MarUnivCloud: Towards a Moroccan inter-University Cloud

Abstract—Cloud Computing is emerging as a very promising technology changing the way we approach Computing.

MarUnivCloud is a project aiming at the promotion of Cloud Computing via the deployment of a real-world private Cloud interconnecting multiple Moroccan Universities. The deployed Cloud will benefit both students and researchers, and can serve as a seed for a futuristic public Cloud.

In this paper, we present the goals, and the mission, of MarUnivCloud. We delineate relevant technical aspects (e.g., architecture and services), and present real-world tested deployments, at two university sites, using open-source software, e.g., OpenStack and OpenNebula.

I. INTRODUCTION

Cloud Computing is emerging as a very promising technology promoting the usage of IT as a Utility instead of a Product. Cloud computing provides IT services on an on-demand and pay-per-use basis. There are plenty of services offered by the Cloud. NIST [1] defines three services as fundamental though: SaaS (Software as a Service), PaaS (Platform as a Service), and IaaS (Infrastructure as a Service).

Cloud computing attracts businesses CEOs, and actors, as it provides a strong business agility model: the initial investment cost for installing an IT infrastructure for businesses can be easily circumvented by opting for the pay-per-use Cloud services, i.e., one can start his business with a budget of 0dhhs for the IT infrastructure. Most important is that business actors can get rid of the significant burden involved around owning an IT infrastructure, e.g., recruiting qualified engineers and technicians, software purchase, maintenance, and providing the logistics for hosting a data center.

Indeed, Cloud computing presents plenty of opportunities for IT users. In this paper we present the initiative of settling a private Cloud connecting Moroccan universities, and providing IaaS services at a first stage, while targeting PaaS and HPCaaS (High-Performance Computing) at a later stage. All Moroccan universities are welcome, and solicited, to join the project.

Like similar predecessor projects connecting Moroccan Universities, e.g., Marwan (Moroccan Academic and Research Wide Area Network) [2] and MaGrid [3], MarUnivCloud is targeting the connection of multiple universities’ sites to the same private Cloud, using the MarWan facility. The latter is a dedicated network, connecting Moroccan Universities, and providing high-bandwidth links.

At the actual stage of the project, two operational Cloud testbeds have been implemented at two separate sites: one at Cadi Ayyad University [4], and the other at Alakhawayn University [5].

Our mission is the promotion of Cloud Computing as a novel and promising technology which is changing the way, we humans, deal with IT forever. It is an IT paradigm shift: Simply put, Cloud computing is taking IT from being a mere Product towards being an indispensable “Utility” like any other utility (e.g., electricity and water) whereby the two main characteristics are: availability and metering. Availability brings “utilities” to our homes, like what happened with electricity and water. Metering counts the volume of the consumed utility, i.e., pay-per-use.

In this context, and to parallel with this IT paradigm shift, we are taking the initiative of boosting the interest and sharing the expertise, in
Cloud computing, amidst Moroccan universities, thus pushing towards building a Moroccan Private Cloud that can compete, in the future, with other Cloud Businesses. This can only be reached by paving the way towards establishing a Seed Private Cloud: MarUnivCloud.

Once established, researchers can investigate and tackle the Cloud computing rudiments and performance-related issues, and promote the seed private Cloud towards a Public one capable of providing Cloud services to Moroccan companies and individuals. A sample service is the providing of CRM (Customer Relationship Management) services to Moroccan companies. This is an indispensable Cloud service for all futuristic standard companies. By providing Cloud services, we would overcome the main drawback of Cloud computing residing in “Data Privacy”. Indeed, by establishing a Moroccan Cloud, “we” keep our data within our premises and avoid sending it abroad.

In this paper, we present the technical details for building a seed private Cloud by delineating the subtleties of two real-world deployments lead at two different Moroccan universities. Both deployments are using open source software. The deployed testbeds provide IaaS, and precisely, Linux VMs (Virtual Machines) for students and researchers. Once deployed and migrated to other sites, and interconnected, this will constitute the seed Cloud upon which researchers can lead their different cloud-related studies and tweak its performance to reach the standards of public Clouds providing services to the general public.

