Performance Analysis and Online Resource Management of a VM System

GRS Project - Final Report

Priyanshu Kumar Rai	Aman Yadav	Ritesh Gupta	Vanshika Jain

Abstract

This project investigates the deployment, performance analysis, and online resource management of virtual machines (VMs) using QEMU [1]/KVM [2] and libvirt [3] on a Linux host. Multiple Linux VMs are configured as distinct nodes, with the host system acting as a centralized controller for monitoring and analyzing resource utilization (CPU, memory, networking, and disk I/O). Each VM is assigned varying CPU and memory configurations to assess performance under different workloads. Benchmark applications, including miniFE, miniQMC, and HPCG, are executed within each VM, and the host system gathers metrics to optimize resource allocation dynamically. The project aims to understand resource utilization patterns and implement strategies for efficient resource management.

Problem Statement

The project focuses on analyzing resource utilization in virtualized environments. Specifically, it studies three Alpine Linux VMs running on a Linux host. The host system monitors and manages the VMs, dynamically adjusting resources such as CPU and memory based on real-time metrics. Key objectives include setting up the VMs, monitoring their performance, running benchmark tests, and implementing online resource management strategies to optimize performance.

Project Goals

- Enable centralized monitoring and online resource management of VMs using QEMU/KVM and libvirt.
- Measure CPU, memory, network, and disk I/O metrics.
- Execute benchmark tests, including the ECP Proxy Apps Suite [4], which includes HPCCG [5], XSBench, RSBench [6], SimpleMOC [7], CoHMM [8], as well as miniFE, miniQMC, and HPCG.
- Implement online resource management:
  - Memory ballooning to dynamically adjust RAM.
  - CPU core allocation using `virsh setvcpus` [9].
  - Storage optimization using storage pooling and sharing strategies.

Project Structure

The project is organized into the following components:

1. VM Configuration and Deployment

- **Base Image Creation:** An Alpine Linux base image (`alpine-base.qcow2`) was created using the `alpine-virt` ISO. This base image serves as the foundation for all VMs.
- **VM Cloning:** Three VMs (`alpine-1.qcow2`, `alpine-2.qcow2`, `alpine-3.qcow2`) were created as clones of the base image using `qemu-img`.
- **XML Configuration:** A base XML template (`alpine.xml`) was used to generate unique XML configurations (`alpine-vm-1.xml`, `alpine-vm-2.xml`, `alpine-vm-3.xml`) for each VM. These configurations define the VM's resources, disk images, and network settings.
- **VM Definition and Management:** The VMs were defined using `virsh define` and started using `virsh start`. The `virsh` commands were also used for monitoring and managing the VMs.

2. Resource Monitoring and Metrics Collection

- **Metrics Reporting Script (`report-memory.sh`):** This script runs inside each VM to collect real-time metrics, including memory usage, CPU usage, disk I/O, and network statistics. The metrics are sent to the host system via a TCP connection.
- **Centralized Listener (`plot_adv.py`):** A Python-based listener running on the host system receives metrics from the VMs and stores them in CSV files. The listener also dynamically adjusts VM resources based on predefined thresholds.

3. Resource Management

- **Dynamic RAM Adjustment:** The `plot_adv.py` script uses libvirt APIs to dynamically increase or decrease the RAM allocated to each VM based on memory usage thresholds.
- **Dynamic CPU Core Adjustment:** The script also adjusts the number of CPU cores allocated to each VM based on CPU usage thresholds.
- **Gradual Deallocation (`memory_hog.c`):** A C program simulates memory-intensive workloads and gradually deallocates memory to test the system's ability to handle dynamic resource adjustments.

4. Visualization and Analysis

- **Dash-based Dashboard (`plot_adv.py`):** A web-based dashboard built using Dash displays real-time metrics for each VM. The dashboard includes:
  - Memory usage over time with high and low threshold indicators.
  - CPU usage trends.
  - Disk I/O activity.
  - Network usage (bytes received and transmitted).
- **Data Storage:** Metrics are stored in CSV files (`metrics_data/`) for historical analysis.

5. Automation and Scripts

- **Runner Script (`runner.sh`):** Automates the repeated execution of the `memory_hog` binary inside the VMs.
- **XML Generation Script (`gen-xml.sh`):** Automates the creation of VM XML configuration files from a base template.

6. Supporting Files

- **`pyproject.toml`:** Defines the Python project dependencies, including Dash, libvirt-python, pandas, and matplotlib.
- **`README.md`:** Provides an overview of the project and instructions for setup and usage.
- **`commands.txt`:** Contains useful `virsh` and `qemu-img` commands for managing the VMs.

Implementation Details

Dynamic Resource Management

The `plot_adv.py` script implements dynamic resource management using the following strategies:

- **Memory Scaling:**
  - If memory usage exceeds 80% of the allocated RAM, the script increases the RAM by 100 MB.
  - If memory usage falls below 20%, the script decreases the RAM by 500 MB.
- **CPU Scaling:**
  - If CPU usage exceeds 90%, the script adds an additional CPU core (up to a maximum of 2 cores).
  - If CPU usage falls below 30%, the script removes a CPU core (down to a minimum of 1 core).

