Implementing the Cloud Software to Data approach for OpenStack environments

Lenos Vakanas, Stelios Sotiriadis, and Euripides G.M. Petrakis

Technical University of Crete (TUC), Intelligent Systems Laboratory (InteLLigence), Chania, 73100, Greece
lenosvakanas@gmail.com, (s.sotiriadis, petrakis)@intelligence.tuc.gr

Abstract. Cloud computing offers a development platform with many benefits such as low cost application development and deployment along with minimization of maintenance and upgrades. Despite the technologys numerous advantages, health care and other application fields related with sensitive and confidential information have been reluctant to seize its offerings. This is because of the requirement for data processing data on remote cloud datacenters and therefore the transferring of sensitive data over the Internet. A solution to this problem is the reverse cloud approach that allows software to be transferred near to the data source and to be instantiated into a new cloud environment in order to eliminate the problems of processing sensitive data remotely. To achieve this we developed an innovative software to data service that allows virtual machines in the form of running instances or images to be migrated between OpenStack environments. Further, the service allows easily reconfiguration (regarding hardware features) along with monitoring and calculating of virtual machine use in the OpenStack federation.

Keywords: Cloud Computing, Inter-Cloud, Virtual Machine Migration, FIWARE, OpenStack, OpenStack Migration

1 Introduction

Cloud computing is an emerging technology that provides virtualized resources on a pay per use and on demand mode. It offers great advantages over the traditional solutions such as flexibility and elasticity that allows adaptation of the computing resources (i.e. cpu, memory, hard disk, bandwidth) to the actual needs of an application and easier application deployment (in Platform as a Service) using APIs of pre-installed software modules referred as Generic Enablers (GEs). It further encompasses business models for automatic resource management through flexible user interfaces and APIs and also Quality of Service (QoS) control as cloud aim to ensure optimum QoS to customers including, reasonable response time, load throughput measuring, security and privacy based on predefined Service Level Agreements (SLAs). Today, various cloud deployment models have been produced in order to meet the diverse needs of the digital economies and the various types organizations[4]. The most common are the (a) Public

Cloud, where resources usually owned and managed by enterprises or organizations are available to general public, (b) Private Cloud where resources are exclusively owned and operated by a single entity for satisfying the needs of the enterprise owner, (c) Community Cloud that serve consumers (e.g., organizations or individuals) sharing common interests or concerns and usually it can be owned and managed by someone in the community or by a third party and (d) Hybrid Cloud that is the combination of a public and a private cloud provides the ability, for an entity, to own and manage a private cloud and use functionality offered by public cloud providers by leveraging the advantages of both models.

Despite the benefits of cloud computing, certain applications are still reluctant and unwilling to use this novel technology due to the sensitivity nature of the data that is in their possession (e.g., patient data in health care). In particular, health care providers cannot accept patient data transferred over the Internet or data storage in remote locations because of legal or ethical restrictions (i.e., data privacy and trust). A vital requirement is that health care providers are required to process patient data on their own premises. Application owners with similar concerns are businesses or government related organizations and entities that manage sensitive or classified data that are reluctant to cloud based solutions due to security risks of transferring data over the Internet. Similarly, scientific or research organizations might express similar concerns. For these entities, the major problem is the amount of data that have to be transferred over the Internet as it might be costly or it may take much transfer big amounts of data.

In this work we approach the problem of Virtual Machine (VM) migration over OpenStack[1] cloud environments to allow virtuazlized software that will process data in its source, by utilizing a hybrid cloud model [9]. The approach also known as reversed cloud approach" [8] builds upon the idea of bringing the software to the data rather than transferring the data to the public cloud where the software is installed.

To achieve it, we utilize a cloud federation architecture composed by a public cloud which is the software provider, a private cloud which is the consumer and an OpenStack based service that will transfer and deploy the software from the public to the consumer cloud. The cloud federation offers significant advantages such as increased elasticity [10] and service diversity [7]. Next the consumer can process data locally and pay the usage fees to the provider based on a software to data bespoke model. Based on this solution a provider (e.g. a health care organization) can utilize cloud technology and offerings of public clouds, without breaking the restrictions related to data privacy. This work proposes and implementats the Software to Data (S2D) OpenStack service that provides mechanisms to transfer, deploy and monitor the software from a public cloud provider to a private cloud where the data owner resides. The implementation consists of two modules, firslty the module being responsible for transferring and deploying the software between the public and private clouds and secondly, the module responsible for monitoring the usage of the software including up-time

[1] https://www.openstack.org

and usage information providing this to service provider (e.g., for accounting and billing purposes).