The rest of the paper is organized as follows. In Section 2, we reiterate over the project main goals. Section 3 delineates the overall proposed Cloud architecture. In Sections 4 and 5, we present the deployment subtleties of IaaS at Site 1 and 2. Finally, we conclude and present future work is Section 6.

II. MARUNIVCLOUD: A COMMUNITY CLOUD FOR MOROCCAN UNIVERSITIES

In this project, we mainly target the promotion of knowledge in the field of Cloud computing among the different Moroccan universities, by setting up a real-world Cloud platform. The project aspires to meet its target via the promotion of (1) Research, and (2) University Education Quality.

A. Research Promotion

As a novel IT field, Cloud computing exhibits plenty of hot research topics, e.g., High-Performance Computing in the Cloud, Mobile Cloud Computing, Big Data Mining, Software-Defined Networking for the Cloud, the Impact of Virtualization on Cloud performance, Security in the Cloud, Performance Evaluation of Cloud Computing Platforms, etc. These hot topics can be effectively addressed only if we dispose of a real-world Cloud computing platform.

At this stage of the project, we are focusing on the deployment and performance evaluation of Cloud computing platforms. This ought to be the most convenient research venue, heretofore, as we are still investigating the settlement of the Cloud infrastructure. We are mainly targeting the studying of relevant load-balancing schemes especially that MarUnivCloud will connect different sites, and looks like a “miniaturist” Cloud of Clouds where the instantiated VMs need to be balanced among the different Cloud sites. Besides, we will investigate different open-source platforms (e.g., Openstack [6], OpenNebula[7], and Eucalyptus[8]), and decide on the most suitable one for our deployment and services to offer. This would involve researching relevant implementation internals, and conducting comparative real-world experimentation. Relevant real-world performance evaluation studies will be lead and matched against theory.

As of Cloud services, we are focusing on providing Infrastructure as a Service (IaaS) at a first stage. IaaS is the building Cloud service for all other cloud services. PaaS and SaaS are built on top of IaaS. The provided IaaS is meant for university students and researchers.

B. University Education Quality Promotion

Any educational institution should ideally be able to:

- provide students, staff, and faculty with commodity software (e.g., email accounts, operating systems, etc.) and hardware (e.g., PCs, Servers, etc.);
- provide researchers and postgraduate students with the required special software and hardware to run experiments that are likely to involve a great deal of processing and computation;

With educational institutions being more and more under budget shortage and wanting to better reallocate their limited internal resources to better support their corporate priorities, cloud computing is a promising field. Students, administrative staff and faculty can be made to use the services of SaaS and IaaS clouds, developers can use all the software they need for their development online and all the hardware for hosting their applications through a PaaS cloud, while researchers whose projects require a great deal of processing power and/or additional
server capacity can do so in minutes and at the click of a button through an IaaS cloud [17].

The Cloud installed at Cadi Ayyad University (member of the MarUnivCloud Consortium) allows provisioning IT infrastructure and reducing maintenance costs [4]. Indeed, instead of providing physical machines to students/staff, the cloud provides Virtual Machines (VMs). As an example, VMs use is particularly suitable for networking exercises like routing labs or firewall configuration where students usually need multiple physical machines to complete the exercise. While on shared servers, root access would never be granted, with our cloud students can have root access on their own image, and any damage they do can be isolated and rolled-back by simply re-instantiating the image. Before using the cloud, PCs were assigned among students at the beginning of each semester, and at the end of the semester PCs need to be collected and maintained (i.e. components functionality check, wiping of installed software, new OS installation, etc.). This process consumes a considerable amount of time of IT resources management department depending on how many PCs were used. Using the cloud considerably optimizes this process since only image re-instantiation is required which is a matter of minutes instead of hours. Moreover, before deploying the cloud, faculty was most of the time complaining about errors due to OS misconfiguration, missing software components or missing programming libraries. Now with predefined VMs use, all attending students use the same development environment, perfectly known to the lecturer.

