بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

International Islamic University Chittagong

Department of Computer Science and Engineering

Lab Report

COURSE CODE : CSE-3632
COURSE TITLE : Operating System Lab
NAME OF THE PROJECT : system resource monitoring tool
DATE OF SUBMISSION : 28.06.2024

Submitted By:
Shumaita Binte Burhan
ID : C213225
Semester : 6th
Section: 6AF
Department of CSE, IIUC

Submitted To:
Mohammad Zainal Abedin
Assistant Professor,
Email: jakcse99@gmail.com
Department of CSE, IIUC

Introduction

The system monitor is a tool built with Python, using tkinter and matplotlib. It provides real-time monitoring of CPU usage, memory usage, and network activity through easy-to-understand graphs. This tool helps users optimize their computer's performance, detect issues early, and manage resources effectively. Whether for personal or professional use, it's a valuable tool for keeping systems running smoothly and efficiently.

Objective

The objective for this project is to develop a comprehensive system resource monitoring tool comprising CPU, Memory, and Bandwidth monitors. The primary goals include:

1. **Real-time Monitoring:** Provide continuous updates on CPU usage, memory utilization, and network activity.

2. **Graphical Representation:** Display performance metrics through clear and informative graphs using matplotlib.

3. **Resource Optimization:** Assist users in optimizing system performance by monitoring resource usage trends.

4. **User-Friendly Interface:** Offer an intuitive interface through tkinter for easy navigation and data visualization.

Working Procedure

For my project, the above image is a demo as output. I also represented the CPU and Memory Usage with bars and percentage. And the Bandwidth displays the MB sent, received and total in 2 decimal points. Each of these monitored values update in every 0.5 seconds, giving the users the most recent values from which the user can monitor accordingly. Implement a graphical user interface (GUI) for improved user experience.

Initialization:

- The SystemMonitorApp class is instantiated with the root Tkinter window.
- The main window's title and size are set.
- Frames for CPU, Memory, and Network monitoring sections are created.
- Labels and progress bars for CPU and Memory usage, and a label for Network usage are set up.
- Variables to track the last network data received and sent are initialized.
- Lists to store data for CPU, Memory, and Network usage over time are initialized.
- Graphs for CPU, Memory, and Network usage are created using Matplotlib.
- The monitoring update loop is started.

Frame Creation (create_frames method):

- Three labeled frames are created for CPU, Memory, and Network sections.
- These frames are added to the main window with padding.

Widget Creation (create_widgets method):

- A label and progress bar for CPU usage are created and added to the CPU frame.
- A label and progress bar for Memory usage are created and added to the Memory frame.
- A label for Network usage is created and added to the Network frame.

Graph Creation (create_graphs method):

- Figures for CPU, Memory, and Network usage graphs are created using Matplotlib.
- Each figure is embedded into the respective frame using FigureCanvasTkAgg.

Monitoring Update Loop (update_monitor method):

- CPU and Memory usage data are retrieved using psutil and displayed on progress bars.
- Network usage data is retrieved, and the amount of data received and sent since the last update is calculated and displayed.
- CPU, Memory, and Network usage data are appended to their respective lists.
- The graphs are updated with the new data.
- The method schedules itself to run again after 1 second, creating a continuous monitoring loop.

Future Scope

Enhance visualization with detailed, interactive graphs and historical data analysis. Allow customizable monitoring intervals and metrics, including disk and GPU usage. Implement threshold-based alerts and custom alert rules. Support continuous data logging and export in formats like CSV and JSON. Enable remote monitoring of multiple systems and cloud integration. Optimize performance with efficient data handling and multi-threading. Extend cross-platform support to Linux, macOS, and Windows. Improve UI responsiveness and customization options. Develop an API for integration with other tools like Nagios and Grafana. Ensure security with user authentication and data encryption.

Conclusion

The System Monitor application provides a comprehensive and real-time view of CPU, memory, and network usage on a system. By leveraging Tkinter for the UI and Matplotlib for data visualization, it offers an intuitive and user-friendly experience. The current implementation successfully tracks and displays system metrics, while future enhancements aim to make it more versatile, customizable, and efficient. This project lays a strong foundation for a robust system monitoring tool that can cater to both casual users and advanced professionals.

References

1. Nine, N. (2022, February 28). Bandwidth Monitor in Python. YouTube. Retrieved June 15, 2023, from https://youtu.be/O76lnYFvbTU 2. Nine, N. (2022, June 26).
2. CPU & RAM Usage Monitor in Python. YouTube. Retrieved June 15, 2023, from https://youtu.be/rdxt6ntfX24 3. Tabassum, A. (2023, May 9).
3. CPU, Memory and Bandwidth Monitor. CPU, Memory and Bandwidth Monitor- Github. Retrieved June 15, 2023, from https://github.com/blurryface14/CPU-memory-and-bandwidth-monitor.git
4. Psutil Documentation**: https://psutil.readthedocs.io/**
5. Tkinter Documentation**: https://docs.python.org/3/library/tkinter.html**
6. Matplotlib Documentation**: https://matplotlib.org/stable/contents.html**
7. Python Official Documentation**: https://docs.python.org/3/**