Enabling User-Centric Data-Intensive Application Deployment in Clouds Using the Open Cloud Exchange

Cosmin Dumitru, Ralph Koning, Ana-Maria Oprescu, Yuri Demchenko, Paola Grosso, Cees de Laat
University of Amsterdam
System and Network Engineering Group
Amsterdam, The Netherlands
{c.dumitru,r.koning,a.m.oprescu,y.demchenko,p.grosso,delaat}@uva.nl

***Abstract*—The Open Cloud eXchange (OCX) concept has been defined as *a place for inter-connection and peering between cloud providers and customers*. The concepts surrounding OCX focus mainly the network aspects and are currently being implemented within GÉANT (gOCX). The current goal of the OCX is to seamlessly provide layer 0-2 connectivity between cloud sites and cloud users. In this work we investigate the advantages and challenges of using the OCX instead of traditional (public Internet) multi-cloud connectivity. To this end, we deploy a user-centric cloud resource scheduler and execution engine for bags-of-tasks. Based on our previous demonstration at TNC2014 we put froward the plan for the SC14 demonstration of data-intensive applications on the gOCX infrastructure.**

I. Introduction

The Open Cloud eXchange (OCX) [11], [13] concept has been defined as *a place for inter-connection and peering between cloud providers and customers*. The concepts surrounding OCX focus mainly the network aspects and are currently being implemented within GÉANT [10], [19]. The current goal of the OCX is to seamlessly provide layer 0-2 connectivity between cloud sites and cloud users.

Many data-intensive cloud computing applications require high performance and stable network interconnections in order to provide the optimum service to users. It is envisioned that the OCX will enable these applications with a service-like view of the private network connecting multiple cloud provider sites.

The problem of interconnecting multiple cloud providers through private links in the presence of data-intensive applications can be viewed from multiple perspectives:

- from the perspective of the connectivity providers
- from the perspective of the cloud providers
- from the perspective of application and middleware developers
- from the perspective of the users (application owners)

OCX is distributed and implemented as a set of interconnected OCX nodes or partners over the multi-gigabit GÉANT infrastructure. This paper describes our findings and experience while using the GÉANT OCX (gOCX) infrastructure to perform ultra-high-definition video transcoding during the TERENA Network Conference 2014 [12], [8] and in preparation for the SC14 demonstration. We aim to provide novel insight into the challenges surrounding the emerging concept of Open Cloud eXchange with respect to the four perspectives mentioned above.

II. Demonstration Description

The main goal of the demonstration at TNC2014 was to investigate how data-intensive applications may benefit from the potential power of the OCX concept compared to using the routed internet.

We demonstrated that the GÉANT OCX (gOCX) infrastructure can be used for multi-server video encoding by transcoding the very high resolution frames of Sintel [7], from the original lossless format to a compressed format more suitable for web viewing. This is a highly processor intensive task to reach the target performance of 24 frames per second (i.e. the frame rate used in TV and cinema productions).

The initial implementation of the gOCX contained 3 gOCX sites (SURFnet, GRNET, SWITCH) connected by the GÉANT network. These gOCX sites connected the video data located at the University of Amsterdam to the compute resources of the pioneering cloud providers Okeanos and CloudSigma.

We aimed to identify a first set of extra challenges cloud application developers and users would face having this degree of flexibility available. We proceed to describe the technical details of the demonstration and its outcome.

A. An OCX-aware application deployment environment

For the demonstration we leveraged Vampires[14], a cloud scheduler and execution engine for data-intensive bags-of-tasks applications, an extension of the BaTS[18], the budget-aware-task scheduler. Bags-of-tasks are a type of workload which consists of independent jobs which can be executed in any order. Typical examples include image processing, parameter sweeps, design space exploration or any other type of batch jobs which have no interdependency. The provisioning component is built using the *jclouds* Java library, which provides a common API for multiple cloud providers. The user-specified resources were provisioned from the selected cloud providers and then the application was deployed on to them.

For the purpose of the TNC2014 demonstration, we used a bag-of-tasks workload consisting of a multimedia application: elastic transcoding [5] and processing [3] of very high resolution

images. The input data (image set) is the 4K (4096x1720 pixels) version of the 10 minute open source movie *Sintel*. The application deployment configuration was controlled by the user via a web application (GUI). The scheduler used a straightforward self-scheduling policy to distribute the tasks to provisioned resources.