The proposed S2D solution is deployed using the Intellicloud[2] and FIWARE filab infrastructures as testbed system to demonstrate interactions between public and private clouds. Both clouds are deployed as OpenStack systems and FIWARE datacenter resource management system (on top of Openstack) so are compatible, homogeneous and easily interoperable. The S2D solution interacts with the private and public clouds using a REST API. In Section 2 we descibe the realated approaches and the motivation of this work. In Section 3 we present the modelling of the S2D service. The rest of the paper is organized as follows. In Section 4 we present the description of the proptotype solution that includes a foundamental performance evaluation followed by conlusions and issues for future research in Section 5.

2 Related Approaches

Cloud computing has been advanced as a technology providing virtualized resources to Internet users based on a bespoke manner based on hardware, software and platform that could be delivered as a service. Different cloud models include (a) Infrastructure as a Service (IaaS), that is the ability to pay for use of hardware such as storage, computing power or network where, the consumer is responsible for installing and maintaining the operating system and the provider take cares of upgrades and maintenance of the hardware infrastructure, (b) Platform as a Service (PaaS) that provision platform components to consumers including basic tools and software services for deploying applications and (c) that allows a consumer to use services provided by the cloud provider or even by other consumers [6]. For this cloud model, the consumer has no control over the services software or hardware but can only use it thought provided APIs or interfaces of the service provider.

Cloud computing utilizes virtualization that provides a layer of abstraction between the hardware and the software. The product of virtualization is called VM and represents a fully functional virtualized system where cloud developers and users can use to develop their services [5]. A cloud provider may have certain amount of servers and network bandwidth but with virtualization allows serving a larger number of consumers with diverse service demands. As a result, clouds are able to efficiently exploit their computing power. For many areas such as industry, agriculture etc. this has been proven to be an efficient solution with regards to the minimization of operational costs and increased elasticity; yet not in the healthcare domain. Data stored in cloud are usually available over the Internet and could contain confidential and private health information. This has become an issue to the dissemination of cloud solutions in health care, in particular a large scale commercial solution for the health care industry that was based on a public cloud technology provided by Google had to be abandoned in

[2] http://www.intelligence.tuc.gr

2012[3] since there were issues with sensitive data processing and storage.

Today there are various standards, regulations and recommendations such as national legislation, ISO standards (ISO 80001[4]) and the need to comply with security standards (ISO 27000[5]), thus there are severe restrictions to data transfer and storage. As a result, cloud computing based on the Internet and it openness, becomes a hurdle to its adoption in health care. To overcome it, the FI-STAR project[6] designs and implement suitable software to data solutions based on Generic Enabler (GE) technology provided by FIWARE [7] to build secure healthcare applications. This work is motivated by the FI-STAR project and moves a step forward by focusing on the migration of VMs among OpenStack clouds and to an easy to deploy and monitor cloud service.

FIWARE is an innovative, open cloud-based infrastructure for cost-effective creation and delivery of FI applications and services named GEs. Another related project is the XIFI FP7[8] that facilitates a development and deployment environment where a federation of several cloud environments (OpenStack based) is taking place. Here different cloud model services provided by a common platform that aims to highlight the way for a unified European marketplace that expolits the FI concept [3]. A more detailed discussion of the literature approaches for cloud federation and inter-cloud could be found in [7] and in [6]

We expect that FIWARE could serve as the required framework to develop a software to data solution based on a real-time solution utilizing a Software Oriented Architecture (SOA) [1]. This is based on the nature of the GE implmenetation (to be packed as a VM instance) and on the standardized solution of the XIFI federation with regards to the OpenStack cloud platform. GEs are considered as software modules that offer various functionalities, protocols and interfaces for operation and communication that includes the cloud management of the infrastructure, the utilization of various IoT devices for data collection and the provision of APIs (e.g. tools for data analytics) and communication interfaces (e.g., gateways etc.). GEs are implementations of open specifications of the most common functionalities that are provided by FIWARE and are stored in a public catalogue, thus developers easily browse and select appropriate API interfaces to use. Since FIWARE is based on OpenStack, the software to data solution utilizes OpenStack interfaces to implement its features. Openstack cointains various components that are services that manage the cloud resources as discussed in [2]. These are as follows:

1. Nova Service to manage the pool of hardware resources.
2. Quantum Service to manage the clouds networks and IP addresses. Users can use this system to create their own networks with Internet access.

