Designing a massively distributed IaaS toolkit by leveraging OpenStack

Jonathan Pastor, Adrien Lebre
ASCOLA Research Group
Mines Nantes / Inria / LINA
Nantes, France

Laurent Pouilloux
Avalon Research Group
LIP ENS Lyon UMR 5668
Lyon, France

Frédéric Desprez
Corse Research Group
XXX TODO XXX
Grenoble, France

firstname.lastname@inria.fr

ABSTRACT

The use of computing ressources delivered by cloud providers has become very common. These providers leverage on big data-centers containing tens of thousands of servers. However concentrating ressources in a few data-centers leads to critical situations in term of performance, reliability and data confidentiality. Leveraging the micro DC concept, we propose to address such concerns by operating cloud resources in a fully decentralized manner.
Although a reference architecture describing fundamental services constituting an IaaS manager has been proposed, a detailed overview of the mechanisms that are needed to build a massively distributed cloud is still missing. Reimplementing such a system from scratch would be a non sense goal, considering the architecture complexity and velocity of open-source initiatives such as OpenStack.
In this paper, we first depict a detailed map of the services composing the OpenStack cloud manager, and for each of its component we give leads for targeting a fully distributed functioning. These leads will be demonstrated in the second part this paper, via a proof of concept that have been validated on the Grid'5000 testbed.

Keywords

Cloud computing, IaaS Architecture, locality, OpenStack, Distributed Hash table, peer to peer

1. INTRODUCTION

The success of Cloud Computing has driven the advent of Utility Computing (UC). However to answer the escalating demand for computing resources, Cloud Computing providers must build data centers (DCs) of ever-increasing size. This concentration of computing ressources leads to issues like connectivity to the application/data located in a similar geographical zone during disasters or outages. Besides facing the well-known issues large-scale DCs have now to deal with energy considerations that limit the number of physical resources that one location can host.

All these problems can be tackled by hybrid or federated Cloud solutions [1], that aim at extending the resources available on one Cloud with those of another one. However a recent IEEE report [3] shows that network traffic continues to double roughly every year. Consequently, bringing computing resources closer to the end-users, thus minimizing the energy impact and saving bandwidth.

The concept of micro/nano DCs at the edge of the backbone [2] may be seen as a solution to reduce the network overhead. Hence, we propose to extend each points of presence (PoP) with a number of servers dedicated to hosting virtual machines (VMs). From the physical point of view, network backbones provide appropriate infrastructures, i.e., reliable and efficient enough to operate UC resources spread across the different PoPs. Ideally, UC resources would be able to directly take advantage of computation cycles available on network active devices, i.e. those in charge of routing packets.

Proponents of Cloud federations would argue that is it possible to perform a federation a micro-Clouds hosted on each micro DC, thus having a service node in each micro-Cloud. However this solution is not without its flaw: aside from wasting one node in each PoP, when a service node fails all the node generating the computing power become unusable.

We propose the LUC Operating System (OS), an advanced system being able to unify many UC resources distributed on distinct sites, that would enable Internet service providers (ISPs) and other institutions in charge of operating a network backbone to build an extreme-scale LUC infrastructure with a limited additional cost. Instead of redeploying a complete installation, they will be able to leverage IT resources and specific devices such as computer room air conditioning units, inverters or redundant power supplies already present in each center of their backbone.

The remainder of this article is structured as follows. In Section 2, we discuss the definition and properties of a massively distributed cloud. Section 3 gives an overview of the LUC OS design proposal. In Section 4, we describe existing mechanisms that we will revisit to build our massively distributed IaaS manager. In section 5 we introduce the method that will be used to integrate our dynamic scheduler in OpenStack, thus building the first version of the LUC OS. Finally, we discuss perspectives and conclude this article in Section 6.

2. FROM CENTRALIZED TO FULLY DISTRIBUTED CLOUDS

2.1 Distributing the clouds: From few large DCs to many spread Nano DCs

In the current model of cloud computing, cloud providers are delivering services to their users that are located all over the world. Some very successful services have build worldwide infrastructure to be able to handle every requests of their users with a good quality of service (QoS). However in the current model, the production of computing resources is concentrated in few large data-centers that contains tens of thousand of servers, in order to benefit from economies of scale introduced by SKU (stock keeping unit) standardization and saving in technical staff capacity. Moreover, this concentration of servers in geographically restricted area with extremely performant networking enables IaaS mechanisms to take advantage of such fast connectivity. However the ever increasing data-centers size has become a problem, as these data-centers require dedicated electrical and cooling infrastructure.