Metrics Reporting

The `report-memory.sh` script collects the following metrics from each VM:

- **Memory Usage:** Total and used memory in KB.
- **CPU Usage:** Percentage of CPU utilization.
- **Disk I/O:** Disk read/write activity in KB/s.
- **Network Usage:** Bytes received and transmitted.

These metrics are sent to the host system every second using the `nc` (netcat) command.

Visualization

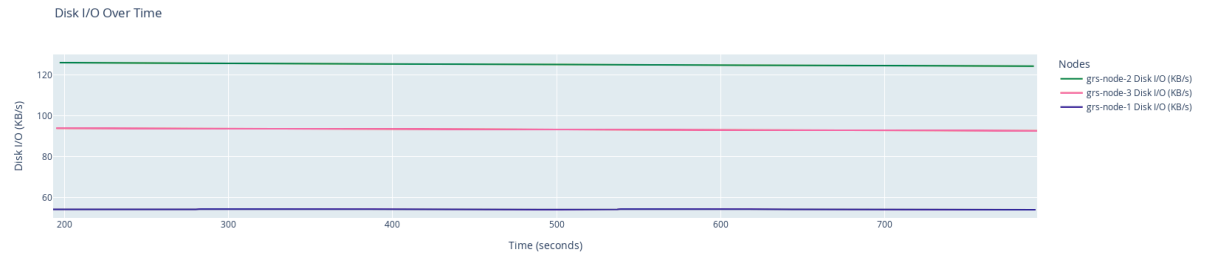
The Dash-based dashboard provides real-time visualization of the collected metrics. It includes:

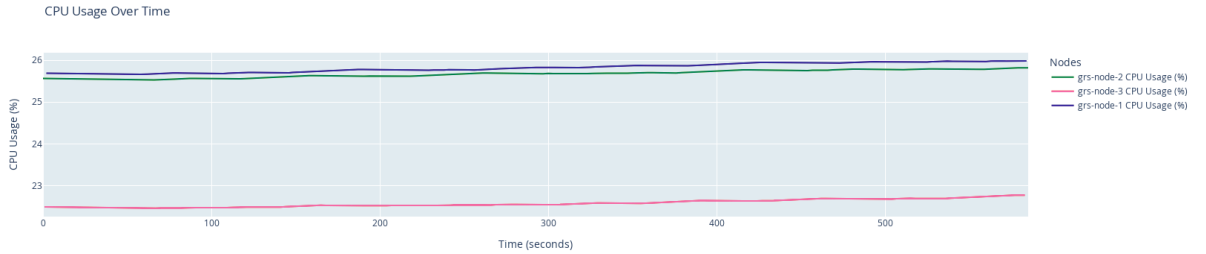
- **Memory Usage Graph:** Displays current and maximum memory usage with threshold indicators.
- **CPU Usage Graph:** Shows CPU utilization trends over time.
- **Disk I/O Graph:** Visualizes disk read/write activity.
- **Network Usage Graph:** Tracks bytes received and transmitted.