[3] http://www.google.com/intl/en_us/health/about/
[4] http://www.iso.org
[5] http://www.27000.org
[6] https://www.fi-star.eu/fi-star.html
[7] http://www.fiware.org
[8] https://fi-xifi.eu

3. Glance Service for managing service of images in the cloud. Clouds images include both operating system and additional software.
4. Cinder for block storage used in Openstack that allows the users to request and consume those resources via a self-service API.
5. Keystone to perform authentication system of the cloud and it manages the users.
6. Horizon Service as user interface to control and use the above mentioned services. Horizon is a web-based dashboard where administrators and users can manage and view their resources.
7. Ceilometer as a metering and monitoring service aimed to provide all the necessary measurements to establish a reliable and accurate billing system,
8. Heat as the orchestration engine to provide the ability to launch multiple cloud applications based on templates.

3 The Software to Data (S2D)

This section presents the modeling of the Software to Data (S2D) service that includes the various actors and environments interacting when migrating a VM for instance a public to a private cloud service provider. It should be mentioned that the S2D service utilizes the Openstacks REST API to allow users to perform migration. Next we present the design of the service (Section 3.1), its functionality (Section 3.2) and its implementation (Section 3.2).

3.1 Modelling the Software to Data (S2D)

S2D is designed as a modular cloud SaaS in order to allow easy deployment and configuration using API interfaces. In detail, it is designed to allow migration configuration using either a web-based user interface or an xml file containing all the needed information. In the first case, a user perform migration using the web interface following various configurations in order to setup the service. Also, the xml document has a predefined structure validated by an xml schema and the user requires to fill all the fields which will be used by the service for configuration purposes. The migration service executes three operation:

1. The mode being responsible for transferring and deploying the software from public to private cloud.
2. The mode responsible for monitoring the usage of the software (for instance its up-time).
3. The mode that allows the usage of the S2D as an API that offer the user an alternative way containing all the needed information in an XML document.

As mentioned before S2D is designed to allow efficient migration as the process of moving a running instance (and its intilial configuration e.g. pre-installed software) from the public cloud to the private cloud while maintaining its hardware, software and network configurations. After migration the private cloud will inlcude a ready running instance in respect to setup, rules, security group and a

public IP. The S2D model is composed by various actors such as the user that utilizes the front-end interface to setup the (using GUI or the XML file) the migraiton configuration, the S2D service that allows VM migration (instance of image) and the back-end system that includes the public and private cloud as demonstrated in Figure 1.

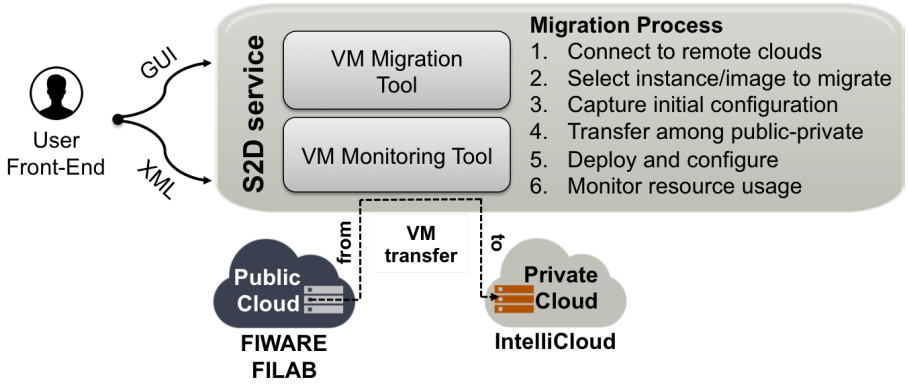

Fig. 1. The S2D model

The following list details the specification of each actor.