The web interface to the application had three main areas: a) the preview area, where the user can view in real time the result of the transcoding, b) the control and status area and c) the configuration area at the bottom. In the configuration area the user could select the resources which would be used to execute the application. The workload parameters were also configured here: what image set to use and what transformations to apply to it. Each Cloud Service Provider offers different types of virtual machines, with varying CPU and memory capacities and the interface allows users to select the amount and type of resources to be requested from each Cloud Service Provider. To showcase the advantages offered by using OCX, the interface allows users to also select the network resources to be used by the application. There are two options available: the Internet or the high speed OCX network.

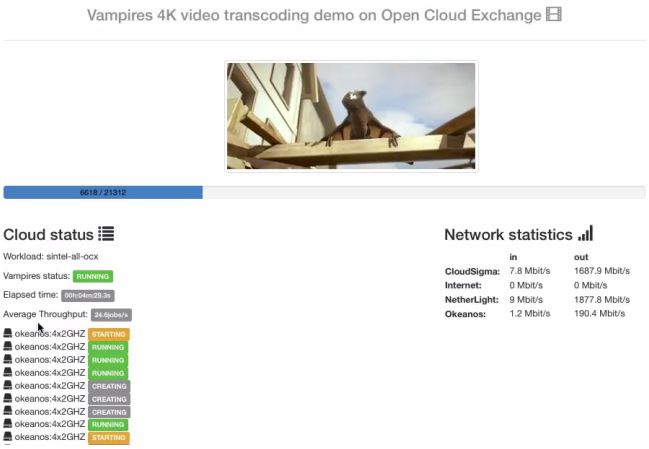

Figure 1. The GUI of the OCX Demonstration Application showcasing the preview area and the status information during execution

Once the application and the required resources have been configured, the execution phase may begin. As images are processed, the preview area starts displaying the last processed image together with real-time network statistics.

B. The OCX-connected storage and computing infrastructure

The workload data source and destination are stored on FIONA [9], the network appliance developed by the USCD. This appliance has 1.4TB of flash backed with 18T of hard disk storage, and is designed to be a source for high quality video streams at speeds above 10Gb/s.

FIONA is running a nginx [4] web server on a private interface to serve images to the cloud sites and a flask [2] web application to receive the processed image data. The output image is then served on the public interface together with interface statistics and become embedded in the demonstration GUI.

The total amount of data transferred during one execution was approximately 160GB. We tried to use similar types of virtual machines at both cloud providers sites. The instances were configured to have 4 CPU cores running at 2GHz and 4GB of RAM. All virtual machines were connected to both the OCX network and the Internet. The demonstration users were able to chose between the two connectivity options for each execution.

C. Network setup

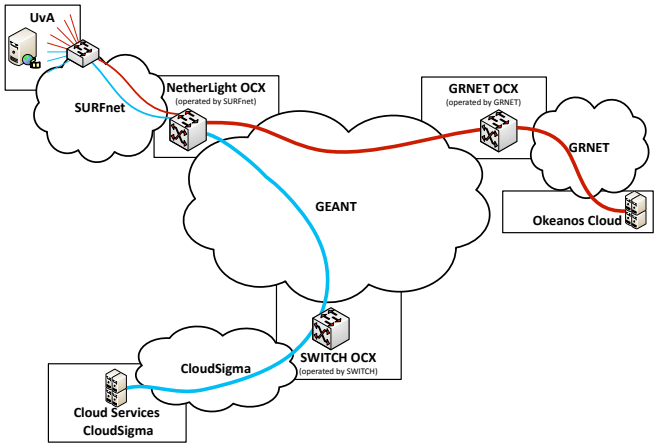

Figure 2. Network topology for the TNC2014 demonstration

Figure 2 shows the global network topology. The storage infrastructure at the University of Amsterdam (UvA) was directly connected to the compute infrastructure of CloudSigma (commercial cloud) and Okeanos (GRNET's research cloud) through various OCX sites tapping into the GÉANT backbone. The network used private 10G and 1G Ethernet services provided by the cooperating NRENs to the three endpoints.

