A New Approach to the Design of Flexible Cloud Management Platforms

Juliano Araujo Wickboldt
Lisandro Zambenedetti Granville
Federal University of Rio Grande do Sul, Brazil
{jwickboldt, granville}@inf.ufrgs.br

Fabian Schneider
Dominique Dudkowski, Marcus Brunner
NEC Laboratories Europe, Germany
{Fabian.Schneider, Dominique.Dudkowski, brunner}@neclab.eu

***Abstract*—Current cloud management platforms have been designed to deal mainly with computing and storage resources. Networking, on the other hand, is often focused only on ensuring basic connectivity between virtual machines. That means, advanced requirements, such as delay and bandwidth guarantees or handing of network control to the customer, are not supported in today's platforms. Another important shortcoming is that resource management strategies are mostly implemented as part of the core of platforms, leaving little or no room for personalization by the operator or the customer. Therefore, in this paper we present the building blocks of a new conceptual architecture of a cloud platform aiming to add advanced yet robust network configuration support and more flexibility at the core of the platform to better fit the needs of each cloud environment.**

I. Introduction

Most of the open source platforms available for cloud infrastructure management (*e. g.*, , OpenNebula [1], Eucalyptus [2], OpenStack [3]) separate management concerns into computing, storage, and networking (see Section II). Computing and storage management are fairly consolidated and respectively related to management of virtual machines (VMs) and virtual storage volumes. These two management concerns are basically responsible for handling physical resources, such as processing power, volatile and persistent memory, in each node of a cloud environment.

On the other hand, network management in clouds is still in a very early stage. Typically, this management concern encompasses enabling proper communication between virtual components, such as virtual interfaces of VMs, possibly also virtual switches and routers. Differently from traditional network management in physical networks, it is important to keep in mind that these virtual resources can be created, destroyed, or migrated in a very dynamic fashion. Most of the current cloud platforms rely on somewhat old-fashioned tools for network related tasks, using for example, DHCP servers, manual VLAN configuration, NAT or forwarding rules using iptables. In this paper we intend take a new approach to this matter by adding robust support for network management in our cloud management platform already from the design phase.

From the very definition of what cloud computing encompasses [4], cloud platforms were always conceived as centralized controlling systems; operating like a black-box where one describes rather simply a set of resource needs and gets back such resources, somehow provisioned by the cloud. Two main shortcomings may be pointed out in this approach: (*i*) resource requirements are often expressed in a too simplistic manner; and (*ii*) few opportunities are presented to cloud operators to tweak resource allocation strategies. As a result, a whole class of applications that could benefit from cloud environments, cannot do so because of poor support for specifying their strict requirements. This statement holds specially in the case of highly distributed and network intensive applications and affects more directly the cloud customer. From the point of view of the cloud provider, influencing resource allocation strategies, and therefore optimizing the overall use of physical resources, provides an opportunity to reduce investments and operational cost (*i. e.*, CAPEX and OPEX). Consequently, we want to address these points by designing our architecture with flexibility for the operator and even the customer in mind.

Based on the shortcomings just exposed, in this paper we introduce the building blocks of a new conceptual architecture for cloud platforms focusing mainly into the following three aspects:

- Robust support for networking employing modern paradigms and technologies (*e. g.*, , software defined networks (SDN));
- Integrated support for richer specification of virtual infrastructures (containing computing, storage, and network altogether) and application-specific requirements (*e. g.*, , elasticity rules);
- Flexibility at the core of the platform in such a way that that operators can describe and personalize resource allocation and optimization strategies.

II. Background & Related Work

Note that this research is carried out in the context of the SAIL EU project [5], in the Cloud Networking (CloNe) work package [6]. Thus, some of the concepts presented in this paper are either brought from or inspired by the project's guidelines. In this section we first discuss the typical life cycle of a virtual infrastructure before reviewing cloud management approaches and solutions.

978-3-901882-48-7

On a high level we consider a simple Infrastructure as a Service (IaaS) model, where a user is interested in "buying" infrastructure to deploy a service or application on top of a cloud providers resources. More complex business models have been discussed in other work [7].

A. Life cycle of a virtual Infrastructure

Request: Initially, the user sends a specification of a set of resources to the cloud provider. This specification should reflect the users service/application requirements as much as possible. While sometimes the requirements are complex and difficult to represent, it is outside the scope of our scope. Yet, few standards are under way that can be used for describing virtual infrastructures, such as Virtual Infrastructure Description Language (VXDL) [8], Open Cloud Networking Interface (OCNI) [9], and Open Virtualization Format (OVF) [10].