- The front end of the service is the entry point of the user where he/she can see and control which are the Graphical User Interfaces (GUI) and the information being provided to the service. Information is transferred from the front end to the back end using REST API calls performed by the UI directly to the service API.
- The UI allows users to interact with the service in order to migrate and monitor an instance. Specifically, the user, through the GUI, uses the instance migration tool in order to transfer an instance and the monitoring tool to get the usage data of his instances.
- The instance migration tool performs the following functionalities:
  1. It provides the necessary features to a user in order to migrate the instance along with guidance through the process. The layout of the migration tool is a set of numbered steps where during those steps the user has to provide the required information in order for the service to be able to perform the actions on the public and private cloud.The user has to follow the steps sequential in order to have a successful migration.
  2. The migration tool contains the necessary functionality to perform an instance migration between the two Openstack clouds. As discussed above, the user completes various steps for a successful migration with each step asking for the required information for a specific action on the cloud. The migration tool processes the information provided by the user on each step and performs the corresponding action on the cloud. First, the migration tool will do any processing needed on the information before performing the request on the cloud and after a successful response it

may save temporarily some information of the request and/or response for future use.

- The instance monitoring tool performs the following functionalities:
  1. This tool allows the user to get the usage data of every instance he/she owns. The user only needs to provide the information in order to be authenticated by the clouds system and the service fetches the usage data for his instance using the OpenStack API[9]. The usage data includes uptime of each instance, number of cores in use, CPU hours, storage used and other useful metrics.
  2. The user can get the usage data of his/her instances by providing the authentication credentials. This tool performs two API calls to the public and private cloud system respectively, including the authentication of the user and the action to retrieve all the usage data of the users instances.
- The back end of the service offer all the functionality of the service. The public cloud back end stores temporarily the image selected to be migrated to the private cloud.
- Besides the user interface, the service provides the ability to migrate instances by performing an API call containing all information in an xml document. This document is designed to have a specific syntax and contains all the information which the service will need to perform the necessary actions on the two clouds and migrate the user's instance. As mentioned above, this API call contains the functionality to validate the xml document, extract the information from it and feeds that information to the migration tool. The process follows the same path as the user does from the GUI, meaning this tool performs API calls on the migration tool in the same order and information.

3.2 Functionality of the Software to Data (S2D)

To achieve VM migration, the service guides the user through the process and at the same time performs automated actions resulting in a less complex time consuming process. S2D is realized by means of the following functions:

1. Authentication: The user provides a tenant ID, username and password in order to be authenticated to the cloud system. The authentication process generates a token which is being used by every action of the user (or the service itself) and is a unique for every user and it has a lifecycle predefined by the cloud. Usually a token lifetime is valid for 24 hours.
2. Get Instances: The service retrieves a detailed list of the instances registered to the user.
3. Get Images: The service retrieves the images available to the user including public snapshots.
4. Get Instances Details: Retrieves the information which describes an Instance (reagrding hardware and network).

[9] http://developer.openstack.org

5. Create Snapshot: Creates a snapshot of the running instance which the user selected for migration. In addition, the service stores all the properties of this instance which will be later be used for launching the new instance in the private cloud with the same configurations such as security group rules or flavor. For this action, the user provides a name for the snapshot and the name of the instance he/she wants to build the snapshot.
6. Download Snapshot: The service downloads and stores temporarily the previously created snapshot.
7. Upload Snapshot: Because a snapshot is also an image, the service creates a new image containing the data of the previously downloaded snapshot. The user provides the name of the new image, the format and its accessibility factor (public or pricate image). The service will perform the following actions automatically based on the initial user configuration.
   - It will create a new blank image with the given name as detailed by the user.
   - It will update the blank image according to the format (e.g. qcow2) and access (e.g. public) specified by the user.
   - It will upload he snapshots data to the new image that is information about the instances operating system and software which will be used to launch the same instance in the target cloud.
8. Keypair Actions: The keypair is the private key of the user for remote connection must be allocated to the user and provided to the new instance before its creation. As a result, the service will save the name of the keypair that the user chooses.
   - Create Keypair: The user provides the name of the new keypair and the service will create a new one with that name and allocate it to the user.
   - Import Keypair (*from public cloud*): The user selects the keypair from the public cloud and the service imports it in cloud the private cloud. In detail, the service firsts retrieves the list of users keypairs from public cloud and after the selection of the user, imports it in the private cloud.
   - Select Keypair (*from private cloud*): The service retrieves the list of keypairs allocated to the user so the user could selects one.
9. Launch Instance: The user provides the name of the instance and its security group name. Afterwards, the service will automatically set the following.
   - Keypair: The keypair of the instance that has been previously selected by the user.
   - Image: The image will be the new image which the service has been created from the data of the snapshot.
   - Flavor: The flavor defines the computational resources required for an instance. The service will fetch it from public and set the same flavor in private cloud.
   - Security Group: The service will either create a new security group (with the specification regarding ports) or will allocate to the instance the default one depending on the users input. If the user sets the name as default or in case of empty selection, the service will use the default security group. If the user sets a different name, the service will generate

a new security group with that name. The rationality behind the service is that all actions will allow an instance to be launched in private cloud and immediately be operational (including all of its software) without any further configuration by the user. Independent of the users input, the service will automatically:
- Get the security group that the instance had in public cloud.
- Get all of the rules defiened within that security group.
- Insert all of the rules in the security group that the user selected for the instance in private cloud.

