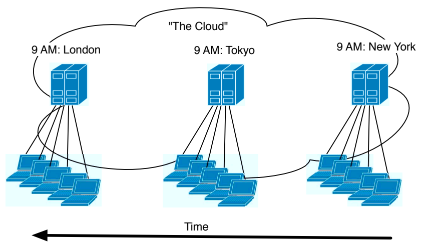

Fig. 2. Conceptual view of "follow-the-sun".

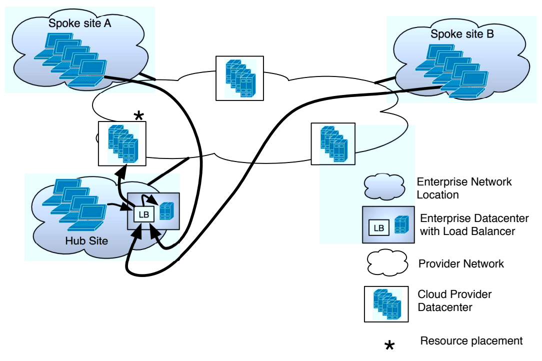
Spoke site A
Spoke site B
Hub Site
LB
Enterprise Network Location
Enterprise Datacenter with Load Balancer
Provider Network
Cloud Provider Datacenter
Resource placement

Fig. 3. Cloudbursting with local load balancing.

- How should the cloud, its resources and the services it offers be presented to users, and what are the tradeoffs between user and provider control?
- What intelligence and mechanisms are required by a service provider to offer the envisioned services?

A. *Cloudbursting*

Figure 3 illustrates a more complex (and perhaps a more realistic) cloudbursting scenario. First, resources in "the cloud" would typically be located in a set of *cloud provider datacenters*. With our focus on cloud services offered by network service providers, these datacenters are typically geographically distributed throughout a *provider network*. While single location enterprises certainly exist, it is common today for *enterprise networks* to be distributed across different geographic locations, as the example in figure 3 illustrates. Finally, enterprises often have their own *enterprise data centers*, which might be located at a main enterprise site (the "hub site" in Figure 3), with remote locations ("spoke sites") accessing resources in that data center via network services provided by the network provider, e.g., via a virtual private network (VPN). With this setup, cloudbursting involves the enterprise using resources in the cloud, i.e., resources in a cloud provider data center, when not enough resources are available in the enterprise data center.

Workloads: Clearly cloudbursting may not be ideally suited to all workloads and/or applications. An ideal cloudbursting workload would be applications that are: (i) readily scalable with additional resources and (ii) where the overhead involved in making the cloud-based resources ready to process the workload is relatively small. Examples include computationally intensive workloads such as biological sequencing, financial analysis and graphics rendering [1], or the distribution of rich media content within the enterprise [2]. On the other hand, an overloaded database backend system in a multi-tier application (in the enterprise data center) would be a less ideal application for cloudbursting as replicating a large database onto resources in the cloud provider data center would involve significant overhead and startup costs.

Service abstraction: This example serves to illustrate a number of implied requirements for how the service is presented to users and the underlying cloud architecture that enables it. In Figure 3 the enterprise in question is assumed to receive cloud resources from the provider cloud datacenter that is closely located to the enterprise hub site. Assuming this is desirable, the question is how this placement decision is achieved in practice. I.e., who makes the decision (the enterprise user or the cloud provider), and what information is utilized to make the decision?

For example, the provider cloud application programming interface (API) might expose to the enterprise the fact that there are multiple provider datacenters in different geographic locations and leave it up to the users to select the one that best fits their needs. I.e., the user might use information about the location of its enterprise locations and how traffic flows between those locations to decide which cloud data center to use. This approach is fairly complex from a user's perspective, and while it simplifies some service aspects from a cloud provider perspective, it reduces provider flexibility. Specifically, with placement control in the hands of the user, the provider would not have the flexibility to perform placement based on capacity or availability contraints.