Parsing: Subsequently, the initial specification is parsed and interpreted by the cloud infrastructure management system, generating an internal representation, which we call Hybrid Flash Slice (HyFS). The slices are "hybrid" because they may include different kinds of resources (*i. e.*, computing, storage, and network), as opposed to, *e. g.*, only network or only computing. By "flash" we refer to the dynamic nature of the slice, in which a slice can be created, destroyed, expanded, and shrunk in a dynamic fashion on a time scale that is comparable to the scales found in today's compute-centric cloud platforms. It might be the case that a single cloud provider cannot fulfill the complete request and multiple cloud providers need to join forces. However we leave such scenarios for future work as such a split will add complexity in terms of interfaces between cloud providers rather than technical complexity to our approach.

Resource Allocation: In order to map the HyFS representation to the available resources the cloud management system will implement a set of algorithms, capable of allocating the correct amount of physical resources (*e. g.*, processors, memory, and network links) and assigning virtual resources on top of those (*e. g.*, virtual machines, virtual links, and virtual switches). Cloud providers will want to develop their own algorithms or at least adapt those existing ones to better fit the needs of each cloud environment.

Adaption: When a HyFS is deployed the cloud provider still needs to guarantee the agreed service levels. Therefore, deployed slices should be monitored on two levels: *(i)* virtual infrastructure level and *(ii)* application level. For the former the cloud provider can easily instrument its infrastructure. Regarding application monitoring, the users should provide access to some Key Performance Indicators (KPIs). Both application and virtual infrastructure level monitoring are vital to enable elasticity of applications in the cloud. In other words, monitored information can be used by optimization algorithms for deciding when to modify (*e. g.*, grow, shrink, or moving resources) the resource allocation on a HyFS.

B. Review of Cloud Management Approaches

Cloud computing has been only vaguely defined until Vaquero et al. [4] provided a consolidated definition in 2008. Their definition particularly reveals the traditional view of the cloud computing paradigm in which the focus remains around computing and storage resources and their provisioning in a virtualized form.

The limitations in the support of comprehensive networking functions is also apparent in the most recent software platforms for cloud computing and has been highlighted by Benson et al. [11]. Consider for example these major open-source software platforms for cloud computing: Nimbus [12], OpenNebula [13], and Eucalyptus [14].

In Nimbus the underlying network is controlled through DHCP servers and random assignment of MAC addresses [15]. Network hardware or soft switches are not considered in the platform. Network configuration functions in OpenNebula are restricted to the creation of a DHCP IP range that will in turn be automatically configured in each virtual machine. Eucalyptus [14] supports a number of networking modes to deal with network setup. In SYSTEM Mode, an External DHCP server is used. In STATIC Mode, an Internal DHCP server with static MAC-to-IP assignments is used. In MANAGED Mode, virtual machines receive private IP address and traffic between virtual machines is isolated through VLAN tags. It is also possible to assign public IP addresses, to create forwarding rules for specific services, and to route traffic between different VLAN tagged networks through the platform controller.

To sum up, all these solutions work with the assumption that the cloud resources lay on top of a fully connected Layer 2, sometimes VLAN enabled, switched network. Virtual machines are "connected" together by assigning them IP addresses in the same range. When it is necessary, VLAN tags isolate traffic of different customers. Specially for network intensive applications, more advanced network control and management functions are still required. Such functions could include creating virtual overlays to interconnect virtual machines, assigning QoS parameters to every link through which these virtual machines communicate, and advanced monitoring of both physical and virtual resources.

The network capabilities of OpenStack [3], another major cloud management project, are currently being extended in the Quantum [16] project. Key network features supported through Quantum are the transparency of VLAN creation for the customer, the association of interfaces with the network as an explicit step, and the ability of plugins to expose API extensions that introduce more complex functionality, such as QoS. However at the time of writing, no detailed information is available.

From the discussions in the introduction and the assessment of related work in this section, we conclude that a more holistic cloud platform must comprise network features that go significantly beyond the current basic network functions found in the prevailing software platforms.

Furthermore, because these platforms for cloud computing do not consider network features as a primary goal, implement-

ing functions for resource allocation on top of them is difficult as this would require the redesign of and hence significant modifications to the platforms' implementation core. Note, that we do not focus on algorithms or mechanisms for resource allocation in the cloud. In this area we can and will leverage the insights from several papers on network testbeds, such as VANI [17] or from the (Proto)GENI project or PlanetLab.

Essentially, in this paper, we target at a new conceptual approach that considers both computing/storage and networking on the same level, in which we develop uniform interfaces that allow to specify, control, and manage *hybrid* clouds no matter what types of resources they may use.

III. Architecture

Figure 1 depicts the conceptual building blocks of the architecture of the Hybrid Flash Slice (HyFS) Manager. The four major components include the *Cloud Resources* which are controlled and monitored by the *Control Logic*. Different types of resources can be controlled by different Drivers, yet all of them provide a unified API to the *Custom Programs* and the *User Interface*.