10. IP Number Actions: The user can choose from a list of availalbe IPs or could choose to create a new one to allocate it to the new instance. Available IP is one that belongs to the specific user but it is not allocate to any other of his instances. If the user chooses to create a new IP, the service will perfom the following automatically:
   – Creation of a new IP.
   – Allocation of the IP to the users account.
   – Allocation of the new IP to the instance.
11. Instance Overview: The service fetches all the information about the instance which the user created. The fetched information is directly collected by the clouds ensuring the success of the process.
12. Get Instances Usage Data: The user provides his/her cloud credentials and after a successful authentication by the clouds system, the service fetches the usage data of the instances.
13. Reset System: This action allows the user to reset the state of the service. This means that the users session will be deleted along with any information he provided till that point, including any images downloaded. However, any actions that had already been executed on any of the two clouds are not affected. This mean that if the user wants an action done on a cloud reverted, he has to do it through the clouds dashboard (e.g. deleting a snapshot).

3.3 Implementing the Software to Data (S2D)

The migration tool communicates with the clouds iby performing calls on Openstack components which are the Nova compute service, the Glance image service, the Keystone identity service and the Quantum networking service. These properties (endpoints and ports) are not the same for every cloud so the service will need to configure it using the migration tool. Next we describe the steps to allow migration (Subsection 3.3.1, the implemenation of the instance monitoring tool (Subsection 3.3.2) and the description of the API call using an XML document (Subsection 3.3.3).

3.3.1 S2D: Steps to perform migration: The following demonstrate the steps required to perform migration between the public and private cloud.

Step 1 Public cloud authentication: The user configures the public cloud interactions for authorization purposes with the next deatils

- Parameters: Cloud endpoint that is the IP of the cloud, Tenant identification (usually related to the project name of the user OpenStack system), Username and Password.
- Description: The service prepares and performs the call for authentication to the public cloud which that the user specified and after successful authentication the service displays the details of the user (Username, Tenant, Source Cloud). The following shows the API call to get a token.
- Keystone: POST, http://(IP):(port)/v2.0/tokens for v2.0 OpenStack

Step 2 Creation of a Snapshot: The migration tool generates a snapshot of the selected instance with the next details.

- Parameters: Selected instance details, Snapshot Name
- Description: The service retrieves and displays the instance list owned by the user. Then, the user selects an instance that he/she wants to create a snapshot. After the cloud system has successfully created it, the service displays the name of the snapshot and instance name but also saves the instance security group and flavor details for later use in the creation of the new instance at the target cloud.The following demonstrate the OpenStack API call specification URLs.
- Nova - Compute: Instance list: GET, http://(IP):(port)/v2/(Tenant _ID)/servers/detail
- Nova - Compute: Create snapshot: POST, http://(IP):(port)/v2/(Tenant _ID)/servers/(Instance_ID)/action
- Nova - Compute: Security Group: GET, http://(IP):(port)/v2/(Tenant_ID)/servers/(Instance_ID)/os-security-groups
- Nova - Compute: Flavor: GET, http://(IP):(port)/v2/(Tenant_ID)/flavors /(Flavor_ID)

Step 3 Snapshot download: The migration tool downloads the instance by tranfroming temporary in a snapshot.

- Parameters: Details of the selected image
- Description: The service retrieves and displays the image list owned by the user including snapshots. The user selects the snapshot he/she previously created and downloads it. After the download is performed, the service displays the name of the image and its size. The downloaded image is stored temporarily in the service. The next shows the URL API call.
- Glance - Image: POST, http://(IP):(port)/v2/images/(Image_ID)/file

Step 4 Private cloud authentication: The migration tool performs authentication to the private cloud.

