Cloud Infrastructure Using OpenNebula

Faryal Tarique
Department of Computer Science
University of Porto
Porto, Portugal

Mariana
Department of Computer Science
University of Porto
Porto, Portugal

Tomás Carmo
Department of Computer Science
University of Porto
Porto, Portugal

***Abstract*—This project establishes a customized cloud infrastructure using OpenNebula, tailored for parallel and distributed computing. It incorporates secure user registration, dynamic resource allocation, disk storage, container, and database services employing a Command Line Interface (CLI). Key technologies used include OpenNebula, MySQL, Docker, and Google Cloud Platform. The project provided valuable insights into the capabilities and challenges of cloud technologies.**

I. Introduction

Cloud computing offers scalable and on-demand access to computing resources, reshaping organizational IT infrastructure significantly. This project establishes a customized cloud infrastructure using OpenNebula, specifically designed for parallel and distributed computing. Key features include secure user registration, account management, dynamically adjustable elastic computational resources, disk storage, and vital container and database services. Our infrastructure operates under a User Level Agreement that guarantees efficient job processing and resource optimization. Distinctively, we have developed our own CLI (Command Line Interface), where users select options and interact directly with backend scripts. This setup allows for precise control and management of resources, aligning with the real-time demands of a production environment and bypassing traditional graphical dashboards.

II. Background on Chosen Software and Libraries

In this cloud infrastructure project, we chose OpenNebula and MySQL as our primary software solutions due to their robustness, flexibility, and compatibility with our requirements. OpenNebula was selected over other cloud management platforms like Eucalyptus and CloudStack because of its powerful command-line interface (CLI), extensive community support, and comprehensive documentation. This choice allows for precise, script-based control of cloud resources, crucial for deploying and managing private and hybrid cloud environments efficiently. The CLI approach aligns with our need for a non-GUI based operation, optimizing both system performance and resource allocation in real-time scenarios.

MySQL was adopted as the database management system for its proven reliability, performance, and widespread adoption in production environments. Compared to alternatives such as PostgreSQL or MongoDB, MySQL offers the necessary features for effective user management and session data handling with a simpler configuration and maintenance profile within our infrastructure. Collectively, these technologies ensure that our cloud system is not only operational and scalable but also robust and manageable, supporting dynamic resource allocation and high reliability.

III. Materials and Methods

A. Machines Used and Their Characteristics

The initial development phase utilized an Ubuntu 20.04 virtual machine (VM) hosted on Oracle VirtualBox. This VM was allocated 4 GB of RAM and dynamic storage starting at 20 GB, which could be extended as needed for the OpenNebula cloud environment. This setup was primarily used for development and testing purposes, focusing on account management and script testing. However, due to limitations with nested virtualization in VirtualBox, we transitioned to a more robust platform for full-scale deployment.

Migration to Google Cloud: Confronted with the nested virtualization constraints of VirtualBox, the project was migrated to Google Cloud VMs. This migration was critical for enabling effective VM management and deployment capabilities, utilizing Google Cloud's support for nested virtualization to meet our project's scalability and performance needs.

B. Software Used and Their Versions

Our project utilized a stack of integrated software tools selected for their stability, performance, and community support:

- **Operating System:** Ubuntu 20.04, chosen for its stability and broad software compatibility.
- **VirtualBox:** Employed for creating and managing the initial development VM.
- **OpenNebula:** This cloud management platform was used for resource management, primarily through its CLI to automate and script cloud operations.
- **MySQL:** Selected for its robust performance in handling complex database management tasks required for user and session data.
- **Docker:** Installed within Ubuntu for containerization, providing consistent and scalable application deployment environments.

IV. Installation And Implementation Details

A. Account Management

Our project includes a secure and efficient user registration and login system. This system categorizes users into two roles: Admins and Regular Users, employing role-based access

```
The arguments should be: [group_name]
root@rootft:/home/rootft/Desktop# onegroup create "Admin"
ID: 100
root@rootft:/home/rootft/Desktop# onegroup create "Regular User"
ID: 101
root@rootft:/home/rootft/Desktop#
```

Fig. 1: Two new groups for admin and regular user created in Open Nebula

```
root@rootft:/home/rootft/Desktop# sudo ./cloud_management.sh
1) Register
2) Login
3) Quit
Please enter your choice: 1
Enter username: Faryal
Enter password:
Is this an admin user? (yes/no): yes
User creation output: ID: 5
User created successfully with ID: 5
Admin privileges granted to Faryal.
Please enter your choice:
```

Fig. 2: User Registration

controls to manage cloud resources effectively. Here's a step-by-step guide on how our account management features are implemented:

User Registration: Users register through a command-line interface (CLI), providing a username and password. Our backend script facilitates this process with OpenNebula, automatically setting appropriate resource quotas and configurations.

