**Tazhibayev Sultanbay**

**SE-2326**

**Assignment 5**

**Advanced Operating Systems | Seilkhanova Kymbat**

**Objective**

Design and implement a basic hypervisor system with support for virtual machines (VMs). The project will focus on the following core tasks:

* Enable the creation and management of virtual machines within the hypervisor environment.
* Implement functionality to start and stop applications running inside each VM, ensuring proper control over their execution.
* Monitor and calculate the percentage of system resource usage (such as CPU and memory) by each VM to assess performance and efficiency.
* Investigate and identify the type of hypervisor technology utilized by Google Colab, providing insights into its virtualization layer.

This project aims to build foundational knowledge in virtualization technologies and offer hands-on experience with hypervisor-level resource management.

**Procedure**

**Step 1: Launch the Ubuntu Virtual Machine**



To begin the virtualization process, the UTM application was opened, which serves as the virtual machine manager. Within UTM, an existing Ubuntu virtual machine was selected and successfully launched. This VM provides the Linux-based environment required for subsequent hypervisor testing and application management. The system booted into the Ubuntu OS, ready for further configuration and experimentation.





**Step 2,3: Virtual Environment Setup**

After launching the Ubuntu virtual machine, the next step was to set up a dedicated Python environment to ensure isolation and dependency management for the hypervisor project. The following actions were performed:



* Installed Python using the system’s package manager to ensure the latest compatible version was available.
* Created a new virtual environment using `python3 -m venv <env\_name>`, which allows for clean package management without affecting the global Python installation.
* Activated the virtual environment with `source <env\_name>/bin/activate`, ensuring all subsequent Python packages and modules would be installed within this isolated workspace.



This environment serves as a controlled space for developing and testing the hypervisor-related scripts and tools.

**Step 4: Script Execution and Verification**

With the virtual environment configured and the necessary scripts prepared, the next step was to execute the main Python script responsible for managing the hypervisor: `lab5.py`. This script was run to verify the correct implementation of several key functionalities:

Virtual Machine Addition: Confirmed that new virtual machines could be successfully added to the hypervisor through the script's interface.

Application Lifecycle Management: Tested the ability to start and stop applications within the virtual machines, ensuring proper interaction with the underlying VM infrastructure.

Output Validation: Carefully reviewed console output and any log files to confirm that the operations completed without errors and behaved as expected.



This step validated the core capabilities of the system and demonstrated successful integration of VM management operations via scripted automation.

**Step 5: Resource Usage Calculation**

The next phase involved assessing how efficiently system resources were being utilized by the applications running within the virtual machines. The following resource metrics were calculated for several applications:

* CPU Usage: Measured the percentage of processor time consumed by each application using system monitoring tools such as `top` or `psutil`.
* Memory Usage: Evaluated the amount of RAM utilized by each process, identifying memory-intensive applications and potential bottlenecks.
* Storage Usage: Determined disk space consumption by analyzing the size of application files, temporary data, and logs using tools like `du` or `df`.



These metrics provided valuable insights into the performance and scalability of the virtualized environment, helping to ensure that the hypervisor and VMs operate efficiently under varying workloads.

**Answer to the Question**

Google Colab operates on a Type 1 hypervisor, specifically KVM (Kernel-based Virtual Machine), which is integrated directly into the Linux kernel. As a native hypervisor, KVM offers efficient virtualization with minimal overhead, allowing for high performance and effective resource utilization in cloud-based environments like Colab.

**Conclusions**

* Successfully designed, implemented, and tested basic hypervisor functionality, including the creation and management of virtual machines.
* Verified the ability to start and stop applications within the VMs, demonstrating control over virtualized environments.
* Accurately calculated CPU, memory, and storage usage statistics for running applications, providing performance insights.
* Determined that \*\*Google Colab\*\* utilizes \*\*KVM\*\*, a \*\*Type 1 hypervisor\*\*, for its virtualization infrastructure, confirming the use of efficient and kernel-level virtualization technology.

These results demonstrate a solid understanding of virtualization concepts and practical skills in managing and analyzing virtual machine behavior.