- Parameters: Cloud endpoint, Tenant identifacation to the private cloud, Username and Password in the private cloud
- Description: The service prepares and performs the call for authentication to the private cloud which that the user specifies. The service displays the log in details of the user (Username, Tenant, Target Cloud). The next shows the URL API call.
- Keystone: POST, http://(IP):(port)/v2.0/tokens

Step 5 Image upload to private cloud: The migration tool performs the uploading process by firstly creating a new empty image and secondly installing the migration snapshot of the public cloud within it.
- Parameters: Name of the image, Format of the image (according to the format of the public cloud image), Accessibility level (public or private)
- Description: After the user fills the parameters and triggers the Upload Image process, the service creates an empty image with the specified name, updates the image by the specified format and accesses it in order to upload the data of the snapshot which is temporarily stored in the service VM. After the end of this process, the service will display the newly created images details. The next shows the URL API call.
- Glance - Image: Create Empty Image: POST, http://(IP):(port)/v2/images
- Glance - Image: Update Image: PATCH, http://(IP):(port)/v2/images/(Image_ID)
- Glance - Image: Upload Image Data: PUT, http://(IP):(port)/v2/images/(Image_ID)/file

Step 6 Keypair creation: The migration tool will create a new keypair or will input the keypair of the user (from the public cloud) into the private cloud.
- Parameters: he keypair based on the user selection or a new keypair that will be created on user demand.
- Description: This step is for configuring the keypair which will be used for the instance after its creation.The user could choose an existing keypair in a private cloud or could create a new on by providing a new name. Also, the user can import a keypair from public cloud (e.g. selecting from a list of available keypairs). Depending on the users choice, the service will select the appropriate keypair for the new instance and will displays the keypair name. he next shows the URL API call.
- Nova - Compute: Get Keypair: GET, http://(IP):(port)/v2/(Tenant_ID)/os-keypairs
- Nova - Compute: Import/Create Keypair: POST, http://(IP):(port)/v2/(Tenant_ID)/os-keypairs

Step 7 Launch of a new instance: The migration tool will create a new instance in the private cloud environment.
- Parameters: Name of the instance, Security Group based on public cloud setting.
- Description: The user sets the name and security group of the new instance and triggers the instance creation process. Then, it creates a new instance with the specified name, the image (as defined in Step 5), the keypair (as defiend in Step 6) and will automatically set the flavor to the same as the snapshot (as configured in the public cloud). The security group can be empty or default meaning that all the security rules of the snapshotted instance will be copied in the default security group afterwards. In case the user specifies a different name, a new security group will be created with the given name and similarly will setup the security rules of the snapshot instance of the public cloud. The next shows the URL API call.

- Quantum - Network: Create Security Group: POST, http://(IP):(port) /v2.0/security-groups
- Nova - Compute: Get Default Security Group: GET, http://(IP):(port) /v2/(Tenant_ID)/os-security-groups
- Quantum - Network: Create Security Group Rules: POST, http://(IP): (port)/v2.0/security-group-rules
- Nova - Compute: Create Instace: POST,http://(IP):(port)/v2/(Tenant _ID)/servers

Step 8 Set the new instance IP: The migration tool will allocate a new IP or select one existing to associate with the newly created instance.

- Parameters: Existing IP available (pre-allocated), New IP created by the migration tool.
- Description: The user can select between an existing IP from the list or create a new one that will be associated to the new instance. The IP list contains available IPs and after the choice of the user the service will display the IP associated to the instance. The next demostrate the API URLs.
- Nova - Compute: IP List: GET, http://(IP):(port)/v2/(Tenant_ID)/os-floating-ips
- Nova - Compute: Tenants Pool: GET, http://(IP):(port)/v2/(Tenant_ID) /os-floating-ip-pools
- Nova - Compute: Allocate IP To Tenant: POST, http://(IP):(port)/v2/ (Tenant_ID)/os-floating-ips
- Nova - Compute: Allocate IP To Instance: POST, http://(IP):(port)/v2/ (Tenant_ID)/servers/(Instance_ID)/action

Step 9 Overview of the migration actions and user selections.

