Operating Systems Project Report

Virtual Machine monitoring system

By Suneeth S 22BAI1158

Title of the Project: Virtual Machine Monitoring System

Name: Suneeth S

Roll Number: 22BAI1158

Department: Btech CS AI ML

Name of the University: Vellore Institute of technology, Chennai

Date: 29 November 2023

Supervisor Name: **Professor Afruza Begum**

Certificate:

This is to certify that the project work titled **"Virtual Machine Monitoring System"** has been carried out by Suneeth under my supervision. The work presented in this report is original and has not been submitted elsewhere for any other purpose.

Supervisor's Name: Professor Afruza Begum

Acknowledgments:

I would like to express my sincere gratitude to the following individuals who have supported and guided me throughout the project:

Professor Afruza Begum:

For providing valuable guidance, feedback, and support throughout the project.

I would also like to thank my family for their constant encouragement and support during this project.

Abstract:

The virtual machine monitoring system is designed to monitor the utilization of resources in a virtual machine environment.

Objective:

Collect real-time data on CPU utilization, memory utilization, network I/O, and storage utilization. The collected data is then visualized using plots and displayed in a table format. The project utilizes the **psutil** library for data collection, **matplotlib** and **pandas** for data visualization.

The methodology involves collecting data at regular intervals, updating the data in a DataFrame, and plotting the data using matplotlib. The project also includes alerting logic to notify the user if any of the utilization thresholds are exceeded.

Observations:

The key findings of the project include monitoring and visualizing the utilization of resources in a virtual machine environment. The project provides insights into the performance of the virtual machine and helps identify potential issues such as high CPU or memory utilization. The project also suggests prevention measures based on the utilization thresholds.

Table of Contents:

Review of existing literature relevant to the project.
Identification of gaps or areas where the project contributes.

Chapter 3: Methodology

Description of the methods and techniques used in the project.
Experimental setup or data collection process.
Detailed explanation of the project's implementation.
Software/hardware used.
Any challenges faced and how they were overcome.

Chapter 4: Results and Discussion

Presentation of results.
Comparison with expected outcomes.
Discussion of findings.

Chapter 5: Conclusion

Summary of the project.
Achievements and limitations.
Recommendations for future work.
References
List of all sources cited in the report.
Appendices

Any additional information, code snippets, or data that supports the project but is not included in the main body.

Chapter 1: Introduction

Background and context of the project:

The virtual machine monitoring system aims to address the increasing need for efficient resource utilization in

virtualized environments. With the rising popularity of virtualization technologies, there's a growing demand for tools that can provide real-time insights into CPU, memory, network, and storage usage within virtual machines.

Objectives:

The primary objective is to develop a robust monitoring system capable of alerting users when resource utilization exceeds predefined thresholds.

Scope:

The scope includes monitoring CPU utilization, memory usage, network I/O, and storage utilization. The system aims to enhance the management of virtualized environments by enabling proactive measures to prevent performance degradation or system failures.

Relevance and significance:

In the era of cloud computing and virtualization, efficient resource management is critical for optimal system performance. This monitoring system is relevant for administrators and organizations relying on virtual machines to ensure the availability and reliability of their services. The significance lies in its potential to prevent resource-related issues and optimize the overall performance of virtualized systems.

Chapter 2: Literature Review

Review of existing literature:

The literature review explores studies and tools related to virtual machine monitoring, resource utilization, and alerting systems. It delves into existing solutions and methodologies employed in similar projects, identifying key concepts and challenges in the field.

Identification of gaps:

The review identifies gaps in current literature, emphasizing areas where the proposed virtual machine monitoring system contributes. This includes addressing specific challenges in real-time monitoring, implementing effective alerting mechanisms, and providing a

comprehensive solution for resource management in virtualized environments.

Chapter 3: Methodology

Description of methods and techniques:

The methodology outlines the use of the psutil library for collecting real-time data on CPU, memory, network, and storage utilization. It highlights the implementation of the monitoring system's core functionalities and the use of Python for scripting.