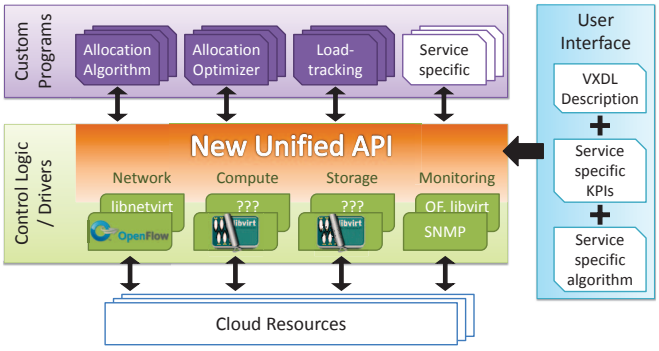

Fig. 1. Conceptual Architecture of HyFS Manager

A. User Interface

The User Interface represents the interaction point with the HyFS framework. Through it the specification of the requested resources is transmitted to the system. The initial specification consists of up to three parts. All requests need a *description* of the desired resources. Optionally, extending the functionality of current cloud control systems, in HyFS the specification can also include *service specific key performance indicators (KPIs)* and *service specific resource management algorithms*.

The **Description** provides details on the resources needed for a given service or application to be deployed. This specification may give more or less information about the initial allocation depending on the previous knowledge the user has about the needs of the service. For example, if very few information is provided in the specification, the algorithms designed for resource allocation will have more freedom in deciding where virtual resources should be placed and how they should be connected. On the other hand, if the application has a well-known set of constraints and goals, the initial description can determine in much greater detail the virtual resource needs.

Service specific KPIs The service specific KPIs are provided by service components and shall be monitored by the control logic. Resource management algorithms can use these KPIs to adapt to changing conditions and utilization.

Service specific algorithms These algorithms will usually monitor a set of metrics and KPIs, provided either by the running application or by the HyFS controlled infrastructure. A simple algorithm would trigger actions (*e. g.*, create/destroy/move virtual machines, expand link capacity, create new connections) when certain thresholds for these metrics are reached. The service specific algorithms use the unified API and can take application needs into account. Typically such algorithms will be reviewed by cloud providers administrators prior to activation in the system.

Moreover, the User Interface provides basic functionality both for the user and the cloud administrator to manage virtual resources individually, such as creating virtual storage volumes, migrating virtual machines, and establishing virtual links.

B. Custom Programs

The custom programs implement the actual management of resources. As mentioned before they can either be provided by the user along with a request for cloud resources, or developed by the cloud provider. We envision that a basic set of programs could be shipped together with HyFS, which can then be tuned to the needs of each cloud provider. We envision at least three types of custom programs: *Allocation Algorithms*, *Allocation Optimization* and *Load-Tracking*.

The core concept behind HyFS custom programs is that they allow the administrator to plug in and run different algorithms (*e. g.*, centralized vs. distributed, complex multi-objective vs. simple heuristics) depending on the particular needs of the each cloud environment. Algorithms will be able to use a set of common functions provided by a programmable unified API implemented by the *Control Logic* to access and modify virtual resource allocations.

Allocation Algorithms are responsible for creating an initial resource allocation based on the initial request. Such an algorithm can focus on generating a fast and maybe non-optimal allocation.

Allocation Optimizer will continuously try to optimize the allocations of all users, thereby correcting potentially non-optimal initial deployments. These programs could implement different goals, *e. g.*, minimize energy consumption, maximize resource utilization, or optimize topology of deployment.

Load-Tracking will adapt the deployed infrastructure to various elasticity parameters, such as demand fluctuations and network conditions. These algorithms will receive events triggered by the monitoring systems or as input from an administrator. Typically, these events will be threshold breaches of the metrics defined in the elasticity parameters of the initial

specification whose associated actions should be interpreted and enforced by the cloud platform.

C. Control Logic / Drivers

The most important part of the Control Logic is to provide a resource independent interface for managing the cloud resources. In HyFS this *unified API* allows to manage all of *network*, *compute*, *storage* and *monitoring* on the same level. Pluggable drivers for different types of resources can provide compatibility with a wide variety of existing and future cloud resources.

Network, Compute and Storage drivers abstract the complexity of many possible different underlying virtualization infrastructures that are available within a data center or across several data centers. They allow allocation and optimization algorithms to employ high-level management functions that can access and modify virtual resources regardless of platform-specific parameters. There might be different drivers in the same category, *e. g.*, OpenFlow or libnetvirt for network control.