III. ARCHITECTURE

Each (participating) university will be devoting a multicomputer cluster site, e.g., consisting of multiple commodity PCs along with a powerful server serving as controller node, see Fig. 1. At a later stage, the hardware can be updated to provide more powerful computing power. At this stage, we are mainly targeting the proof of concept and making the Cloud operational.

Like all Cloud computing platforms, MarUnivCloud hosts a web server, and exhibits a GUI interface accessed via legacy browsers. Once logged into the MarUnivCloud web server, the user will be given the floor to choose the appropriate VM (IaaS) to instantiate. The user will select relevant specifications for the VM to create, e.g., number of vCPUs, memory space, and the Linux operating system flavor.

The web server serves as a proxy that forwards user requests to the appropriate Cloud cluster while accounting and keeping logs of the created VMs, users, and locations, in order to serve as input to the load-balancing module. Ideally, the selection of the appropriate cluster to host the newly instantiated VM would mainly depend on 2 basic criteria: 1. Load-balancing, i.e., the server should keep track of the load levels at every site in terms of memory and CPU capacities, and 2. the user authentication/group. Regarding, the last criterion, a user request should, by default, be “directed” to the cluster of the university where the user is enrolled in order to enforce the “locality principle” unless the load-balancing constraints states differently, e.g., the local site is overloaded.

In the first phase of the project, each university will start by deploying a local independent cluster. Fig. 2 depicts the relevant architecture.

The Local sites will consist of a minimum of three servers. The user will interact with the Cluster via a web server; the latter is collocated with the Openstack Controller Node. The two other servers are Openstack Compute Nodes, which basically host the created VMs. The Local Site Components and relevant processes flow are tackled in more detail in Section IV.

After the deployment of local sites, and the carrying out of appropriate testing, the next phase is to interconnect the sites to constitute the seed for MarUnivCloud.
This step is the most challenging one as it involves migrating the Nova Controller node to a single site, and use another server as a replica. To this end, we will start by connecting two-sites at each stage, test the interconnection, and migrate towards a full interconnection of the sites. When connecting 2 sites, only one controller node will persist; the other controller node (from the second site) will be configured to operate as a mere compute node, see Fig. 3. The 2 sites will be connected via MarWan [2].

The deployed Cloud seed has been settled at AUI [5] using the Openstack Cloud Computing platform [6]. The hardware infrastructure consisted of the following:

- 1 Dell PowerEdgeT320 server: serving as a Controller Node
- 2 Dell PowerEdge2950 servers: serving as Compute Nodes
- 1 switch: 1Gbps

The 3 servers are connected to a switch, and form an internal network. Still, each server has another network interface card (NIC) connecting it to the external network (i.e., Internet).

A. Controller Node Deployment

The PowerEdgeT320 is the controller node. Its role is to provide all the functionalities of Openstack except the ones concerned with hosting virtual machines and network services [9]. It actually runs many of the Openstack daemons such as Keystone, Glance, and Nova. Figure 4 depicts relevant components.

The Controller node is made basically of the following components:

1. Nova-api: This process receives API requests, routes, and authenticates them to the concerned Openstack processes.
2. Message Queue: A central hub through which messages pass between Nova components.
3. Database: It stores the Cloud configuration, flavors, instances in use, network available and projects
4. Nova-Scheduler: This daemon takes a VM request from the message queue and applies scheduling techniques in order to decide on the host to run the new VM instance.
5. Keystone: It is the identity service that deals with authentication and authorization.
6. Glance: It deals with the discovering, registering, and retrieving of virtual machines images.
7. **Cinder:** It provides Openstack users an infrastructure for managing volumes, previously known as nova-volume.

Further details about the interaction between these components are highlighted in Section IV.D, when referring to the overall Processes Flow.

After installing the Compute node, the next step is to set the Networking configuration.

We configure the external and internal networks to be used by the Openstack nodes, e.g., the compute node. The internal network is the one connecting the nodes and is visible only internally using fixed IP addresses, while the external network uses floating IP addresses. Afterwards, we install the Openstack database in the controller. This keeps track of all accountings, e.g., state of running instances, networks available, and VM images information. We used MySQL.