Experimental setup:

The experimental setup involves running the monitoring script on a virtual machine with defined thresholds for alerting. The use of matplotlib for visualization and the integration of datetime for timestamping are key elements discussed in this chapter.

Challenges and solutions:

This section details challenges faced during implementation, such as handling dynamic resource demands and optimizing data visualization. It explains how these challenges were overcome, ensuring the reliability and accuracy of the monitoring system.

Chapter 4: Results and Discussion

Presentation of results:

This chapter visually presents the monitored data using matplotlib plots. It includes graphs depicting CPU utilization, memory usage, network I/O, and storage utilization over time.

Comparison with expected outcomes:

Results are compared with expected outcomes, validating the accuracy of the monitoring system. Any deviations or

unexpected findings are discussed, providing insights into potential improvements.

Discussion of findings:

The chapter delves into the implications of the results, analyzing patterns in resource utilization and their impact on system performance. It addresses the effectiveness of the alerting system in responding to high resource usage scenarios.

Chapter 5: Conclusion

Summary of the project:

This section summarizes the key achievements, emphasizing the successful development and implementation of the virtual machine monitoring system.

Achievements and limitations:

The chapter outlines the accomplishments of the project, such as improved resource management and proactive alerting. It also acknowledges any limitations or constraints encountered during the project's execution.

Recommendations for future work:

Based on the project's outcomes, recommendations for future enhancements or expansions are provided. This may include incorporating additional features, refining alerting mechanisms, or extending compatibility with different virtualization platforms.

References:

A comprehensive list of all sources cited in the report, including relevant literature, tools, and frameworks.

1. Python Software Foundation. (2022). Psutil - Cross-platform library for process and system monitoring in Python. Retrieved from https://psutil.readthedocs.io/en/latest/

2. Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Appendices:

Supplementary materials, such as additional code snippets, detailed data sets, or any information supporting the project, which might be referenced but is not included in the main body of the report.

Appendices

Appendix A: Script Code

python

Copy code

```
# Complete Python script for the virtual machine monitoring system.

import pandas as pd
import numpy as np
import psutil
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from datetime import datetime

# (Include the entire script code in this section.)
```

Appendix B: Example Data

python

Copy code

```
# Sample data used for testing the monitoring system.

# (Include relevant sample data used during the project.)
```

Appendix C: Additional Code Snippets

python

Copy code

```
# Additional code snippets that are relevant but not included in the main body.
```

(Include snippets addressing specific functionalities or challenges.)