Monitoring will provide access to the metrics, both in the service/application level (inside a slice) and in the virtual infrastructure level (*e. g.*, performance parameters from virtual machines and virtual links). Besides being configured to monitor different types of metrics from endpoints across a network, this system should be able to trigger events according to thresholds defined.

The control logic will also need an information database which contains the internal representation of every HyFS deployed on the cloud as well as the users of the system and their access rights.

IV. Conclusion

In this study we have discussed some of the main issues found in today's cloud management platforms. Based on these observations we have taken this first step towards a new conceptual design of a cloud platform with focus on three main aspects: (*i*) support for advanced networking paradigms and technologies from the design phase, (*ii*) support for richer specification of virtual infrastructures, and (*iii*) flexibility to adapt the platform's core resource management strategies via a high-level unified programmable API.

We are already working on a prototype and evaluation scenarios for our proposal. Although much work is yet to be done, a first version of our prototype has been already demonstrated at NOMS'2012 [18] showing basically the feasibility of our approach. In future work we will provide more details on the implementation and discuss our design decisions based on example real-world use-cases. We aim to show the benefits of complex allocation algorithms and elasticity parameters.

References

[1] OpenNebula, "The open source solution for data center virtualization," 2008, Accessed: Feb. 2012. [Online]. Available: http://www.opennebula.org/

[2] Eucalyptus, "The open source cloud platform," 2009, Accessed: Feb. 2012. [Online]. Available: http://open.eucalyptus.com/

[3] Rackspace Cloud Computing, "Openstack cloud software," 2010, Accessed: Feb. 2012. [Online]. Available: http://openstack.org/

[4] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, Dec. 2008. [Online]. Available: http://doi.acm.org/10.1145/1496091.1496100

[5] SAIL, "Scalable and Adaptive Internet Solutions," 2010, Accessed: Feb. 2012. [Online]. Available: http://www.sail-project.eu/

[6] P. Murray *et al.*, "(D-D.1) Cloud Network Architecture Description," EU FP7 Project Deliverable, Tech. Rep., July 2011, Accessed: Feb. 2012. [Online]. Available: http://www.sail-project.eu/wp-content/uploads/2011/09/SAIL_DD1_final_public.pdf

[7] A. Strømmen-Bakhtiar and A. R. Razavi, "Cloud computing business models," in *Cloud Computing for Enterprise Architectures*. Springer London, 2011, pp. 43–60. [Online]. Available: http://dx.doi.org/10.1007/978-1-4471-2236-4_3

[8] G. P. Koslovski, P. V.-B. Primet, and A. S. Charão, "VXDL: Virtual Resources and Interconnection Networks Description Language," in *Networks for Grid Applications*. Springer Berlin Heidelberg, 2009, vol. 2, pp. 138–154. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02080-3_15

[9] Institut Télécom, "pyOCNI - Python Open Cloud Networking Interface," 2011, Accessed: Apr. 2012. [Online]. Available: http://occi-wg.org/2012/02/20/occi-pyocni/

[10] Distributed Management Task Force (DMTF), "Open Virtualization Format (OVF)," Jan. 2010, Accessed: Apr. 2012. [Online]. Available: http://dmtf.org/standards/ovf

[11] T. Benson, A. Akella, A. Shaikh, and S. Sahu, "Cloudnaas: a cloud networking platform for enterprise applications," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, ser. SOCC '11, 2011, pp. 8:1–8:13.

[12] K. Keahey, R. Figueiredo, J. Fortes, T. Freeman, and M. Tsugawa, "Science clouds: Early experiences in cloud computing for scientific applications," in *Cloud Computing and Its Applications (CCA)*, Chicago, USA, October 2008.

[13] B. Sotomayor, R. Montero, I. Llorente, and I. Foster, "Virtual Infrastructure Management in Private and Hybrid Clouds," *IEEE Internet Computing*, vol. 13, no. 5, pp. 14–22, Sept.-Oct. 2009.

[14] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID)*, May 2009, pp. 124–131.

[15] M. Tsugawa and J. Fortes, "A virtual network (ViNe) architecture for grid computing," in *20th International Parallel and Distributed Processing Symposium (IPDPS)*, April 2006, p. 10 pp.

[16] Rackspace Cloud Computing, "OpenStack Quantum Project," 2011, Accessed: Apr. 2012. [Online]. Available: http://wiki.openstack.org/Quantum

[17] H. Bannazadeh and A. Leon-Garcia, "Virtualized application networking infrastructure," in *Proceedings of TridentCom 2010*, 2010.

[18] J. A. Wickboldt, L. Z. Granville, D. Dudkowski, and M. Brunner, "Hyfs manager: A hybrid flash slice manager," 2012, demo Presented of 13th IEEE/IFIP Network Operations and Management Symposium (NOMS 2012): Student Demo Contest, 16-20 April 2012, Maui, Hawaii.