OpenStack and Docker: building a high-performance IaaS platform for interactive social media applications

Alin Calinciuc, Cristian Constantin Spoiala, Corneliu Octavian Turcu, Constantin Filote
Faculty of Electrical Engineering and Computer Science
Stefan cel Mare University of Suceava
Suceava, Romania
{alin.calinciuc, cristian.spoiala}@usv.ro, {turcu, filote}@eed.usv.ro

Abstract— **The current development and broad adoption of cloud technologies brought in new alternative technologies that are being used to make the launch of applications in the cloud faster and easier. Interactive social media applications are faced with the challenge of efficiently provisioning new resources in order to meet the demands of the growing number of application users. This paper describes how Docker can run as a hypervisor, and how we managed to enable for the fast provisioning of computing resources inside of an OpenStack IaaS using the nova-docker plugin that we developed.**

Keywords— Software defined networking; docker; OpenStack; IaaS; cloud computing;

I. Introduction

This paper presents a free open-source technology for building high-performance IaaS platforms for interactive social media applications like NUBOMEDIA[1] by using Docker as a hypervisor inside OpenStack.

In order to allow for the deployment and handling of multiple Docker containers inside the cloud, we use the nova-docker driver to manage instances on top of physical compute nodes. Once the setup up of the platform finalized, we found that the boot time was not short enough. To solve this issue, we developed a patch for the nova-docker driver to allow for the provisioning of Docker images on all compute nodes in order to allow the interactive social media applications to scale in matter of seconds.

Over the past three years, the open source community has played a crucial role in making Docker more compatible inside OpenStack. Even though the concepts behind the containers are already mature, containers have only recently adopted an operating standard at the operating system level which has led to a repolarization of the concept.

Cloud computing is an emerging model that is now more popular than other types of resource-sharing models such as grid computing, parallel processing and distributed computing.

According to the National Institute of Standards and Technology (NIST), cloud computing is “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.” [1].

The service model presented in this paper is the Infrastructure as a Service (IaaS) model. This is outlined as the capability provided to the consumer to access and manage processing power, storage space, networks speed, and other fundamental computing resources, where the consumer is able to install and manage any software type, which can include both operating systems and personal applications. The user does not control or manage the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications, and control over the networking stack (e.g., host firewalls, tunnels) [1].

This paper presents the advantages of running containers on top of bare metal compute nodes inside IaaS platforms through the implementation of Docker as a hypervisor inside IaaS platforms based on OpenStack.

II. NUBOMEDIA Presentation

NUBOMEDIA is an elastic Platform as a Service (PaaS) cloud for interactive social multimedia. We can say that NUBOMEDIA is one of the first cloud platforms particularly designed for running interactive multimedia applications. Its design is based on media pipelines which are chains of elements that offer media capabilities, for example: video content analysis, encryption, augmented reality or transcoding. NUBOMEDIA is somehow unique because of its cloud infrastructure that acts as a single virtual super-computer incorporating all resources of available on one basic server. Because of this, NUBOMEDIA applications can adaptively scale to meet the required load, maintaining Quality of Service (QoS), Quality of Experiment (QoE) [12] and Service Level Agreement (SLA) guarantees [2].

The three main technologies used in NUBOMEDIA are Kurento Media Server (KMS), OpenBaton, OpenStack, OpenShift and the NUBOMEDIA PaaS manager.

[1] http://www.nubomedia.eu - NUBOMEDIA

KMS[2] is a WebRTC media server that provides interactive multimedia communications. OpenBaton[3] is a Network Function Virtualization Orchestrator (NFVO) that is in charge of managing the lifecycles of media servers. OpenShift Origin[4] is used for hosting applications that consume media server capabilities using Docker containers. The PaaS manager[5] provides developers with an interface for deploying their applications on top of the NUBOMEDIA platform. The entire architecture runs on top of OpenStack which is able to allow virtual machines using KVM hypervisors to run in parallel with Docker containers on bare metal.

III. OPENSTACK IAAS

OpenStack is a free open-source platform for cloud computing which enables data-center admins to control the data-centers they are running, in an easy-going manner [3].

The IaaS relies on multiple software components that have different roles. The projects that have the most important role inside OpenStack are: Nova which manages the computation, Cinder which manages the storage resources, and Neutron which manages the networking resources through an entire data-center or across multiple data-centers.

IV. DOCKER VS KVM COMPARISON

Docker is a free open platform for developers and system administrators which allows them to create, deploy, and run micro-service based applications.

With the use of Docker Engine, which is a lightweight and portable packaging tool, and Docker Hub, as service that enables the distribution of the Docker containers, Docker allows applications to be rapidly packed from different components and deployed. This eliminates the back and forwards between development, QA, and production environments. As a result, software development companies can deliver faster and run the same application, without changing it, on notebooks, data center virtual machines (VM), or and any cloud platform [4].

