Cloud Agnostic Virtual Machine Image Service

João Pereira and Paula Prata

Instituto de Telecomunicações (IT)
Department of Informatics, University of Beira Interior
Rua Marquês d'Ávila e Bolama, 6201-001 Covilhã, Portugal
joaodrp@it.ubi.pt,pprata@di.ubi.pt

Abstract. Facing the Cloud Computing paradigm and the role of virtualization, the management of virtual machine images is becoming increasingly important. Images are delivered by Infrastructure-as-a-Service frameworks to hypervisors that create virtual machines based on them. Most infrastructure frameworks save images in a storage system that serves as image repository. However, problems arise when managing environments with different cloud infrastructures, as their image repositories are tied to their own requirements. This article presents and evaluates an image server comprised in VISOR, a cloud agnostic virtual machine images management service.

Keywords: Cloud Computing, Virtual Machine Images, Web Services, REST, Event-Driven Programming, Storage Systems

1 Introduction

Cloud Computing has allowed us to access a wide range of machines as virtual machine (VM) instances. Since virtualization is the foundation of cloud Infrastructure-as-a-Service (IaaS), the management of large amounts of VM images can become an exponential bottleneck. Although most IaaS frameworks provide an image repository service to manage VM images, such services are tied to the IaaS requirements and implementation details. This leads to integration and interoperability issues when managing multiple heterogeneous IaaS.

Currently there are many IaaS offers, among which we highlight Amazon Web Services (AWS) [1], Eucalyptus [8], Nimbus [2] and OpenStack [3]. Among these, only OpenStack isolates the image management functionalities in an image service, called Glance. In AWS, images are stored in the Amazon Simple Storage Service (S3). Eucalyptus provides the Walrus storage service, in which VM images are stored. Nimbus uses its own storage system called Cumulus [4] to store VM images, which can also rely on the Apache Hadoop Distributed Filesystem (HDFS) [11] as the backend filesystem. OpenStack (through Glance) stores images in its distributed storage system called Swift, Amazon S3 or on a local filesystem. There is also the FutureGrid (FG) platform [7], an interesting proprietary test-bed for scientific projects. FG integrates an image service (FGIR) in its architecture, in order to manage VM images across different internally supported IaaSs, used to deploy cloud test-beds in the FG platform.

Unlike existing solutions, VISOR is not intended to fit in a specific IaaS framework (as Glance) but rather to overreach interoperability limitations across different IaaSs and storage systems. It is also not a proprietary service (as FGIR) but rather an open source project (`http://www.cvisor.org`). It aims to provide a flexible metadata schema, multiple data communication formats, high modularity, performance and compatibility with multiple IaaS storage systems. We have targeted a set of IaaS and their storage systems, but given the system modularity, it is possible to easily extend it with other systems compatibility.

VISOR is implemented as a set of distributed Web services that we call "subsystems". The three main subsystems are the VISOR Meta System (VMS), VISOR Auth System (VAS) and VISOR Image System (VIS). VMS is responsible for managing image metadata, which is a set of attributes describing a VM image. The VAS is responsible for managing users accounts. Finally, VIS is the service main subsystem and clients' front-end. It communicates with VMS and VAS in order to manage image metadata and authenticate requests, respectively. It is also responsible for transferring images between clients and backend IaaS storage systems. VMS and its metadata schema were already described in a previous paper [10]. The description of VAS is out of the scope of this paper, in which we will focus on the description of the VIS.

2 VISOR Image System

The VIS architecture is represented in Figure 1. VIS comprises three main layers: client interfaces, server application and the storage backend abstraction. Besides the VIS internals, there are also the configuration and log files. VIS loads its settings from a configuration file, which users should customize. This file includes configuration parameters for the server application (i.e. host and port address to bind the server, log file to use, log level) and for the compatible storage backends (e.g. user's access credentials, host address, bucket to store images in).

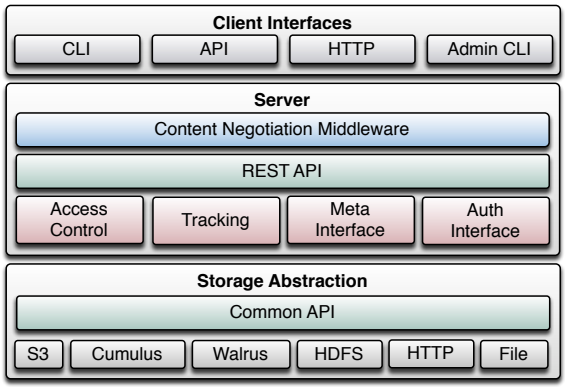

Fig. 1. VISOR Image System layered architecture.