The different Openstack components communicate with each other using the messaging-passing technique. In our case, we opted for RabbitMQ [10]. The latter is a well-known message broker that allows coordination between the processes in distributed systems.

At this point, we are ready to install the Openstack packages: The first package to install is the Identity service (Keystone). As its name suggests, this acts as a backbone for all of the other Openstack services. It provides a catalog of available services along with their API endpoints [11]. The configuration of Keystone consists basically on populating the (keystone) database with users, tenants, roles, and the corresponding services and their API endpoints.

The next service to install is the Image Service (Glance). This allows users to create virtual machines images. Its installation and configuration follows the same pattern as Keystone. It has its own default database that holds entries about VM images, e.g., flavors, and virtual machine images’ URIs.

**B. Compute Node Deployment**

Unlike the controller node, which is unique, there are multiple nodes serving as compute nodes. In our case, we configured two servers as compute nodes. There are only two packages to install: (1) Nova and (2) Cinder.

On the compute nodes, the Nova processes to install are the Nova-compute and Nova-network. The Nova-compute process deals with the spawning and deletion of VM instances. The Nova-network deals with the configuration of the VM network and the assignment of IP addresses.

The last service to install, in the compute node, is the Block Storage Service (Cinder). This manages the creation, attachment and detachment of persistent block storage to virtual machines instances.

In future releases, nova-network is going to be deprecated and its functionalities completely moved to another Openstack project called Neutron [12]. However, for the purpose of our project, nova-network meets the necessary functionalities: we used the nova-network Flat DHCP mode to manage the distribution of fixed/flat IP addresses among the virtual machines.

To test our platform, we used Cirros cloud images [13]. Cirros is lightweight OS that has been developed specifically to run on Cloud infrastructures. It provides log files from which we were able to track bugs and debug them. These consisted mainly of Network problems.

**C. Processes Flow**

In this section we delineate the sequence of events involved in the creation of a new virtual machine instance. Figure 4 depicts the relevant processes flow.

![Provisioning Processes Flow](image)

Figure 5. Provisioning Processes Flow

The first step, in the flow, consists on the user sending an http request to the controller node, asking for a new virtual machine (1). The http request is handled by the web server. The latter translates the http request into an instance creation request embedded in a REST (Representational State Transfer) API request, and sends it to Nova-api.

Upon request reception, the Nova-api daemon sends it for validation and asks for access permission
through Keystone (2). When validated, the Nova-api creates a new database entry for the new instance (4). It also sends a RPC (Remote Procedure Call) request to the Nova-scheduler which picks the request from the RabbitMQ queue, and questions the database to locate the appropriate host (5).

In step (6), the Nova-scheduler sends an RPC request to Nova-compute on the appropriate host. The Nova-compute on the host sends a request to the nova-conductor daemon (on the controller node) to fetch the corresponding image flavor, and send it back to the Nova-compute (7). Nova-compute uses a REST call to Glance to get the image URI and the image metadata, and then uploads it from the image storage (8). The Nova-compute sends afterwards a request to Nova-network to configure the network so that the VM instance is allocated an IP address (10). The next step is Nova-compute making a REST call to Cinder-api in order to attach volumes to the instance (11). Note that each time an OpenStack service (e.g., Cinder, Glance, and Nova) receives a REST call, an authentication request is sent to Keystone (9, 12).

Finally, Nova-compute generates the data necessary for the hypervisor, which is KVM [15] in our case, to spawn a new VM instance via the libvirt [16] interface on the chosen compute node (13).

To ease access to VMs, a GUI is provided. Instead of CirROS, which was used solely for testing, Ubuntu 13.10 Desktop images were uploaded as the virtual image to be used by the students. However, since the default desktop environment, Unity, is resource-heavy, we modified the Ubuntu image to provide LXDE (Lightweight X11 Desktop Environment) desktop environment instead [14].