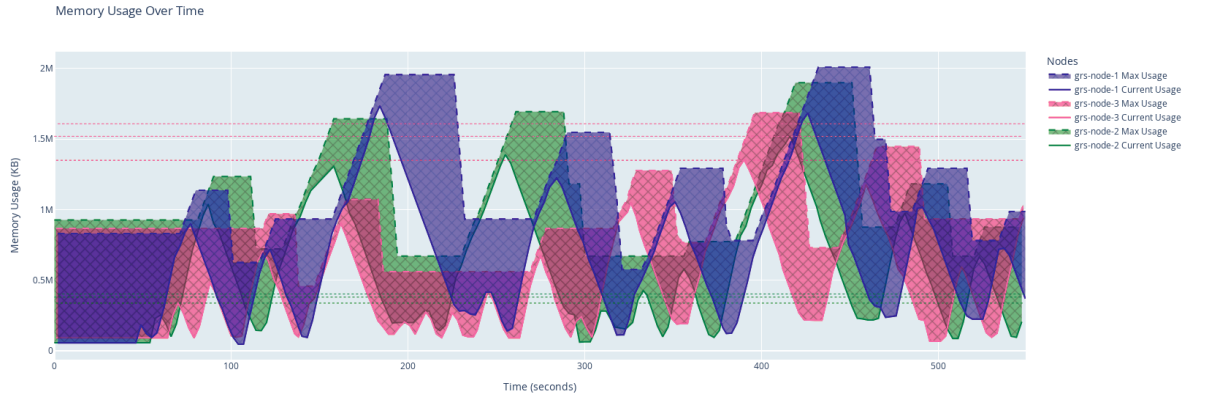
Project Outcomes

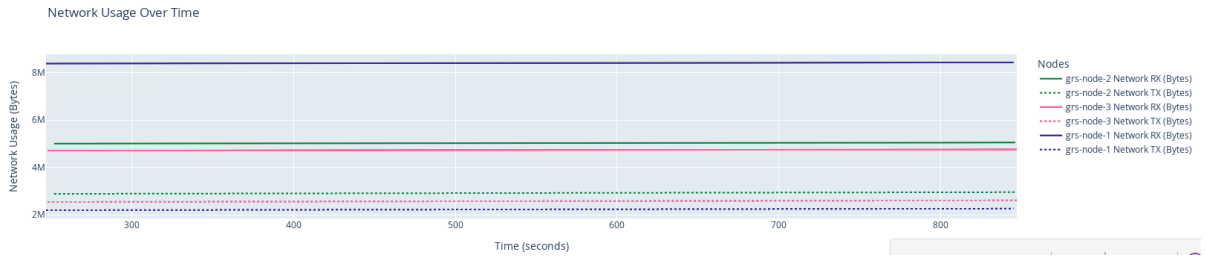
- A fully automated system for deploying and managing VMs using QEMU/KVM and libvirt.
- Real-time monitoring and visualization of VM metrics.

- Dynamic resource management strategies to optimize VM performance.
- Comparative analysis of the performance impact of different workloads under varying resource constraints.

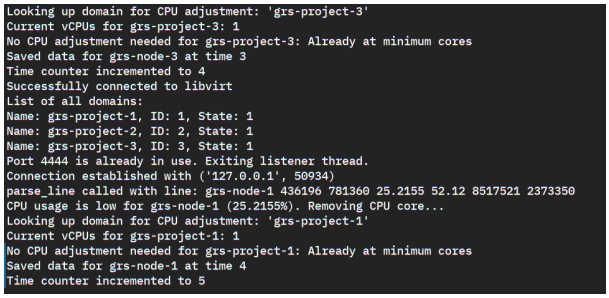









```
Deallocating ~100 MB. New target size: 0 MB
Buffer fully deallocated.
All memory deallocated. Exiting normally.
Binary exited with code 0. Restarting in 1s...
Starting /root/memory_hog...
Attempting to allocate initial 719 MB (754092906 bytes).
Successfully allocated 719 MB.
Filling memory with random data...
Finished filling memory. Current allocation: 719 MB.
Starting gradual deallocation (100 MB chunks per second)...
Deallocating ~100 MB. New target size: 619 MB
Deallocating ~100 MB. New target size: 519 MB
Deallocating ~100 MB. New target size: 419 MB
Deallocating ~100 MB. New target size: 319 MB
Deallocating ~100 MB. New target size: 219 MB
Deallocating ~100 MB. New target size: 119 MB
Deallocating ~100 MB. New target size: 19 MB
Deallocating ~100 MB. New target size: 0 MB
Buffer fully deallocated.
All memory deallocated. Exiting normally.
Binary exited with code 0. Restarting in 1s...
Starting /root/memory_hog...
Attempting to allocate initial 1712 MB (1795636881 bytes).
Successfully allocated 1712 MB.
Filling memory with random data...
```



Work Done

The following tasks have been completed:

1. **Host System Preparation:** Installed and configured QEMU, KVM, and libvirt on the host system.
2. **VM Deployment:** Created and configured three Alpine Linux VMs.
3. **Metrics Collection:** Implemented scripts for collecting and reporting VM metrics.
4. **Dynamic Resource Management:** Developed and tested memory and CPU scaling strategies.
5. **Visualization:** Built a real-time dashboard for monitoring VM metrics.
6. **Automation:** Automated the deployment and management of VMs using shell scripts and Python.

Future Work

- Extend the system to support additional resource types, such as storage and network bandwidth.
- Integrate advanced benchmarking tools for more comprehensive performance analysis.
- Implement machine learning algorithms to predict resource usage and optimize allocation.

Bibliography

[1] "QEMU." [Online]. Available: https://qemu.org/

[2] "The Linux Kernel Virtual Machine Project." [Online]. Available: https://linux-kvm.org/page/Main_Page

[3] "The Virtualization API." [Online]. Available: https://libvirt.org/

[4] "ECP Proxy Apps Suite." [Online]. Available: https://proxyapps.exascaleproject.org/ecp-proxy-apps-suite/

[5] "High Performance Computing Conjugate Gradients: The original Mantevo miniapp." [Online]. Available: https://github.com/Mantevo/HPCCG

[6] "rsbench: A Neuro-Symbolic Benchmark Suite for Concept Quality and Reasoning Shortcuts." [Online]. Available: https://unitn-sml.github.io/rsbench/

[7] "A 3D Method of Characteristics (MOC) reactor simulation Mini-App featuring MPI and OpenMP parallelism." [Online]. Available: https://github.com/ANL-CESAR/SimpleMOC

[8] “Heterogeneous Multiscale Method.” [Online]. Available: https://github.com/exmatex/CoHMM

[9] “Modifying the number of virtual CPUs.” [Online]. Available: https://www.ibm.com/docs/en/linux-on-systems?topic=cpus-modifying-number-virtual