KVM (Kernel-based Virtual Machine) is a full software virtualization hypervisor that can be found on all Linux distribution kernels. The main hardware able to run KVM are Intel or AMD processors [13].

Unlike a VM running a full operating system, a container can only run one single process on, either Intel or AMD processors, even though, there are two types of containers:

- containers that resemble a full operating system and run their own init, inetd, etc. are called system containers;
- containers that run only one application or multiple services providing one single capability. These types of containers are called application containers.

Containers inside a physical machine do not generally have separate IP addresses, which is a huge advantage nowadays because IPv4 addresses are very limited.

Docker container images need less disk space than VM images, because the part that multiple containers have in common can be shared between them most of the time. This leads to faster deployment of Docker containers, because the images should usually be copied over the network on the compute node before the instance can start [5].

We created a tool[6] for measuring the time needed to start a Docker container vs. the time needed to start a KVM instance inside an OpenStack environment. Our measurements revealed that containers can start much faster than VMs (less than 1 second compared to 120 seconds, on our hardware) because unlike VMs, containers do not need to boot another copy of the operating system.

To conclude, administrators and developers are interested in containers for five main reasons [6] [10] [11]:

- Fast boot time - Our measurements have shown than containers can start much faster than VMs (less than 1 second compared to 2 minutes on our hardware) because unlike VMs, containers do not need to boot another copy of the operating system;
- No need to migrate containers - You can start a new one and send the traffic to the newly created instance and remove the old one, all in just few seconds;
- Application containers are lightweight, minimizing the required resources and the bandwidth needed for deployment;
- Direct access to hardware functions;
- The containers can run on any environment or hardware that is capable of running a Linux operating system.

V. CLOUD PLATFORM ARCHITECTURE

A. Standard architecture

A standard architecture of OpenStack is by default composed of the micro-services described in section III, with nova-compute having qemu-kvm as a hypervisor, as it can be seen in Figure 1.

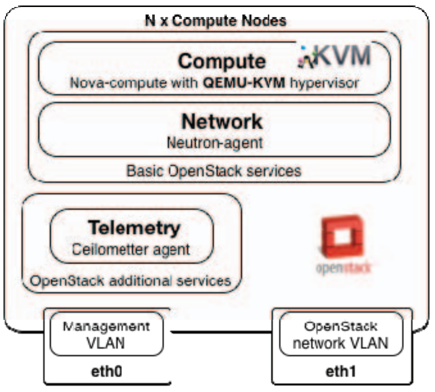

Fig. 1. Compute Node architecture

OpenStack supports a large variety of hypervisors[7]. From all types of hypervisors, containers are definitely a hot topic today and The OpenStack User Survey [7] indicates that more than half of the participants are interested in working with containers on top of their OpenStack public or private cloud. Being part of the free open source community, containers have

[2] http://www.kurento.org/whats-kurento - Kurento Media Server

[3] http://openbaton.github.io/ - OpenBaton VNFM

[4] https://www.openshift.org/ - OpenShift Origin

[5] https://github.com/fhg-fokus-nubomedia/nubomedia-paas - the NUBOMEDIA PaaS manager

[6] https://github.com/alincalinciuc/instance-boot-time-openstack

[7] http://docs.openstack.org/developer/nova/support-matrix.html

gained significant popularity in the last few years among developers and system administrators.

As it can be seen in Figure 2 generated by Google Trends, Docker is now more popular than KVM or Xen, which are the top two references in virtualization technologies.

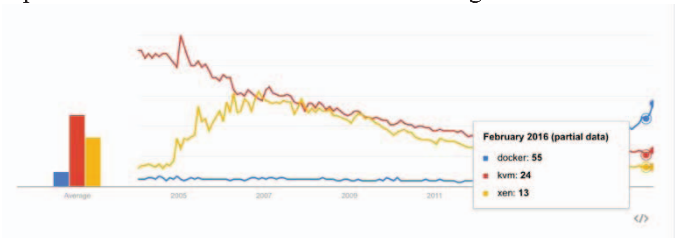

Fig. 2. Docker vs KVM and Xen in Google Trends

Our scope is to enable users to create and manage Docker containers in a consistent way with how they are used to start virtual machines using the Nova service. Furthermore, we aim to provide developers with one single set of APIs to handle their workloads, on both containers and virtual machines.

There are multiple projects that are trying to introduce the Docker container technology inside OpenStack. Some of the most important are Magnum, Kolla, Murano and Nova-Docker.

