Journal of Network and Computer Applications 63 (2016) 140–149

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Efficient image deployment in cloud environments

Álvaro López García *, Enol Fernández del Castillo

Instituto de Física de Cantabria — IFCA (CSIC—UC), Avda. los Castros s/n. 39005 Santander, Spain

ARTICLE INFO

Article history:
Received 20 February 2015
Received in revised form
28 October 2015
Accepted 29 October 2015
Available online 5 February 2016

Keywords:
Cloud computing
Image deployment
OpenStack
Scheduling

ABSTRACT

The biggest overhead for the instantiation of a virtual machine in a cloud infrastructure is the time spent in transferring the image of the virtual machine into the physical node that executes it. This overhead becomes larger for requests composed of several virtual machines to be started concurrently, and the illusion of flexibility and elasticity usually associated with the cloud computing model may vanish. This poses a problem for both the resource providers and the software developers, since tackling those overheads is not a trivial issue.

In this work we implement and evaluate several improvements for virtual machine image distribution problem in a cloud infrastructure and propose a method based on BitTorrent and local caching of the virtual machine images that reduces the transfer time when large requests are made.

© 2016 Elsevier Ltd. All rights reserved.

1. Introduction

As it is widely known, the Cloud Computing model is aimed on delivering resources (such as virtual machines, storage and network capacity) as an on demand service. The most accepted publication defining the Cloud from the United States National Institute of Standards and Technology (NIST), emphasizes the *rapid elasticity* as one of the essential characteristics of the Cloud Computing model: "capabilities can be elastically provisioned and released, (…), to scale rapidly outward and inward (…)" (Mell and Grance, 2011). Moreover, users and consumers consider them as the new key features that are more attractive (Zhang et al., 2010; Armbrust et al., 2010) when embracing the cloud.

If we take into consideration virtual machines delivered by an Infrastructure as a Service resource provider, these two outstanding features imply two different facts. On the one hand, *elasticity* is the ability to start and dispose one or several virtual machines (VMs) almost immediately. On the other hand, an *on demand* access implies that VMs are allocated whenever the user requires them, without prior advise and without human intervention from the Resource Provider (RP).

Any cloud must be able to deliver rapidly the requested machines to provide a satisfactory elastic and on-demand perception according to any Service Level Agreement SLA (Aceto et al., 2012) established with the users or customers. With this fact in mind, the Cloud Management Frameworks (CMFs) —and as a consequence the resource providers operating a cloud— face a challenge when they are requested to provision a large number of resources, specially when running large infrastructures (Chen et al., 2009b).

These on-demand and elastic perceptions that a cloud should be able to deliver mostly depend on the time needed to serve the final service, so a rapid provisioning should be one of the objectives of any cloud provider. Besides the delays introduced by the Cloud Management Framework there are other factors contributing to this delay from the user standpoint. For instance, inter-datacenters transfers of large amounts of data (Femminella et al., 2014) are a good example of contributors to the final delivery time. Any reduction in each of these factors will yield on a better reactivity of the cloud, leading to an increase of the ability to satisfy elastic requests on-demand.

Therefore, in order to deliver a rapid service, this spawning delay or penalty has to be decreased. It is the duty of the cloud provider to be able to provision efficiently the resources to the users, regardless of the size of the request, minimizing the costs of mapping the request into the underlying resources (Manvi et al., 2014). Hence, it is needed to study how current CMFs can minimize the start time of the virtual machines requested. Our main contribution in this paper is the proposal of an improvement of the current Coud Management Frameworks in two sides: firstly, the CMFs should implement more advanced and appropriated image transfer mechanisms; secondly, the cloud schedulers should be adapted so as to make use of local caches on the physical nodes. Moreover, in this paper:

- We will study how the deployment of images into the physical machines poses a problem to an Infrastructure as a Service

* Corresponding author.
E-mail addresses: aloga@ifca.unican.es (Á. López García), enolfc@ifca.unican.es (E. Fernández del Castillo).

http://dx.doi.org/10.1016/j.jnca.2015.10.015
1084-8045/© 2016 Elsevier Ltd. All rights reserved.

(IaaS) resource provider and how it introduces a penalty towards the users.
- We will discuss several image transfer methods that alleviate this problem and review the related work.
- We implement and evaluate some of the described methods in an existing CMF.
- We propose an improvement of the scheduling algorithm to take profit of the VM images cached at the physical nodes.

The paper is structured as follows: In Section 2 we discuss and present the problem statement that. In Section 3 the related research in the area is presented and discussed. Section 4 contains the evaluation of some of the methods described in the previous section. In Section 5 we propose a modification to the scheduling algorithms, and evaluate in combination with the studied image transfer methods. Finally, conclusions and future works are outlined in Section 6.