2.1 Client Interfaces

In VISOR there are four distinct types of clients: users, developers, external services and administrators. Users interact with VIS through the provided client command-line interface (CLI). Developers are those interacting with VIS through its programming Application Programming Interface (API). External services directly interact with VIS through its HTTP Representational State Transfer (REST) [5] interface. Administrators can use the administration CLI to manage the VIS server status (i.e. start, stop, restart).

2.2 Server Application

VIS supports multiple data communication formats trough content negotiation with seamless responses encoding, both for metadata and error messages. This process is done through the *content negotiation middleware*. It negotiates and encodes the service responses based on the HTTP request Accept header. By now, the service provides compatibility with JSON and XML. Additional format encoders can be easily plugged into the system, since they are implemented as isolated middleware components. The *REST API* exposed the service functionalities abroad. It includes operations for registering, listing, searching, retrieving, updating and deleting VM images metadata and associated files. At registering time, users can choose in which compatible storage system the image file should be saved in. The *access control* component is responsible for controlling user accounts and sharing of VM images, by leveraging in access permissions. The *tracking* module is responsible for tracking operations through the service and record that information, creating a full history of each VM image life cycle. It is also intended to generate statistical data, useful for providing a high level overview of the repository usage. The *meta and auth interfaces* are APIs for internal use, towards the VIS communication with VMS and VAS, respectively. Whenever VIS needs to process some metadata, it calls the meta interface in order to communicate with VMS. In the same way, whenever it needs to authenticate a request, it uses the auth interface to communicate with VAS. The VIS server is an event-driven application. Thus it is responsible for tracking I/O operations status and managing resource's availability in a non-blocking manner. By avoiding common multi-threading problems [6, 9], event-driven applications provide I/O concurrency through high performance non-blocking operations.

2.3 Storage Abstraction

Under the server layer is the storage backend abstraction, providing seamless integration with multiple storage systems through independent plugins. On top of these plugins is a common API, which seamlessly abstracts such storage systems heterogeneity. VIS interacts with the common API to accomplish image files management requests, acting as a bridge between storage systems and clients. Currently, this layer provides compatibility with Amazon S3, Nimbus Cumulus, Eucalyptus Walrus, Apache HDFS, a read-only HTTP backend and the server

local filesystem. From the initially targeted systems, VISOR yet not supports the OpenStack Swift. Storage plugins are isolated from the service API, thus it is easy to extend the service with support for additional storage systems.

3 Conclusions

We have presented our work towards a cloud agnostic image server (VIS), which is the main subsystem of the VISOR service. VISOR has been actively improved to address the need of having a solid, centralized and generic VM image repository, exposing the service abroad to heterogeneous IaaSs and their underpinning storage systems. We have relied in highly scalable technologies like event-driven frameworks in order to achieve high availability and performance. Performance tests (which will be presented in a future publication) have shown VISOR as a high performance service. The service is being actively developed as an open-source project, exposing the system to potential contributions. The source code and further informations about VISOR can be found at the project home page `http://www.cvisor.org`.

References

1. Amazon web services, `http://aws.amazon.com/`
2. Nimbus, cloud computing for science, `http://nimbusproject.org`
3. Openstack: Open source software for building clouds, `http://openstack.org`
4. Bresnahan, J., LaBissoniere, D., Freeman, T., Keahey, K.: Cumulus: an open source storage cloud for science. In: Proceedings of the 2nd international workshop on Scientific cloud computing. pp. 25–32. ACM (2011)
5. Fielding, R.T.: Architectural styles and the design of network-based software architectures. Ph.D. thesis, University of California, Irvine (2000)
6. Flanagan, C., Freund, S.N.: Fasttrack: efficient and precise dynamic race detection. SIGPLAN Not. 44, 121–133 (2009)
7. von Laszewski, G., Diaz, J., Wang, F., Younge, A.J., Kulshrestha, A., Fox, G.: Towards generic futuregrid image management. In: Proc. of the 2011 TeraGrid Conf.: Extreme Digital Discovery. pp. 15:1–15:2. TG '11, ACM, NY, USA (2011)
8. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: The eucalyptus open-source cloud-computing system. In: Proc. of the 2009 9th IEEE/ACM Int'l Symp. on Cluster Computing and the Grid. pp. 124–131. CCGRID '09, IEEE Computer Society, Washington, DC, USA (2009)
9. Pariag, D., Brecht, T., Harji, A., Buhr, P., Shukla, A., Cheriton, D.R.: Comparing the performance of web server architectures. In: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007. pp. 231–243. EuroSys '07, ACM, New York, NY, USA (2007)
10. Pereira, J., Prata, P.: Visor: Virtual images management service for cloud infrastructures. In: The 2nd International Conference on Cloud Computing and Services Science, CLOSER. pp. 401–406 (2012)
11. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). pp. 1–10. IEEE (2010)