In summary:

- Magnum is providing APIs which are container specific for multi-tenant containers-as-a-service within OpenStack;
- Kolla provides active OpenStack control plane where each OpenStack micro-service can run inside a Docker container;
- Murano is designed to provide application catalog solutions that allow packaged applications to be deployed inside OpenStack, like the installation of Kubernetes [8];
- Nova-docker is a hypervisor driver for Openstack Nova Compute. It was first introduced with the Havana release, but it was also under development for Icehouse, Juno and Kilo. The driver was allowed to reach feature-parity and maturity faster than if it under OpenStack, due to its status of outside project.

Consequently, we conclude that using Docker as a hypervisor with nova-docker on top of OpenStack is the best solution for micro-service based interactive social media applications because of image size, direct access to hardware features and short boot time.

B. Configure IaaS to enable Docker as a hypervisor

Docker can be used to manage multiple containers on a single physical machine. However, when it is used in Nova, it becomes much more powerful since it becomes possible to manage several hosts, which in turn manage hundreds of containers [9].

We do not aim to replace virtual machines completely with Docker containers, but rather use them specifically when it makes more sense to use containers instead of virtual machines.

Considering the advantages of containers, we configured several compute nodes inside an OpenStack private cloud with Docker as a hypervisor, and performed several tests to prove that Docker is currently a production-ready technology on OpenStack.

To this end, we used nova-docker which is a Docker driver for OpenStack Nova being developed under Apache License 2.0. We created and added a patch on top of it in order to allow instant provisioning of any Docker container image that is available on Glance across all compute nodes.

First, in order to install and configure nova-docker, you have to configure the master node to support the addition of compute nodes running Docker as hypervisor.

Consequently, the Glance service has to be modified in order to support Docker images by adjusting the */etc/glance/glance-api.conf* configuration file and adding:

```
container_formats = ami,ari,aki,bare,ovf,docker
```

In order to allow Nova compute to decide when an instance should be running on Docker container or KVM virtual machine, the following adjustments need to be done:

- Create new flavors for the new hypervisor type;

```
nova flavor-create d1.large auto 1024 5 2
nova flavor-key d1.large set hypervisor_type=docker
nova flavor-key d1.large unset availability_zone
```

- Create new *Host Aggregates* in Nova with Docker hypervisor tag and add the compute nodes running that particular hypervisor type to the group;

```
nova aggregate-create docker nova
nova aggregate-set-metadata docker
hypervisor_type=docker
nova aggregate-add-host docker DockerCompute1
```

- Add a new filter for detecting the aggregation type needed for a specific flavor by adjusting the */etc/nova/nova.conf* on the master node;

```
scheduler_default_filters=RetryFilter,AvailabilityZoneFilte
r,RamFilter,ComputeFilter,ComputeCapabilitiesFilter,ImageP
ropertiesFilter,CoreFilter,AggregateInstanceExtraSpecsFilter
```

- After having the master node configured, a Docker image has to be added to Glance:

```
docker save alincalinciuc/ubuntu_14.04_openssh | glance
image-create --is-public=True --container-format=docker --
disk-format=raw --name alincalinciuc/ubuntu_14.04_openssh
```

At this stage, everything should be configured at the master node. The next step is to configure the compute nodes that need to have Docker as hypervisor using the tutorial available on the OpenStack wiki.

C. Nova-docker patch

Now, the only thing left to do is to have the Docker container images available on all the Docker compute nodes. Therefore, a Jenkins job or create a cron job has to be created in order to run the nova-docker patch.

The nova-docker patch is a python script that we developed, needing access to the OpenStack master in order to get a list of all Glance Docker type images. Then, it gathers a list of all compute nodes that are currently running. In the end, having the list of images and compute nodes allows the plugin to connect to each compute node from the master and to pull the latest version for each image.

This way, we are sure that we have all compute nodes ready to immediately start Docker containers at any time. The overall architecture of our testbed can be seen in Figure 3.

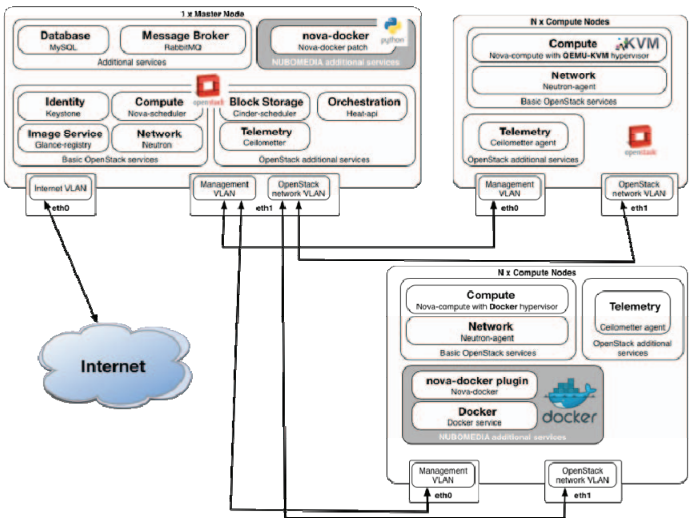

Fig. 3. OpenStack Architecture with both Docker and KVM as hypervisors

D. Interactive social media application advantages

The main advantage of having the nova-docker patch configured is that all Docker images being configured on Glance are available on all compute nodes, at any time. This enables any interactive social media application to scale very fast. In only 1 second, the running application can require and provision new workers.