Secure Login: Authentication is required for users to access cloud resources. We implement advanced session management techniques to enhance security and ensure proper session termination after inactivity or logout.

Account Management: Users can update their account settings, including password changes. Admin users have broader capabilities, such as managing user roles, adjusting resource quotas, and monitoring user activities across the cloud environment.

B. VM Management

The management of virtual machines (VMs) is a critical component of our cloud services. Admins have enhanced capabilities, including setting quotas and overseeing all active VMs. We prevent resource monopolization by enforcing quotas for regular users, such as a maximum of two VMs, 4 GB of memory, and two CPU cores.

```
root@rootft:/home/rootft/Desktop# sudo ./cloud_management.sh
1) Register
2) Login
3) Quit
Please enter your choice: 2
Enter username: Faryal
Enter password:
GROUP is: Admin
Admin login successful!
1) Account Management      3) Disk Storage
2) Virtual Machines (VMs)  4) Logout
Please enter your choice:
```

Fig. 3: User Login Result

```
Login successful!
1) Account Management      3) Disk Storage
2) Virtual Machines (VMs)  4) Logout
Please enter your choice: 1
1) Update Password
2) Delete My Account
3) Quit
Select an option: 1
Enter your old password: Hello
Enter your new password:
Password updated successfully.
```

Fig. 4: Update Password Option

```
1) Update Password
2) Delete My Account
3) Quit
Select an option: 2
Are you sure you want to delete your account? (yes/no): yes
Account deleted successfully.
Please enter your choice:
```

Fig. 5: Delete User Account

The VM management script is an interactive Bash script designed to facilitate the administration of virtual machines (VMs) within an OpenNebula environment via a command-line interface. It initiates by presenting a menu-driven interface that offers various options for VM management, including creating, deleting, monitoring, scaling VMs, viewing all VMs, and setting global quotas. Key functions include `create_vm`, which checks user quotas before creating a VM; `delete_vm`, which removes a specified VM; `monitor_vm`, for displaying VM status; and `scale_vm`, which adjusts VM resource allocations based on user input. The script incorporates role-based access control, restricting certain functions, such as viewing all VMs and setting global quotas, to admin users only. This is achieved by verifying if the `user_group_id` equals 100, identifying the user as an admin. The script ensures robust error handling and user feedback, making it an essential tool for administrators requiring direct, script-based management of cloud resources. Each operation within the script provides immediate feedback regarding the success or failure of the action, enhancing usability and facilitating effective troubleshooting in a production environment.

VM Creation and Management in GCP: After moving our project to Google Cloud due to nested virtualization limitations with Oracle VirtualBox, we continued with VM creation and management scripts there. Our setup in Google Cloud allowed for enhanced performance and the successful deployment of VMs through scripted automation.

Note: The main menu of our script offers options like register and login initially. After login, users are presented with

```
Please enter your choice: 2
1) Create VM           3) Monitor VM          5) See All VMs          7) Quit
2) Delete VM           4) Scale VM            6) Set Global VM Quota
Select an option: 1
Enter VM template ID: 0
```

Fig. 6: The template for the VM instance is set

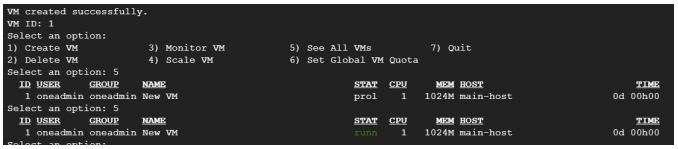

Fig. 7: VM creation

another menu which includes options for account management, virtual machines (VMs), and disk storage. Each category invokes a separate Bash script that handles the corresponding tasks, tailored to the user's permissions and role.

C. Database Service

In the digital era, efficient and scalable database management is crucial for businesses and organizations to streamline operations and manage data effectively. This report focuses on the creation of a database service using OpenNebula within a virtualized environment. The service aims to provide users with the ability to manage databases through a user-friendly script interface.