V. IAAS DEPLOYMENT AT SITE 2

The deployed Cloud seed has been settled at CAU [4] using the OpenNebula Cloud Computing Platform [7]. The hardware infrastructure consisted of the following:
- 1 IBM System x3650 M3 server: serving as a Front End Node
- 2 IBM System x3650 M3 servers: serving as Cluster Nodes
- 1 switch: 1Gbps

To construct a private cloud using OpenNebula, the physical infrastructure should adopt a classical cluster-like architecture. We need a front-end machine that holds all the OpenNebula installation and execute OpenNebula services, a set of hypervisor-enabled hosts where Virtual Machines (VM) are executed, Datastores to hold the base images of the VMs, and at least one physical network joining all the hosts with the front-end. In our setup, Datastores are accessible through the front-end using a Network Attached Storage system (NAS) that allows us to store up to 18TB on redundant disk storage.

Fig. 6 depicts a simplified view of the UCA cloud physical infrastructure. Only the frontend machine is reachable via the public network to allow combining local resources with resources from Amazon EC2 or those from a remote Cloud partner infrastructure (such as Johannes Kepler University Cloud) also running an OpenNebula instance. All other components are connected via a high bandwidth private network.

![Figure 6: CAU Cloud Architecture](image)

To set up UCA private cloud, we followed the following steps:
- The first step was to configure the system datastore to hold images for the running VMs. We used shared storage to be able to clone, save, and migrate VMs. OpenNebula can also work without a Shared FS, but this would force the deployment to always clone the images and we would only be able to perform cold migrations.
- The second step was to configure a bridge in the physical hosts to allow OpenNebula to connect it to the virtual machine network in order to offer network connectivity to the VMs.
- The third step was to Install and configure OpenNebula on the FrontEnd along with the Information Manager (IM), Transfer Manager (TM) and Virtual Machines Manager (VMM) drivers. Information drivers are used to gather information from the cluster nodes, transfer
drivers are used to transfer, clone, remove and create VM images, and virtualization drivers are used create, control and monitor VMs.

- The last step was to install and configure hypervisors on the cluster nodes. Many hypervisors are supported (e.g. KVM, Vmware, and VirtualBox) and different hypervisors can be handled by the same FrontEnd Instance.

The access to the private cloud is either through Command Line Interface (CLI) or the OpenNebula Sunstone Graphical User Interface. In both access modes, we are able to perform the following typical management operations:

- manage users: create users, list users, login as user, change password and delete users.
- manage cluster nodes: add nodes while specifying the storage type (shared or non shared) and the hypervisor to use (kvm, xen or vmware).
- retrieving information about a cluster node: memory used by VMs, CPU, etc.
- manage images, edit images templates, register images and list available images.
- retrieving information about a specific image: its source, type, number of virtual machines using it, etc.
- manage networks, specify network templates, create networks, and specify the pool of IP addresses VMs should use.
- manage virtual machines, create VMs using images, list VMs, and display the characteristics of a specific VM.

The UCA private cloud is connected to Johannes Kepler University (JKU) cloud in Austria under the Cloud Computing for Cooperation (C3Lab) Initiative [18]. JKU Cloud is running the same OpenNebula instance and has the same physical infrastructure characteristics. The aim of the installed Cloud is not to expose to the world a cloud interface to sell capacity over the Internet, but to provide local cloud users and administrators with a flexible and agile private infrastructure to run virtualized service workloads within the administrative domain.

VI. CONCLUSION AND FUTURE WORK

We presented the MarUnivCloud project along with its mission and goals. We delineated relevant implementation aspects, and presented two real-world deployments at two different universities.

We strongly believe that MarUnivCloud will definitely boost expertise in the field of Cloud Computing in Morocco.

As a near future work, we are planning to migrate the single deployments to span other Moroccan universities, and start on the interconnection phase. Once interconnected, MarUnivCloud will serve as an ideal platform for Moroccan students and researchers in the field of Cloud computing. This will definitely pave the way towards a Moroccan public Cloud capable of providing and meeting the futuristic needs, on Cloud services, for the general Moroccan public.

REFERENCES

[9] OpenStack Project, “Chapter1. Architecture”. Online: http://docs.openstack.org/grizzly/basic-install/apt/content/basic-install_architecture.html