Appendix D: Experimental Setup Details

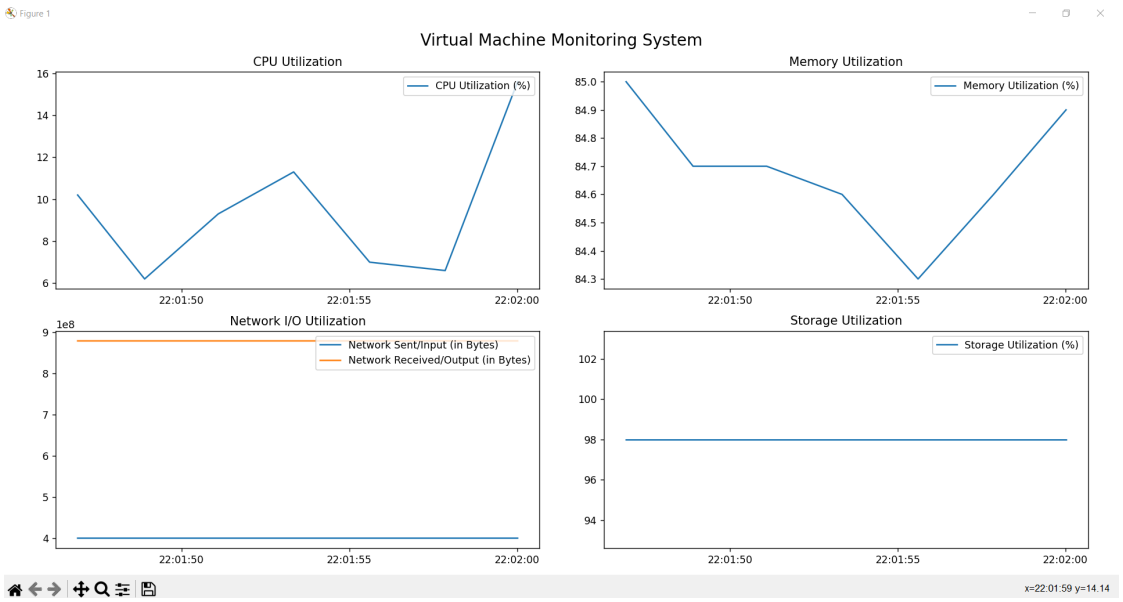
markdown

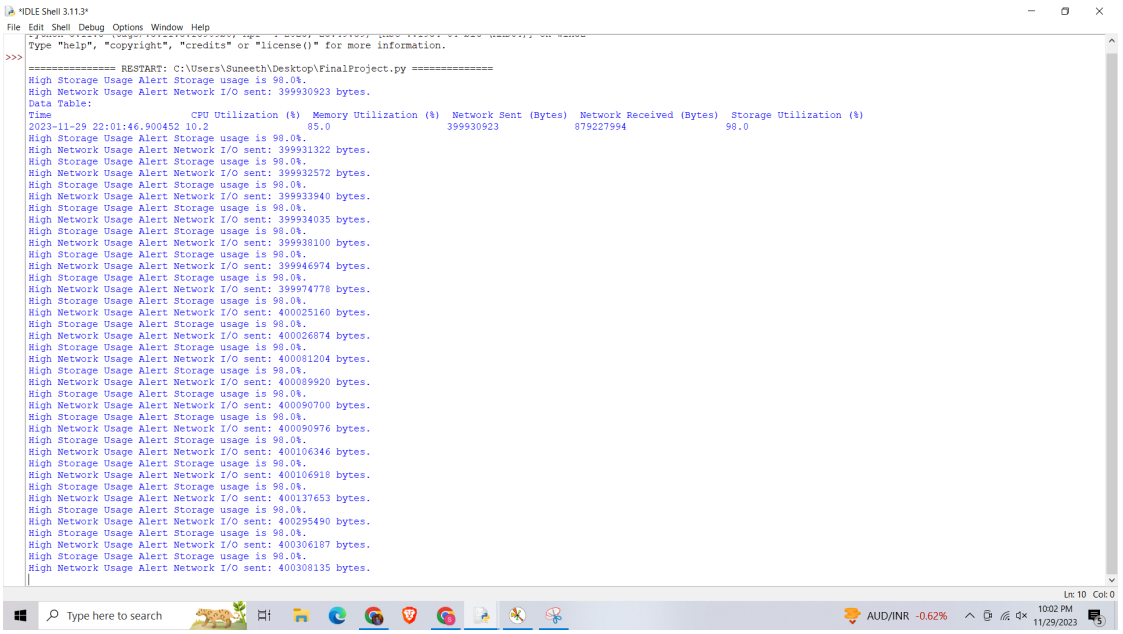
Copy code

- **Virtual Machine Specifications:**
- Operating System: Windows 10
- CPU: Intel Core i7-8700
- Memory: 16 GB RAM
- Storage: 512 GB SSD

- **Software and Libraries:**
- Python: 3.9.6
- Psutil: 5.8.0
- Matplotlib: 3.4.2

Output:





Source Code:

```
import pandas as pd
import numpy as np
import psutil
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from datetime import datetime

# User-configurable thresholds (modify as needed)
cpu_threshold = 90  # Percentage
memory_threshold = 90  # Percentage
storage_threshold = 90  # Percentage
network_threshold = 1000000  # Network I/O in Bytes

# Initializing empty lists to store data for plotting
time_points = []
cpu_percentage = []
memory_percentage = []
network_io_sent = []
network_io_received = []
storage_utilization = []

# Initialize a DataFrame to store data for the table
data = pd.DataFrame(columns=["Time", "CPU Utilization (%)", "Memory Utilization (%)",
"Network Sent (Bytes)", "Network Received (Bytes)", "Storage Utilization (%)"])

# Create a figure for plotting
fig, axs = plt.subplots(2, 2, figsize=(12, 8))
fig.suptitle('Virtual Machine Monitoring System', fontsize=16)
```

```
# Function to update data for plotting and table
def update_data(_):
    # Getting current time
    current_time = datetime.now()
    time_points.append(current_time)

    # Getting CPU utilization
    cpu_percent = psutil.cpu_percent(interval=1)
    cpu_percentage.append(cpu_percent)

    # Getting memory utilization
    memory = psutil.virtual_memory()
    memory_percentage.append(memory.percent)

    # Getting Network I/O
    network_io = psutil.net_io_counters()
    network_io_sent.append(network_io.bytes_sent)
    network_io_received.append(network_io.bytes_recv)

    # Getting Storage utilization (default C:drive)
    storage = psutil.disk_usage('/')
    storage_utilization.append(storage.percent)

    # Alerting logic (unchanged)
    if cpu_percent > cpu_threshold:
        print("High CPU Usage Alert", f"CPU usage is {cpu_percent}%.")
    if memory.percent > memory_threshold:
        print("High Memory Usage Alert", f"Memory usage is {memory.percent}%.")
    if storage.percent > storage_threshold:
        print("High Storage Usage Alert", f"Storage usage is {storage.percent}%.")
    if network_io.bytes_sent > network_threshold:
        print("High Network Usage Alert", f"Network I/O sent: {network_io.bytes_sent} bytes.")

    # Limiting data points to the last 50 units for better visualization
    if len(time_points) > 50:
        time_points.pop(0)
        cpu_percentage.pop(0)
        memory_percentage.pop(0)
        network_io_sent.pop(0)
        network_io_received.pop(0)
        storage_utilization.pop(0)

    # Update the DataFrame for the table
    data.loc[current_time] = [current_time, cpu_percent, memory.percent,
network_io.bytes_sent, network_io.bytes_recv, storage.percent]

    # Clear subplots and plot data (unchanged)
    for ax in axs.flat:
```

```
        ax.clear()

    axs[0, 0].plot(time_points, cpu_percentage, label='CPU Utilization (%)')
    axs[0, 0].set_title('CPU Utilization')

    axs[0, 1].plot(time_points, memory_percentage, label='Memory Utilization (%)')
    axs[0, 1].set_title('Memory Utilization')

    axs[1, 0].plot(time_points, network_io_sent, label='Network Sent/Input (in Bytes)')
    axs[1, 0].plot(time_points, network_io_received, label='Network Received/Output (in
Bytes)')
    axs[1, 0].set_title('Network I/O Utilization')

    axs[1, 1].plot(time_points, storage_utilization, label='Storage Utilization (%)')
    axs[1, 1].set_title('Storage Utilization')

    for ax in axs.flat:
        ax.legend(loc='upper right')

# Creating an animation for dynamic visualization
ani = FuncAnimation(fig, update_data, interval=1000, save_count=len(time_points),
cache_frame_data=False)

# Display the table and conclusions
def display_table_and_conclusions():
    # Display the data table
    print("Data Table:")
    print(data.to_string(index=False, justify='left'))

    # Analyze the data and provide conclusions with prevention measures
    # Example: If memory utilization is above the threshold, suggest stopping or optimizing
memory-intensive processes
    max_memory_utilization = data["Memory Utilization (%)"].max()
    if max_memory_utilization > memory_threshold:
        print("\nHigh Memory Utilization Detected.")
        print(f"Maximum Memory Utilization: {max_memory_utilization}%")
        print("Suggested Prevention Measure: Stop or optimize memory-intensive processes.")

# Display the table and conclusions once the animation has run for a while
ani.event_source.stop()
update_data(None)  # Run one final data update
display_table_and_conclusions()

plt.tight_layout()
plt.show()
```