1) Setup Environment: To host the database service, we created a OpenNebula VM template with mostly default settings, meaning a simple virtual bridge network (initialized previously with *onevnet*), an Ubuntu 20.04 image (initialized previously with *oneimage*), 1 CPU and 1024 MB of memory, but with a startup script that installs the necessary software for the user to make use of this service, in this case, MySQL.

2) Database Configuration: Upon setting up the virtual environment, MySQL was configured to handle database operations. The configuration steps included:

- **Database Creation and Management**: Creation of databases using SQL commands through the MySQL command-line interface managed by OpenNebula and actions such as table creation or querying.
- **User Accounts**: Establishment of user accounts with appropriate permissions to manage databases securely.

3) Script Development: To enhance usability and manageability of the database service, a Bash script was developed. This script provides a menu-driven interface for users to interact with the MySQL databases. The script features include:

- **Menu Options**:
  - **Create Database**: Allows users to create new databases dynamically.
  - **List Databases**: Provides a list of existing databases within the MySQL instance.
  - **Create Table**: Enables users to define database tables with specified fields and constraints.
  - **Insert Data**: Facilitates insertion of data records into predefined database tables.
  - **Query Data**: Executes SQL queries to retrieve and display data from database tables.
  - **Exit**: Allows users to gracefully exit the script.

  The script leverages MySQL command-line tools to execute SQL commands directly, ensuring seamless database management operations without the need for extensive knowledge of SQL or database administration.

This way, the implementation of a database service using OpenNebula within a virtualized environment demonstrates the versatility and efficiency of cloud-based solutions in managing databases. The integration of a menu-driven script enhances user accessibility and simplifies database administration tasks. Future enhancements could include automation of backup processes, integration with monitoring tools, and further optimization of resource allocation within the virtualized environment.

D. Container Service

Container services provide a lightweight and efficient way to deploy, manage, and scale applications. In our cloud infrastructure, we also included container services using Docker, a well known platform for containerization. Docker allows developers to package applications and their dependencies into a standardized unit called a container, which can run consistently across various environments. To create a Docker container, you will need your application folder and a Dockerfile, a docker configuration file that specifies everything needed to set up the environment, install necessary software, copy application files, and define the command to run the application. We also took advantage of Docker Hub is a cloud-based registry service provided by Docker that allows you to find, store, and share Docker images to allow users to select Docker images from Docker Hub and run them.

The container service setup involved creating a OpenNebula VM that would host the container service, similar to the database service, using simple VM characteristics, only available to the admin.

By leveraging Docker, we created an interface hosted on the VM, where a user is allowed to:

- **Run a container:** Either by **running a Docker Hub image**, providing the container image name, the port mapping between the host and the application and the local given container name, or by **deploying their own application**, which had to include a Dockerfile. This way, the user had to provide the application path, a container image name, the port mapping and a local container name, similar to before.
- **Manage a container:** To manage specific containers, we listed all containers to the user, prompted with a container ID, and displayed options such as **start, stop, restart or delete** container, **inspect** container details and **access container shell**.

This way, the user has a handful of container operations available to him, without having the need to host and allocating them in their personal computer.

Present the data and results from your tests. Use graphs, tables, and charts to make your data easier to understand.

V. Discussion

The cloud infrastructure project using OpenNebula demonstrated the potential for building scalable and flexible cloud

environments tailored to specific needs. Key aspects like user management, VM creation, and database services were implemented through CLI-based interactions, providing precise control over resources. However, several challenges were encountered during development. As a result of these obstacles, the system's performance ultimately did not met project goals.

A. *Difficulties*

1) VirtualBox Limitations: Initially, VirtualBox was chosen for VM creation, but it lacked support for nested virtualization, crucial for running complex cloud environments. This limitation necessitated a switch to a more robust platform capable of handling such requirements.

2) Google Cloud Network Issues: After migrating to Google Cloud, the project faced network errors that affected VM performance. These issues required extensive troubleshooting to ensure stable network configurations, demonstrating the complexities involved in cloud deployments.

VI. Conclusions

The project on cloud infrastructure using OpenNebula was a valuable learning experience, illustrating both the capabilities and challenges of cloud technologies. The implementation of CLI tools and the management of cloud resources underscored the importance of flexibility and control in cloud environments. Although initial setbacks with VirtualBox and network errors in Google Cloud presented significant challenges, the project could have been completed with more time and orientation.