VI. CONCLUSIONS

Planning an IaaS Architecture for an interactive social media application requires choosing between virtual machines and Docker containers. Containers have a huge advantage over VMs because of performance improvements and reduced startup time. The thing that is still missing is the prefetching of the Docker container images on compute nodes. In order to solve this issue and make the boot time even smaller, we developed a patch for the nova-docker driver that pulls all Docker images available on Glance on all compute nodes that are running Docker as a hypervisor. Using this tool, all images are always on the compute nodes, so the instantiation of a Docker container can be done with no delay.

The improvement gained using the nova-docker patch developed allowed us to create an IaaS platform suitable for interactive social media applications that require rapid scale.

This paper presents a solution for improving the nova-docker driver for OpenStack. In the near feature, we expect that we can integrate our patch for the nova-docker on the main project so this functionality will be available to all OpenStack users.

ACKNOWLEDGMENT

The paper and the research effort driving to it have been co-founded by the European Commission under project "NUBOMEDIA – an elastic Platform as a Service (PaaS) cloud for interactive social multimedia" with reference FP7-ICT-2013.1.6 GA-610576.

REFERENCES

[1] P. Mell, T. Grance, “The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology,” NIST-National Institute of Standards and Technology U.S., Department of Commerce, Special Publication 800-145, September 2011.

[2] A. Cheambe, M. Pasquale, F. Mulinga, B. García, M. Gallego, G. Carella, L. Tomasini, A. Calinciuc, C. Spoiala, “Design and Implementation of a High Performant PaaS Platform for Creating Novel Real-Time Communication Paradigms,” 19th International Innovation in Clouds, Internet and Networks (ICIN) Conference, March 2016.

[3] R. Sampathukumar, “Disruptive Cloud Computing and IT: Cloud Computing SIMPLIFIED for every IT”, Xlibris, 2015.

[4] D. Liu, L. Zhao, ”The research and implementation of cloud computing platform based on docker,” IEEE 11th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2014, pp. 475 – 478.

[5] W. Felter, A. Ferreira, R. Rajamony, J. Rubio, “An Updated Performance Comparison of Virtual Machines and Linux Containers,” IBM Research Report, RC25482 (AUS1407-001), July 2014.

[6] M. Raho, A. Spyridakis, M. Paolino, D. Raho, “KVM, Xen and Docker: a performance analysis for ARM based NFV and Cloud computing,” IEEE 3rd Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), pp. 1–8, November 2015.

[7] O. Foundation, "A snapshot of OpenStack users’ attitudes and deployments," October 2015. [Online]. Available: https://www.openstack.org/assets/survey/Public-User-Survey-Report.pdf.

[8] K. Cacciatore, P. Czarkowski, S. Dake, J. Garbutt, B. Hemphill, J. Jainschigg, A. Moruga, A. Otto, C. Peters, B. E. Whitaker, "Exploring Opportunities: Containers and OpenStack - White Paper," 2015. [Online]. Available: https://www.openstack.org/assets/pdf-downloads/Containers-and-OpenStack.pdf.

[9] D. Bernstein, “Containers and Cloud: From LXC to Docker to Kubernetes,” IEEE Cloud Computing, 2014, Volume 1, Issue 3, pp. 81-84, September 2014.

[10] L. Affetti, G. Bresciani, S. Guinea, “aDock: A Cloud Infrastructure Experimentation Environment based on OpenStack and Docker,” IEEE 8th International Conference on Cloud Computing, pp. 203-210, July 2015.

[11] J. Stubbs, W. Moreira, R. Dooley, “Distributed Systems of Microservices Using Docker and Serfnode,” 7th International Workshop on Science Gateways (IWSG), pp. 34 – 39, June 2015.

[12] C. Lee, S. Kim, E. Kim, "Expediting P2P Video Delivery through a Hybrid Push-Pull Protocol," Advances in Electrical and Computer Engineering, vol.15, no.4, pp.3-8, 2015, doi:10.4316/AECE.2015.04001.

[13] A. Patrascu, V.-V. Patriciu, "Digital Forensics in Cloud Computing," Advances in Electrical and Computer Engineering, vol.14, no.2, pp.101-108, 2014, doi:10.4316/AECE.2014.02017.