2. Problem statement

Whenever a virtual machine is spawned its virtual image disk must be available at the physical node in advance. If the image is not available on that host, it needs to be transferred, therefore the spawning will be delayed until the transfer is finished. This problem is specially magnified if the request consist on more than a few virtual machines, as more data needs to be transferred over the network. As the underlying infrastructure increases its size the problem becomes also bigger, the number of requests need to be satisfied may become larger.

Regardless of the Cloud Management Framework (CMF) being used, the process of launching a VM in an IaaS cloud infrastructure comprises a set of common steps:

1. A VM image is created by somebody —e.g. by a system administrator or a seasoned user—, containing the desired software environment.
2. The image is uploaded to the cloud infrastructure image catalog or image repository. This image is normally stored as read-only, therefore, if further modifications (for example any user customization) need to be done on a given image, a new one must be created.
3. VMs based on this image are spawned into the physical machines.
4. The running VMs are customized on boot time to satisfy the user needs. This step is normally referred as contextualization and it is performed by the users.

The first two steps are normally performed once in the lifetime of a virtual machine image, meaning that once the image is created and is available in the catalog, then it is ready for being launched, so there is no need to recreate the image and upload it again. Therefore, assuming that the IaaS provider is able to satisfy the request (i.e. there are enough available resources to execute the requested VMs), whenever a user launches a VM, only the two former steps will introduce a delay in the boot time.

The last step, that is, the contextualization phase is made once the virtual instance has booted, and it is normally a user's responsibility and beyond the Cloud Management Framework control (Campos et al., 2013; Li et al., 2012). Hence, the field where a IaaS resource provider can take actions to reduce the boot time of a virtual machine is the spawning phase. This phase involves several management and preparation operations that will depend on the Cloud Management Framework being used. Generally, these operations will consist on one or several of the following steps:

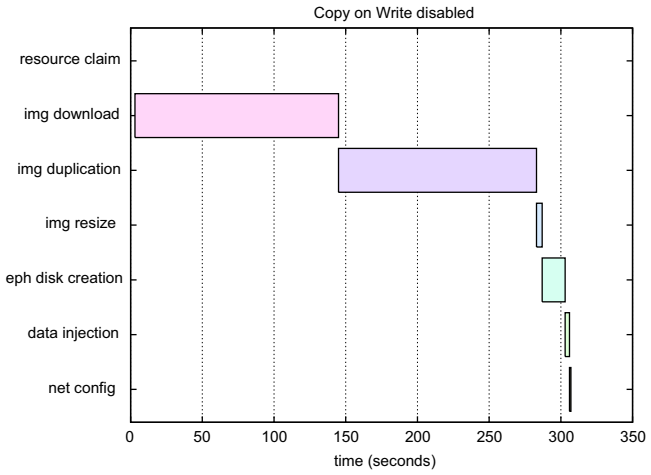

Fig. 1. Chart of the boot process for one VM on an OpenStack cloud. The image used was 10 GB large with an 80 GB ephemeral disk.

Scheduling phase: where the software selects the most suitable nodes to satisfy the user's request.

Image transfer: if the image data is not available in the selected physical machine, the CMF has to transfer it from the catalog into that host.

Image duplication: once the image is available at the node. Some CMF duplicate the image before spawning the virtual machine. This way, the original image remains intact and it can be reused afterwards for another VM based on that same image.

Image preparation: consisting in all the further image modifications prior to the virtual machine spawning, needed to satisfy the user's request. For instance, this step can comprise the image resize, image format conversion, user-data injection into the image, file system checks, etc.

Taking as an example the OpenStack cloud testbed described in the experimental setup of Section 4.1, Fig. 1 shows the boot sequence for an instance once the request is scheduled into a physical machine. In this request a 10 GB image was launched with an additional local ephemeral disk of 80 GB. This ephemeral empty space is created on the fly on the local disk of the physical machine, therefore it is not transferred over the network. In this initial setup, the images are stored in the catalog server and are transferred using HTTP when they are needed in the compute host.

As it can be seen, the OpenStack spawning process is broken down into several sub steps:

Resource claim: The compute node checks if the requested resources are available, and claims them before spawning the instance.

Image download: The image is fetched from the image catalog, and it is stored in the local disk.

Image duplication: An exact replica image is created from the downloaded one.

Image resize: The image is resized to fit into the size request by the user. Normally minimal images are stored in order to spare disk and save transfer times, therefore these images need to be resized into the correct final size.

Ephemeral disk creation: An ephemeral virtual disk is created in the local disk. This virtual disk is created on the fly and it is normally located on the local machine disk, since it is a disposable space destroyed when the instance is terminated.

Download English Version:

https://daneshyari.com/en/article/457122

Download Persian Version:

https://daneshyari.com/article/457122

Daneshyari.com