The connection between Okeanos and UvA consisted of the 802.1q vlans 3301-3310 on a 1G Ethernet service which were mapped to 10 predefined networks in the Okeanos cloud interface. The connection between UvA and CloudSigma consisted of a Q-in-Q tagged vlan 2611 on a 10G Ethernet service. CloudSigma used the 50-1000 range as inner vlans, which were randomly assigned to virtual networks created via their cloud API. However, the actual mapping could not be queried via the cloud API and their support department had to be contacted to see what actual vlans are mapped to the virtual networks.

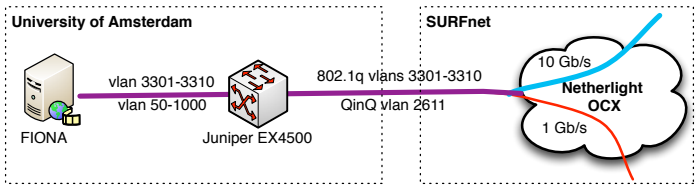

Figure 3. Local network configuration at UvA

Figure 3 shows the local configuration at the UvA. The 10G Ethernet service from Netherlight was connected to a network switch at the UvA. The switch stripped the outer Q-tag and forwarded the inner vlans 50-1000 together with the vlan range

3301-3310 from Okeanos to FIONA which was connected at 10Gb/s.

The vlans that were used for the demonstration were combined on FIONA in a Linux network bridge. This created a transparent Ethernet domain between the UvA and the different cloud sites and offered the advantage of running a DCHP server which provided VMs from both cloud providers with IP addresses and (possibly) other configuration options.

D. Results

We have analyzed the behavior of the system in two scenarios: regular internet inter-site connectivity and OCX-based inter-site connectivity.

Using the regular internet connection, the maximum achieved throughput from the UvA storage server to any of the two cloud providers was 1Gbit/s. This translated to roughly 10 processed images per second.

Using the OCX network, the maximum achieved throughput from the UvA storage server to any of the two cloud providers was 3Gbit/s, i.e. a three-fold increase compared to the public internet scenario. This increase in the network throughput allowed for processing speeds close to 30 frames per second. This magnitude of processing rates would allow users to visualize the resulted movie frames without buffering or delays.

III. Lessons Learned

A. Heterogenous cloud APIs

One challenge for the application developers is the variety of cloud APIs. There are currently a number of efforts to standardize these APIs, such as the Open Cloud Computing Interface [17], or to abstract away functionality in order to have a common core API, regardless of the provider. However, in practice this proves to be more difficult than expected, as some providers have only partial API implementations, confusing application developers who expect that everything works in a consistent manner across all providers, i.e. the APIs share the same semantics. For example, CloudSigma sees the virtual image as a disk which can be reused, while Okeanos sees the virtual image as an ephemeral entity, which disappears when tore down.

B. Resource Descriptions and Performance Equivalence

Each cloud provider offers resources to the users in different ways. CloudSigma users have the flexibility to define their own types of instances, with a custom number of CPUs, each having possibly varying performances. The amount of RAM memory and disk can also be customized. On the other hand, providers like Amazon and Okeanos offer users predefined instance profiles, with fixed amounts of memory, CPU and disk. Efforts like the Infrastructure Description Language (INDL) [15] or PROTOGENI's rspec [6] aim at providing a consistent way of describing compute and/or network resources. However, these efforts haven yet to see adoption across a wide number of providers.

Due to these different cloud resource packaging policies, it is difficult to match or equivalate resources from different cloud providers using just static information. This means that users have to profile their applications to gain a good understanding of what kind of performance they can expect from the offered resources.

C. Cloud interoperability

In order to easily deploy the application during the demonstration, both cloud providers had to be pre-configured with similar virtual machine instance images. We found it difficult to deploy the same image to both providers as they used different image formats. This proved to be time consuming as the image occasionally had to be rebuilt. Automated deployment tools, such as Ansible [1], are of great use to reduce build time and keep consistency across different cloud image formats.

D. Heterogeneous cloud boot-up times and API reliability

The amount of time required to provision resources varies among providers [16]. Users and application developers have to be aware of this when deploying their applications on cloud resources. In our experience, the boot-up time varied between 1 to 10 minutes across providers. In the case of applications which need to quickly scale or start this aspect requires extra planning and synchronization. Apart from slow start-up times, there were also cases where the API calls for provisioning or querying resources simply failed. This was either due to provider-side reasons, such as faulty hardware nodes, or due to hitting the upper bound on resource requests (maximum number of acquired resources). The latter occurred during the stress testing procedure prior to the demonstration.