- Parameters: This step does not include any parameters.
- Description: As a last step, the service will provide the full overview of the instance details as it is fetched by the public and private clouds system. It should be mentioned that the information are gathered from the interacted clouds and are not saved during the process to demonstrate whether the migration executed successfully. The next shows the URL API call.
- Nova - Compute: Instance Details: GET, http://(IP):(port)/v2/(Tenant _ID)/servers/(Instance_ID)
- Nova - Compute: Flavor Details: GET, http://(IP):(port)/v2/(Tenant_ID)/flavors/(Flavor_ID)
- Nova - Compute: Image Details: GET, http://(IP):(port)/v2/(Tenant_ID)/images/(Image_ID)

3.3.2 S2D: Instance Monitoring Tool: This component allows the cloud consumer to get the usage data of all instances in the public and private clouds. The user requires to provide information in order to be authenticated by the clouds system and then the service fetches the usage data automatically. In deatail. the usage date includes up-time of each instance, number of cores in use,

CPU hours, storage used and other useful metrics. In the S2D implementation the consumer and the provider of the software must both have access to the account that the VM is deployed after the migration. This gives the ability to control the VM so that the user can monitor, use and suspend the VM according to his/her needs. The performed call for getting the usage data is the following:

– Nova - Compute: Tenant Usage Data: GET, http://(IP):(port)/v2/(Tenant_ID)/os-simple-tenant-usage/(Tenant_ID)

3.3.3 S2D: XML API Call Implementation: Similar to the GUI, the xml API call performs calls to the instance migration tool within the service for VM migration. The difference is that the information used for migration is not gathered along the steps of Section 3.3.1 but is configured once in an XML document. This XML document must follow a specific syntax that is predefined in order to avoid information being missing. The validity of the document is verified against an XML schema and with successful validation the document is passed to the service processing.

4 Prototype and performance analysis

The S2D service implented its functionalities in the Intellicloud[10] (private cloud) environment of the Technical University of Crete and in the FIWARE cloud (public cloud). Both infrastructure are based on the Openstack platform. We aimed to develop a solution that will be user friendly, while at the same time offering an easy and simple way of migrating a VM and it does not require any expertise or knowledge other than the basics of an OpenStack environment. The back-end gets feed information by the front end or by direct communication with the user through the XML API. The performance evaluation of the prototype involves two experimental use cases that demonstrate the time required for an instance to execute the various steps of Section 3.3.1. Here the assumption is that the user performs a migration of a selected image from the public (FIWARE) to the private (Intellicloud) system.

Figure 2 demonstrates the variation and the differences among the various calls. For the first case (image of Ubuntu12.04LTS-64, 800MB) the total time for migratings the image is 64.93 seconds. In particualr, we can observe that the service executes most of the API calls related with configurations in reasonable time frames. For example, login to difrerent clouds requires less than 0.7 seconds, while time needed for generalized configurations (to create a keypair and set instance IP) is less that 8 seconds. For the second case the image size is over 1GB (with an image Centos 6.5 X86_64,1060.94MB) and the total time required for migratings the image is 79.98 seconds. Again here, we can observe that the service executes most of the API calls related with configurations are executed in reasonable time frames. Yet, the time needed for downloading and uploading

[10] http://cloud.intellicloud.tuc.gr

increases significantly the total time of migration. Finally, it should be mentioned that the most time is needed for donwloading and uploading the snapshot, an action that depends to the actual image size and on the bandwidth speed.The S2D prototype is available for experimentation from the Intelligence Systems Laboratory of the Technical University of Crete[11].

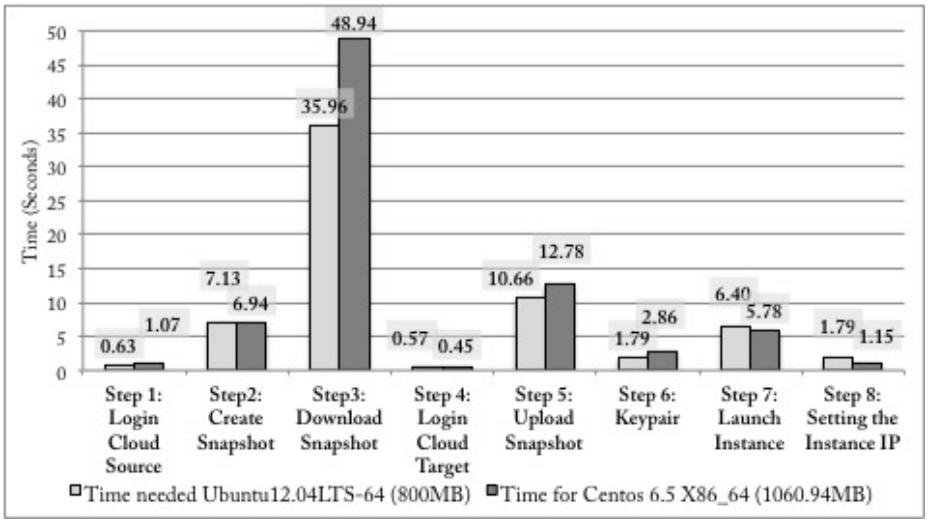

Fig. 2. Performance of S2D regarding migration of Ubuntu12.04LTS-64, 800MB and Centos 6.5 X86_64,1060.94MB images