Mechanisms: Assuming that the application and workload in question would fit a cloud bursting scenario, the next concern is how the enterprise infrastructure would avail itself of the availability of these cloud-based resources and how the resources would be reached and utilized. In the case of an enterprise VPN such as the one depicted in Figure 3, reachability to the provider cloud resources can be readily provided by dynamically extending the enterprise VPN into the provider cloud environment [3]. Once resources in the provider cloud are reachable, one way in which the enterprise application can effectively utilize them is by making use of a *local load balancing* element which can be made aware of the availability of new resources in the provider cloud. In order to transparently integrate provider cloud resources into the enterprise network, it is essential that this process be coordinated. I.e., as more resources are being made available in the provider cloud, the local load balancer should automatically be made aware of this fact. As shown in Figure 3, the implication of such an approach is that workload "request traffic" has to travel all the way to the site where the load balancer is located, only to be immediately shipped off to the cloud via the same network path (if the request will be served from the cloud resources). For workloads where network latency is not a concern this will not present a problem. For many workloads the additional latency of this network path might, however, be problematic. It is also possible that a centralized local load balancer would become a bottleneck under heavy load.

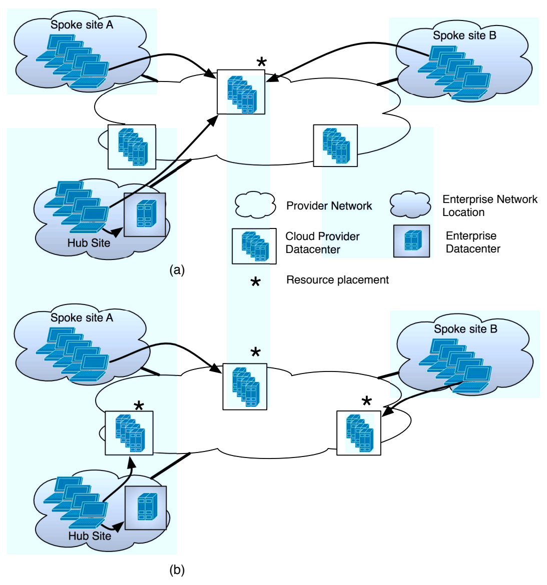

Fig. 4. Cloudbursting with network support.

An alternative approach is depicted in Figure 4. Here the assumption is that the cloud provider decides the placement of cloud resources on behalf of the user. The provider placement decision could utilize information directly provided by the user, and/or network derived intelligence that could be used to determine the best placement strategy. For example, Figure 4 (a) depicts a scenario where network intelligence suggests that the enterprise spoke locations are responsible for most of the load, so that resource placement at the provider data center closest to these sites would be optimal. Figure 4 (b) suggests a different realization where placement of resources occurs at a number of different provider datacenter locations, so as to minimize the distance between the provider data centers and the enterprise sites they serve.

Realization of the approaches depicted in Figure 4 once again has a number of architectural implications. First, these approaches assume an alternative mechanism for enabling the enterprise to use the additional cloud-based resources. For example, request routing mechanisms can be used to balance the load on the pool of available resources [4]. Alternatively, load-aware anycast mechanisms can be utilized [5]. Like the local load balancing approach of Figure 3, these mechanisms have to be controlled in concert with the management of resources in the provider cloud. I.e., when more resources are being made available at a particular provider data center, the load balancing mechanisms need to be notified so that these resources can be effectively utilized.

As before, this scenario presents a tradeoff concerning the sharing of responsibilities and control between the user and the provider. The premise of the cloudbursting scenario is that a lack of resources within the enterprise environment serves as the trigger to activate bursting into the provider cloud. This implies that the user would want to maintain control over the mechanism that determines how much cloud based resources are being utilized. This requirement would have to be accommodated if more functionality is shifted to the cloud provider.