E. Heterogenous billing models

Budget planning is an important aspect of cloud computing. Similar to the API and the resource description model, cloud providers use different billing models. The most common approach is to charge by the time unit. This time unit can be different for each cloud provider. For instance, CloudSigma uses a 5-minute billing interval which allows users to fine tune their cloud deployment configuration. Other cloud providers, such Amazon EC2, use a larger time interval, i.e. 1 hour, which makes accounting simpler to handle and understand, but also incurs a larger cost for experimenting and testing various deployments.

F. Network

At the time of the TNC2014 demonstration, i.e. in the early stages of the gOCX, the network configuration still presented some challenges. At each gOCX site, as well as at the end-points (UvA, Okeanos and CloudSigma), some of the network configuration had to be done manually, a very time-consuming and error-prone process. Next, the different technologies to hand over the vlans (801,2q and q-in-q) presented further challenges in configuring the equipment and required the contribution of the network engineers. However, the OCX concept dictates that the gOCX infrastructure should be accessible to network-agnostic end users. Therefore, the next natural step for gOCX is to make use of the progress in the NSI working group [20], [21] of the Open Grid Forum in order to enable flexible provisioning of multi-domain network services.

a) IP Connectivity: As public IPv4 addresses become more and more scarce, cloud service providers might stop assigning them by default to a cloud instance and start charging extra for provisioning one. In our demonstration, one of the cloud providers opted to limit the amount of IPv4 addresses available and assigned by default only IPv6 addresses. The other provider only used IPv4 addresses. This restriction made communication over the public internet impossible. Private networking solutions, like the gOCX infrastructure, circumvent this problem and enable the use of any private IP range. They also do not limit the available bandwidth, which is generally the cost incurred by using tunnels and VPNs between providers or dealing with firewalls and other network policies.

IV. SC14 Demonstration and Future work

During SC14 we plan to showcase an extended demonstration of the Open Cloud eXchange concept. Resource-wise, we will demonstrate the system with an extended pool of cloud providers available to the users. By extending the pool of available resources, the task of selecting the right set of resources on which to execute the workload will become more difficult. We will include a scenario where users will also be presented with a certain budget range.

The user-centric component of the Vampires scheduler will be extended with a module which performs application performance prediction for a given orchestrated set of network, computing and storage resources. This performance prediction module extends the existing implementation of a theoretical performance model [14]. Additionally, we will increase the number and type of applications showcased. It is our vision that distributed data-intensive applications deployed on clouds will benefit from the increased capacity and flexibility offered by the Open Cloud eXchange concept. The TNC2014 demonstration has focused more on functionality and slightly less on obtaining high (peak) performance. The SC14 demonstration also intends to highlight the benefits of having on-demand high performance computing infrastructure distributed across multiple providers.

We also plan to attach a monetary cost to using the gOCX services and investigate user-centric data-intensive application scheduling. This part will include different cost models for Ethernet services. Moreover, we will consider future generation cost models such as the ones based on actual energy usage of the network components [23]. A first step to energy-aware network billing was already showcased at SC13 and TNC2014 through the GreenPath demonstration [22].

V. Conclusion

In this paper we have presented the GÉANT Open Cloud eXchange platform and the results of the first demonstration which leverages the ability to transparently connect the resources rented by a single user from different cloud providers. We showed the potential of the gOCX infrastructure for data-intensive as well as CPU-intensive applications, such as ultra-high-quality video transcoding.

During the preparation and demonstration execution we have encountered several critical new challenges which we have put forward as lessons learned from our experience with gOCX, for the benefit of different actors: application developers, application owners and cloud providers. The outcome of this was a better understanding of the future requirements of users and applications from complex network infrastructures.

Acknowledgements

We would like to thank all the people in the GÉANT gn3plus-jra1-t2 project, especially Tasos Karaliotas (GRNET/Okeanos), Damir Regvart (CARNet), Kurt Baumann (Switch), Migiel de Vos (SURFnet) and the people from CloudSigma for providing the resources, their help and support during the planning and execution of the demonstration.