As an alternative this concentration of the production of computing resources in these large data-centers, some authors have proposed some alternatives. One consists in leveraging smaller data-centers: Greenberg and al [2] have proposed the concept of geographically spread micro data-centers to bring the cloud closer to end users. Althought this model seems to solve some issues encountered with large data-centers, two crucial questions remains:

- How to connect these geographically spread nano data-centers?
- How to operate efficiently such an infrastructure?

2.2 The Discovery initiative: Nano DCs on top of ISPs backbone

Deploying an infrastructure that can connect hundreds of micro-DCs requires a huge financial and human effort, a better approach would be to leverage existing infrastructures.

The Discovery initiative tackles this problem by proposing to associate these geographically spread micro-DCs to the network backbone deployed by internet service providers (ISP). The idea is to extend each point of presence (PoP) with few utility computing capacity such as commodity servers. This would enables to deploy fully distributed cloud computing infrastructure on top of existing networks.

The question of how to operate this kind of infrastructure still remains: at this level of distribution, latency and fault tolerance become primary concerns, and collaboration between servers of differents location must be organized wisely. For these reasons, the Discovery initiative proposes to develop a new kind of IaaS system that would operate such an infrastructure, by taking into account network locality.

3. THE DISCOVERY INITIATIVE: TOWARDS A FULLY DISTRIBUTED CLOUD-OS

3.1 Architectural proposition

An IaaS toolkit should be delivered with a set of default high level mechanisms whose assembly results in an basic operational IaaS system. In the case where one of the default constituting mechanisms would not be sufficient, it should be redeveloped by leveraging the toolkit's low level mechanisms.

Recent studies have showed that state of the art IaaS manager [5] were constructed over the same concepts. Furthermore a reference architecture for IaaS manager has been described in [4] enabling the design of an IaaS toolkit. The strength of this work resides in the fact that they considered all the needs of an "industrial class" IaaS manager: each of its functions has been widely studied. As this work provides a good level of details, we think that a new architectural proposition for a distributed IaaS manager should leverage this work.

To maximize the chance of being reused by a large community, an IaaS toolkit should enable an easy integration with one or several existing IaaS cloud managers. In our case we have chosen to leverage the OpenStack project: as a result the mechanisms developped with the toolkit will integrate well with existing clouds that are based on OpenStack.

However, since we aim at making a research prototype of a massively distributed cloud, some parts of this reference architecture are outiside the scope of a research prototype. That is why we will volontarily neglect advanced services in order to focus on those that can be considered as vitals, enabling us to propose an architecure for a simple but working massively distributed IaaS manager. We also keep in mind that the architecture should be flexible enough to later include these advanced services in a painless way.

3.2 Minimizing implementation effort, by leveraging OpenStack

OpenStack is an opensource project that target at developing a cloud manager. It is composed of several services that handles the different aspects of a cloud infrastructure (Computing, Networking, Storage, ...). Its architecture is based on the "shared nothing" principles: each sub-service is connected to the others via two different way:

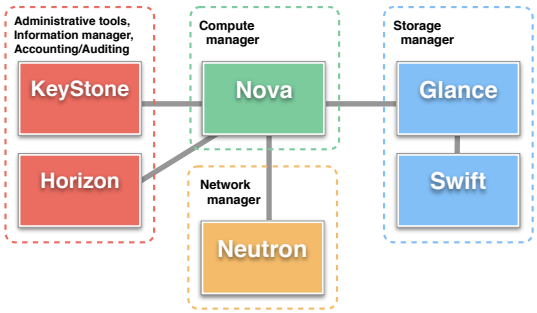

Figure 1: Services composing OpenStack.

- **A messaging queue** that enables the collaboration between sub-services of a same controller.
- **A database** that stores inner states of OpenStack services.

As this project is successful and is used by a large community, we propose that instead of reinventing the wheel by developing each components of the LUC-OS from scratch, to leverage the OpenStack project. This strategy would enable to focus the effort on the key issues such as the distributed functionning, the fault tolerance mechanisms and the organization of efficient collaborations between nodes of the infrastructure.