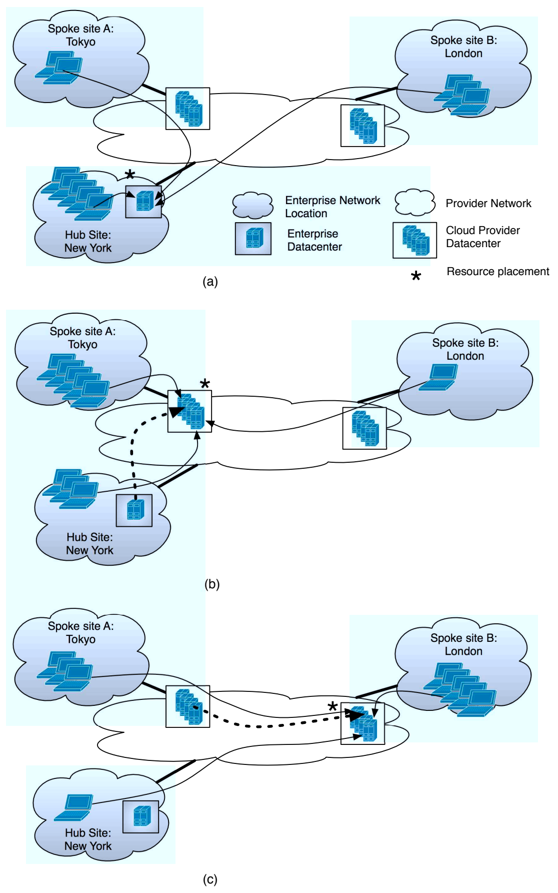

Fig. 5. Follow-the-sun

B. Follow-the-sun

In Figure 5, we modify our running example to explore the follow-the-sun scenario in more detail. The underlying assumption for this use case is that the geographic location of the primary workload driver shifts during the course of the day, and further that it is beneficial for these workload drivers to be in close proximity to the resources they operate on. In Figure 5 we further assume that this resource migration takes place from an enterprise datacenter to a cloud provider datacenter (Figure 5 (a) and (b)), from one cloud provider datacenter to another (Figure 5 (b) and (c)) and finally back to enterprise datacenter (implied between Figure 5 (c) and (a)).
Workloads: An example workload that would benefit from follow-the-sun is a multi-tier application, i.e., web server, application server and database, where the complete stack can be migrated from datacenter to datacenter. The expected benefit from a follow-the-sun scenario is reduced network latency, which translates to improved performance for users and applications. However, the overhead involved in realizing this scenario might be significant. For example, if the database in question is both large and changes rapidly over the course of the day, the performance benefit would have to be weighed against this overhead.
Service abstraction: Again the question is how this kind of

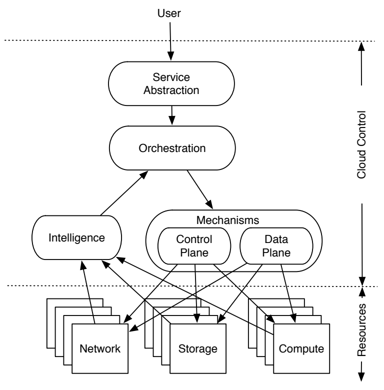

Fig. 6. Cloud Control Architecture

service would be presented to users. Specifically the cloud provider service abstraction needs to allow for our assumed use case of moving between private and provider clouds in addition to moving between provider cloud datacenters. Dealing with the latter case first, the API to the provider cloud need to allow for a user to request the migration of the complete multi-tier stack from a provider datacenter in one geographic location to another. This might be realized either in an on-demand fashion, or as a pre-scheduled task. In both cases the user will need a way to specify the resources that need to be migrated (e.g., the virtual machines and disk images). In the case of migrating between the private cloud and the provider cloud, similar information will have to be furnished to the provider cloud API, except that in this case the source or target would be the enterprise data center.

Mechanisms: Several mechanisms are required to realize this scenario. First, a mechanism is required to provide connectivity between the enterprise network and the provider data center. Providing such connectivity via layer-2 VPN technologies is attractive as it enables transparent resource migration. Second, since we assume the complete multi-tier stack is migrated, mechanisms are required to migrate both storage and compute resources [6]. In the ideal case this type of migration would happen while the application keeps operating, i.e., live migration [7]. Live migration would require compatibility between the underlying virtualization platforms, and requires the VM(s) to keep their current IP addresses. For this reason the use of layer-2 technologies is preferred. However, it is also possible to make use of layer-3 control plane mechanisms [8] or other layer-3 redirection mechanisms [4] to steer traffic towards the location of a migrated instance in a layer-3 environment. In a layer-3 migration scenario, the migration would, however, not be live or fully transparent as the new instance would have a different IP address.