5 Conclusions

This work proposes a solution to address these concerns or restrictions and builds-upon the idea of reverse-cloud approach. The solution revolves around the idea that application fields which cannot move to public clouds and take advantage of its offerings can still benefit by transferring the software close to the data source and deploy it in own private clouds in a hybrid cloud solution. Following this, we developed a service which could transfer and deploy a VM along with its software between two Openstack clouds. Our proposed S2D service implementation creates a backup or snapshot, which contains all the information needed to deploy an identical VM containing the same operating system and software. Then it transfers the snapshot and deploys it to its destination cloud. S2D provides important advantages such as interoperability between openstack clouds, migration of a running instance and re-instantiation, easy transfer of images (of openstack supported format), automatized configuration, easy deployment, setup through an xml file and easy to use GUI. The future work of our research includes exploration of migration of multiple VMs across heterogenous systems.

[11] http://www.intelligence.tuc.gr

In particular we aim to explore transfering of images and conversion in different image and container formats of different cloud providers.

Acknowledgement

The research leading to these results has received funding from the European Unions Seventh Framework Programme (FP7/2007-2013) under grant agreement no 604691 (project FI-STAR). This work is funded also by THALES project CYBERSENSORS, co-financed by the European European Social Fund (ESF) and the National Strategic Reference Framework (NSRF).

References

1. T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.
2. T. Fifield, D. Fleming, A. Gentle, L. Hochstein, J. Proulx, E. Toews, and J. Topjian. *OpenStack Operations Guide*. O'Reilly Media, Inc., 1st edition, 2014.
3. A. Galis and A. Gavras. *The Future Internet: Future Internet Assembly 2013 Validated Results and New Horizons*. Springer Publishing Company, Incorporated, 2013.
4. D. Petcu. Consuming resources and services from multiple clouds. *J. Grid Comput.*, 12(2):321–345, June 2014.
5. A. Semnanian, J. Pham, B. Englert, and X. Wu. Virtualization technology and its impact on computer hardware architecture. In *Information Technology: New Generations (ITNG), 2011 Eighth International Conference on*, pages 719–724, April 2011.
6. S. Sotiriadis and N. Bessis. An inter-cloud bridge system for heterogeneous cloud platforms. *Future Generation Computer Systems*, (0):–, 2015.
7. S. Sotiriadis, N. Bessis, A. Anjum, and R. Buyya. An inter-cloud meta-scheduling (icms) simulation framework: Architecture and evaluation. *IEEE Transactions on Services Computing*, DOI: 10.1109/TSC.2015.2399312, 2015.
8. S. Sotiriadis, E. Petrakis, S. Covaci, P. Zampognaro, E. Georga, and C. Thuemmler. An architecture for designing future internet (fi) applications in sensitive domains: Expressing the software to data paradigm by utilizing hybrid cloud technology. In *Bioinformatics and Bioengineering (BIBE), 2013 IEEE 13th International Conference on*, pages 1–6, Nov 2013.
9. C. Thuemmler, J. Mueller, S. Covaci, T. Magedanz, S. de Panfilis, T. Jell, A. Schneider, and A. Gavras. Applying the software-to-data paradigm in next generation e-health hybrid clouds. In *Information Technology: New Generations (ITNG), 2013 Tenth International Conference on*, pages 459–463, April 2013.
10. W. Vandenberghe, B. Vermeulen, P. Demeester, A. Willner, S. Papavassiliou, A. Gavras, M. Sioutis, A. Quereilhac, Y. Al-Hazmi, F. Lobillo, F. Schreiner, C. Velayos, A. Vico-Oton, G. Androulidakis, C. Papagianni, O. Ntofon, and M. Boniface. Architecture for the heterogeneous federation of future internet experimentation facilities. In *Future Network and Mobile Summit (FutureNetworkSummit), 2013*, pages 1–11, July 2013.