4. TOWARDS A FULLY DISTRIBUTED OPENSTACK DEPLOYMENT

The discovery initiative targets the delivery of an utility computing platform that will be working on top of existing network backbone facilities. Starting the development of such a platform from zero would require a titanic effort: in order to spare a giant development time, the Discovery initiative proposes to leverage the OpenStack project: this will serve as the foundation of the LUC-OS.

In order to structure the LUC-OS on a fully distributed peer to peer functionning, OpenStack would be required to be fault tolerant and to be able to fit on a multi-site configuration. In the current situation, it requires some adaptations: in this section we propose some modifications that have been introduced in the Nova controller, to meet the two aforementioned criterions.

4.1 Replacing the relational backend by a Key/value store

From today's perspective, most of the OpenStack deployment are involving few nodes, thus not requiring more thant one controller. However, to meet the fault tolerance criterion, one needs to use *High availibility* deployment by combining two components *HAProxy* (load balancing) with *Galera* (Relational database replication). In such a deployment, each inner-state of every service composing OpenStack is stored in a database shared between all nodes: when a controller processes a request and performs some actions, changes in the inner-state are also made on all the other controllers. From a certain point of view, it gives the illusion that only one controller is working.

Although the technique described above works fine during deployments over a limited number of geographical sites, it is showing its limitations on massively distributed deployment. For this reason, we propose to replace the relational database used in OpenStack by a more suitable storage backend that would provide a better scalability in terms of geographical distribution. Distributed Hash Tables (DHTs) are distributed systems that are able to store key/value pairs with good scalability and fault tolerance properties. In light of this, we have decided to start with the Nova service, and to check if it was possible to replace the relational database (MySQL) used in Nova by *RIAK*, a *key/value store* that extends the principles followed by the *Dynamo* database with more advanced storage and query features.

The architecture used for the Nova service has been organised in a way which ensures that each of its sub-services does not directly manipulate the database: they have an indirect acces through a service called "nova-conductor" which in turn works with an implementation of the **"nova.db.api"**

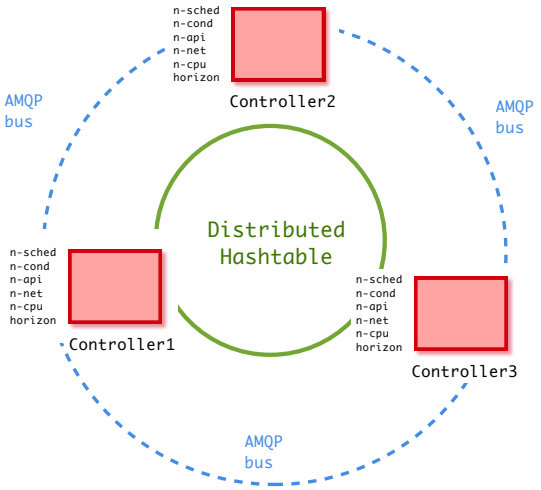

Figure 2: Nova controllers are connected through a shared key/value backend and the AMQP bus.

programming interface. Developers of Nova provide an implementation of this interface that is using *SQLAlchemy* to manipulate a relational database. We developed a second implementation of this interface that replaces every call to the *SQLAlchemy* by call to a custom RIAK driver. This enables to make Nova's services working with RIAK by only changing the database driver: this limits the level of intrusiveness in the original source code.

4.2 Validation via a proof of concept

4.2.1 Monosite

4.2.2 Multisite: 1 controller per site

4.2.3 Multisite: 10 controllers per site

5. FUTURE WORK

5.1 Locality aware entities in OpenStack

5.2 Adding a partition-aware mechanisms

6. CONCLUSION

7. REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A View of Cloud Computing. *Commun. ACM*, 53(4):50–58, Apr. 2010.

[2] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM Computer Communication Review*, 39(1):68–73, 2008.

[3] I. . E. W. Group. IEEE 802.3TM Industry Connections Ethernet Bandwidth Assessment, July 2012.

[4] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente. Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures. *Computer*, 45(12):65–72, 2012.

[5] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, and Q. Li. Comparison of several cloud computing platforms. In *Information Science and Engineering (ISISE), 2009 Second International Symposium on*, pages 23–27. IEEE, 2009.