III. Cloud Control Architecture

Ultimately cloud-based services involve providing users access to resources. Figure 6 depicts the components of a *Cloud Control Architecture* that emerged from the scenarios considered in Section II to enable this:

- *Service Abstraction*: This layer represents the means by which cloud services and service logic are presented to users. Of concern are somewhat pragmatic issues like what application programming interface (API) paradigms need to be supported as well as business/service concerns about how services and service logic are exposed to users through such APIs. Consider, for example, the tradeoff discussed in Section II between exposing provider datacenter locations to users to make their own placement decisions for cloudbursting, versus, the case where that functionality is performed by the provider and has to be exposed, in abstract form, to the user through an API. Further, of particular relevance for our focus of cloud providers that are also network providers, is how such networking functionality is exposed to users. (An aspect that is not very well developed in current cloud service abstractions and APIs.)
- *Orchestration*: This layer combines user requests from the service presentation/access layer with information provided by the intelligence layer to actualize user requests. For example the orchestration layer make decisions concerning initial placement location for user resource requests, resource allocation, resource adjustment and movement of resources. An example from the follow-the-sun scenario in Section II is making a dynamic placement decision for where migrated resources would be moved to, based on both the expected workload and the availability of resources in the provider datacenter.
- *Intelligence*: This layer involves intelligence derived from the actual resources being managed/utilized. This includes information about the cloud infrastructure, e.g., the availability and utilization of compute and storage resources, the geographic distribution of the offered workload, network availability and utilization between cloud provider data centers.
- *Mechanisms*: The realization of user requests is achieved by the orchestration layer through appropriate manipulation of control plane and data plane operations on the mechanisms layer. Control plane mechanisms include different approaches for resource discovery and load balancing. Data plane mechanisms include techniques such as virtual machine manipulation (e.g., cloning and migration), efficient storage access and replication, and network connectivity management. We consider these in more detail below.

The increased use of cloud technologies in enterprise networks, i.e., private clouds, suggest that similar cloud control plane architectures will be realized in both private and public clouds. These control architectures will likely start off with private clouds utilizing control APIs offered by public cloud providers.[1] However, as suggested by the discussion in

[1]For example, open source cloud stacks such as OpenNebula (www.opennebula.org) already provide the basic means to incorporate provider clouds with private instances of OpenNebula.

Section II, a richer federated interaction between provider and private cloud control architectures might be possible.

IV. Mechanisms

In this section we briefly elaborate on some of the data and control plane *mechanisms* that were mentioned in our use case discussion in Section II and constituted the mechanism layer in the cloud control architecture presented in Section III.

A. Data Plane

Data plane mechanisms relate to the efficient allocation, movement and interconnection of cloud and network resources and include:

VM Migration: Virtual machine migration is a well established approach to the movement of compute resources in a LAN environment [7]. Because it allows for the migration of operational VMs without significant application level impact, it is particularly attractive as a management mechanism in cloud environments. As we outline below, local area network (LAN) interconnect mechanisms allow for extending the use of these mechanisms into wide area network (WAN) environments. However, because of the impact of network latencies in a WAN environment care has to be taken to ensure VM migration mechanisms still perform efficiently [6]. This set of mechanisms is directly applicable to the follow-the-sun and related use cases.

VM Cloning: Virtual machine cloning is a method in which the concept of UNIX process forking is applied to an entire VM stack. Multiple replicas of a VM are created. These are logically complete and identical replicas, including the VM's disk, with the exception of an ID that allows each clone to identify itself and the programmer to reason about the result of cloning (much like the PID in process fork). The cloned VMs are transient in that once the task they were created to accomplish is finished, they are discarded and their state is completely eliminated, making it directly applicable to the cloudbursting use case.

SnowFlock [1] is an efficient implementation of the VM cloning paradigm. It recognizes that although VM clones are logical replicas of the entire VM, for many workloads only a small subset of the parent state is actually needed. SnowFlock thus propagates what is essentially the working set of the VM to the clones, achieving extremely high runtime and cloning performance: 32 clones in 32 different hosts are created under a second to complete parallel tasks such as sequence alignment with only a few seconds of overhead. SnowFlock's cloning techniques need to be adapted for a wide-area cloud scenario like the one we advocate, but the potential to achieve similarly high performance results and almost instantaneous clone creation is very promising.