References

[1] Ansible is simple it automation. http://www.ansible.com/home. Accessed: 2014-09-01.

[2] flask. http://flask.pocoo.org. Accessed: 2014-09-01.

[3] Imagemagick: Convert, edit, or compose bitmap images. http://www.imagemagick.org/. Accessed: 2014-01-27.

[4] nginx. https://nginx.org. Accessed: 2014-09-01.

[5] Openjpeg - jpeg2000 codec. http://www.openjpeg.org/. Accessed: 2014-01-27.

[6] Rspec. http://www.protogeni.net/wiki/RSpec. Accessed: 2014-09-01.

[7] Sintel: Open source movie . http://sintel.org/.

[8] videoocx. https://www.youtube.com/watch?v=q7IAAFUcTY0. Accessed: 2014-09-01.

[9] T. DeFanti, P. Papadopoulos, and J. Schulze. Fiona the flash i/o network applicance. http://www.cian-erc.org/pdf/2013_projects/Thrust%201/C1-1b%20THE%20FLASH%20IO%20NETWORK%20APPLIANCE.pdf.

[10] Y. Demchenko, M. de Vos, D. Regvart, S. Filiposka, T. Karaliotas, K. Baumann, D. Arbel, J. van der Ham, R. Strijkers, and E. Escalona. Open cloud exchange (ocx).

[11] Y. Demchenko, J. v. d. Ham, C. Ngo, T. Matselyukh, S. Filiposka, C. d. Laat, and E. Escalona. Open cloud exchange (ocx): Architecture and functional components. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 2, pages 81–87. IEEE, 2013.

[12] Y. Demchenko, R. Koning, C. Dumitru, C. de Laat, M. de Vos, T. Karaliotas, K. Baumann, D. Regvart, and S. Filiposka. Open cloud exchange (ocx): Architecture, components, and demo scenario. https://tnc2014.terena.org/core/poster/11.

[13] J. der Ham, C. de Laat, T. Matselyukh, E. Escalona, M. de Vos, S. Filiposka, T. Karaliotas, A. Mavrin, D. Regvart, K. Baumann, D. Arbel, et al. Open cloud exchange (ocx): Bringing cloud services to nrens and universities.

[14] C. Dumitru, A.-M. Oprescu, M. Živković, R. van der Mei, P. Grosso, and C. de Laat. A queueing theory approach to pareto optimal bags-of-tasks scheduling on clouds. In *Euro-Par 2014 Parallel Processing*, pages 162–173. Springer, 2014.

[15] M. Ghijsen, J. van der Ham, P. Grosso, and C. de Laat. Towards an infrastructure description language for modeling computing infrastructures. In *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, pages 207–214. IEEE, 2012.

[16] M. Mao and M. Humphrey. A performance study on the vm startup time in the cloud. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 423–430. IEEE, 2012.

[17] T. Metsch, A. Edmonds, et al. Open cloud computing interface–infrastructure,". In *Standards Track, no. GFD-R in The Open Grid Forum Document Series, Open Cloud Computing Interface (OCCI) Working Group, Muncie (IN)*, 2010.

[18] A.-M. Oprescu, T. Kielmann, and H. Leahu. Budget estimation and control for bag-of-tasks scheduling in clouds. *Parallel Processing Letters*, 21(02):219–243, 2011.

[19] D. Regvart, Y. Demchenko, S. Filiposka, M. de Vos, and T. Karaliotas. Milestone ms101 (mj1. 2.1): Network architectures for cloud services white paper.

[20] G. Roberts, T. Kudoh, I. Monga, J. Sobieski, J. MacAuley, and C. Guok. GFD.212: NSI Connection Service v2.0. Technical report, 2014.

[21] G. Roberts, T. Kudoh, I. Monga, J. Sobieski, and J. Vollbrecht. GFD.173: Network Services Framework v 1.0. Technical report, 2010.

[22] K. van der Veldt. Carbon-aware path provisioning for nrens. https://tnc2014.terena.org/getfile/1405.

[23] K. van der Veldt, I. Monga, J. Dugan, P. Grosso, and C. de Laat. Carbon-aware path provisioning for nrens. In *proceedings of International Green Computing Conference IGCC14, accepted for publication*.