LAN interconnect: Quickly and transparently utilizing computing and storage at one data center or another (whether in the enterprise or in the cloud) is desirable to break the boundaries between geographically separated data centers. A major contributor to transparency is to allow enterprises to connect to the cloud using Ethernet access, that allows transparent inclusion of cloud resources - both in terms of computing and storage. Of course, transparency achieved with a flat Ethernet network is inadequate due to scalability limitations as a result of the use of a large broadcast domain. Moreover, there is a need to partition resources between enterprises that use a common set of cloud resources. VLANs enabled partitioning of the LAN resources and isolation of access to end-systems within the enterprise. In a similar manner, VPLS allows the partitioning and isolation of resources over the WAN for each individual VPN [9]. VPLS takes the spanning-tree based bridging protocol for LANs that inherently takes advantage of broadcast to maintain a loop-free topology and scales it over a larger network. VPLS utilizes the scalability and resiliency of IP/MPLS to carry transparent LAN services by associating each VPN to a set of MPLS pseudo-wires over the WAN. Layer 2 VPNs and Layer 3 VPNs both can leverage BGP and include automatic discovery of resources within a VPN across sites and the capability to auto-provision MPLS connectivity. Both auto-discovery and auto-provisioning are especially critical in multipoint-to-multipoint VPNs such as VPLS and 2547 VPNs. By allowing enterprises to use Ethernet access, there is minimal configuration required to access provider cloud resources.

Storage replication: Storage (i.e., application data that is written to a file/disk, including databases) is an important component of the resources that are used in the cloud. Whenever we use the cloud resources, whether for the cloudbursting or the follow-the-sun usage scenario, it is important to consider the use of both compute and storage resources holistically. For many applications, it is necessary that compute resources and the application data be co-located for efficiency, latency and/or bandwidth considerations. Moving storage from either the enterprise data center (e.g., for the cloudbursting case) or between cloud data centers (e.g., follow-the-sun scenario) can take a significant amount of time. This time often dominates the total migration time, and can be considerably larger than the time for migrating VM state. It is also important to anticipate the need to move storage/application data to the appropriate cloud site, depending on each of the scenarios we described above. This implies that the data has to be replicated to the remote site while the application is still running at the original site. We thus observe that storage replication approaches are equally applicable for enabling applications to move to and move within the cloud.

Storage replication techniques can be broadly classified into two classes: (i) Synchronous replication: every data block written to a file at the local system is replicated to the remote location before the local write operation returns to the application. (ii) Asynchronous replication: in this case the local and remote copies of the file at the two systems may diverge. The amount of divergence between the local and remote systems is typically bounded by either a certain amount of data, or by a certain amount of time.

The consequence of using asynchronous replication is that there is a certain amount of inconsistency between the local and the remote site and this needs to me managed carefully. The data in a file or database in the cloud has to eventually be in synchrony with the local copy (in the enterprise or in the initial cloud site). However, synchronous copying to keep the cloud-based replica up to date may result in unacceptable user perceived latency. This is especially of concern when the cloud site to which the data is being replicated is far away (e.g., in the follow-the-sun application). One approach that addresses this problem is to use asynchronous replication whenever possible, and to finally switch over to synchronous replication at the last possible moment. The synchronous replication includes ensuring all pending writes that await transfer to the remote site from the asynchronous replication are completed as well. Thereby, we assure that the remote site is consistent with the initial site before the remote cloud resources take over as the primary site for running the application.

As described in [6], copying of the local storage subsystem and asynchronous replication of writes is initiated once the migration to the remote cloud site is initiated. Once the remote storage subsystem has been brought to a consistent state, the architecture switches to a synchronous replication scheme when the live migration of the VM's memory state is initiated. During the VM migration, disk updates are synchronously propagated to the remote disk to ensure consistency when the memory migration finishes and the VM becomes live on the remote host.

B. Control Plane

Control plane mechanisms relate to the manipulation of data plane mechanisms and to facilitate transparent access to cloud based resources. These mechanisms include the following:

Route control (dynamic VPN connectivity): Key to achieving transparency in the use of cloud resources across a variety of applications is to enable these resources to be brought into the enterprise's extended LAN dynamically (because resources get created and removed near instantaneously on an as-needed basis by enterprises) [3], [10]. Thus, there is a need to be able to setup VPLS instances on a dynamic and instantaneous basis. The use of a BGP based VPLS Layer-2 VPNs and the use of route control mechanisms [8] to do a dynamic mapping enables this. Specifically, in VPLS VPNs, membership to a specific VPN is determined by the exchange of network layer reachability information (NLRI) through the exchange of BGP messages. By manipulating this membership information by the cloud control architecture, in concert with the creation of cloud resources, enables VPLS VPNs to be dynamically extended at the same timescales at which cloud resources are being created.

Route control (load balancing): Another key requirement is to allow transparent access to resources in the provider cloud. In cases where multiple instances of essentially equivalent resources are available, layer-3 anycast routing can be utilized to allow endpoints to transparently reach the closest instance. By controlling such layer-3 routing based on the relative load on the different instances, allows this mechanism to be utilized to achieve load balancing between the different instances [5].

Request routing (load-balancing): More traditional request routing mechanisms can also be employed to allow transparent, load-balanced access to cloud resources [4]. DNS-based load balancing in this context might require very granular deployment of local DNS servers in order to ensure both accurate proximity-aware request routing. In a cloud environment, more dynamic allocation of resources (compared to traditional content and hosting environments), allows for an extra degree of freedom in terms of achieving load balancing goals.

V. Conclusion

In this paper we explored the interaction between private and public cloud infrastructures through a number of use cases. Our study suggests that the basic building blocks exist to enable the realization of sophisticated private/public cloud interactions. However, these building blocks need to be orchestrated by a holistic architecture that honors user requests while taking into account cloud and network conditions. Further, finding a service abstraction that can be presented to users and finds the right balance between user complexity, versus provider complexity, and between the exposing and hiding of information remains an open challenge.

References

[1] H. A. Lagar-Cavilla, J. A. Whitney, A. M. Scannell, P. Patchin, S. M. Rumble, E. de Lara, M. Brudno, and M. Satyanarayanan, "Snowflock: rapid virtual machine cloning for cloud computing," in *EuroSys '09: Proceedings of the 4th ACM European conference on Computer systems*. ACM, 2009, pp. 1–12.

[2] J. Van der Merwe, P. Gausman, C. Cranor, and R. Akhmarov, "Design, Implementation and Operation of a large Enterprise Content Distribution Network," September 2003, 8th International Workshop on Web Content Caching and Distribution.

[3] T. Wood, P. Shenoy, A. Gerber, K. Ramakrishnan, and J. Van der Merwe, "The Case for Enterprise-Ready Virtual Private Clouds," June 2009, workshop on Hot Topics in Cloud Computing, HotCloud'09.

[4] A. Barbir and B. Cain and F. Douglis and M. Green and M. Hofmann and R. Nair and D. Potter, and O. Spatscheck, "Known CN Request-Routing Mechanisms," RFC 3568, July 2003.

[5] H. A. Alzoubi, S. Lee, M. Rabinovich, O. Spatscheck, and J. V. der Merwe, "Anycast cdns revisited," in *Proceeding of WWW '08*. New York, NY, USA: ACM, 2008, pp. 277–286.

[6] T. Wood, P. Shenoy, K. Ramakrishnan, and J. Van der Merwe, "CloudNet: A Platform for Optimized WAN Migration of Virtual Machines," 2010, in submission.

[7] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of NSDI*, May 2005.

[8] J. E. Van der Merwe *et al.*, "Dynamic Connectivity Management with an Intelligent Route Service Control Point," Proceedings of ACM SIGCOMM INM, October 2006.

[9] K. Kompella and Y. Rekhter, "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling," Network Working Group, Request for Comments 4761, January 2007.

[10] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, and A. Saxena, "Intelligent Workload Factoring for A Hybrid Cloud Computing Model," in *IEEE 7th International Conference on Web Service (ICWS 2009)*, July 2009.