Tool to monitor the overall system resource utilization

Name:Antarjita Mandal

SRN:PES220100409
University: *PES UNIVERSITY*
Email: antaram215@gmail.com

Dept:Computer Science(B TECH)
Bangalore,India
Semester: 5th
Section:D

Name:Sai Shruthi S

SRN:PES220100361
University: *PES UNIVERSITY*
Email: saishruthi.sistla@gmail.com

Dept:Computer Science(B TECH)
Bangalore,India
Semester: 5th
Section:D

***Abstract*—The overall system resource utilization during a particular time is retrieved and logged into a csv file. This data is analysed to alert the user in case of any resource threshold violations.Real time graphs are plotted to understand the trends. Models are built to predict the cpu and memory utilization.**

Keywords—cpu, memory, utilization, system resources, threshold

I. Introduction

Performance monitoring is useful in developing and refining systems. Records of operating system and process performance can be used to quantify changes to the system and allow accurate comparisons to other systems. They can also be used to predict the performance of similar systems and what type of performance gains may be expected in the future [1]. Our key goal is to alert the user in case they use more resources than they allocate. This is to allow smooth and efficient functioning of the system. Also to understand the systems behaviour it becomes essential to analyze the parameters which have a significant impact on the performance of the system like processor time, memory utilized, pages swapped in and out etc. Easy to eyes graphic visualizations and plots help emphasis on further scope of improvement and highlight the pros and cons of a particular system.

II. Problem Statement

The problem in hand is to build a tool that will monitor the overall utilization of system resources like CPU and memory. In order to achieve this , the system parameters like CPU and memory have to be continuously retrieved and logged into files. This data has to be analysed in order to alert the user in case the utilization crosses the specified threshold. Real time graphs showing the resource utilization have to be displayed to the user. The data logged over time has to be utilized to build models to predict the CPU and memory utilization.

III. Proposed Solution

Our solution to the problem mainly consists of two parts :

A. Background Process

This process runs in the background without intimidating the user in any way other than a log text file solely used for this purpose. The key idea is to keep copying the files needed (or wish to analyse) in the proc directory (which is a virtual file system keeping track of all the system related information like CPU, Memory, Virtual memory etc)[2] in the Unix operating system from the kernel space to the user space to avoid permission violations and to avoid read and write locks implemented on this directory when the kernel wishes to modify it. The alarm set in this code signals the process to run regularly at that specified time interval.[3] At the end of each run we get the updated version of the files in the proc directory deleting all the previous versions.

B. User process

The user process can be invoked at any point in time.The user has to specify the duration (in seconds) for which the user program should run in terms of the number of times to be run and after how much duration(in seconds). The user must also specify the threshold values for CPU and memory utilization . When invoked each time, the user process will fetch the current version of the proc files from the present working directory . These proc files are then read to obtain the required parameters. The parameters which are not directly obtained are calculated from the available data. In case of a threshold violation, a danger message is displayed along with a beep sound. The user is also warned in advance through a warning message if the threshold limit is approaching .The final results are then logged into a csv file along with the timestamp. Once the user process finishes its execution , a summary of the CPU and memory utilization is given through graphs.

The data collected over time is provided as an input to the ARIMA model [4] to forecast the future values of CPU and Memory utilization based on the past values, so that we are well aware of the situation that lies ahead.

IV. System Requirements

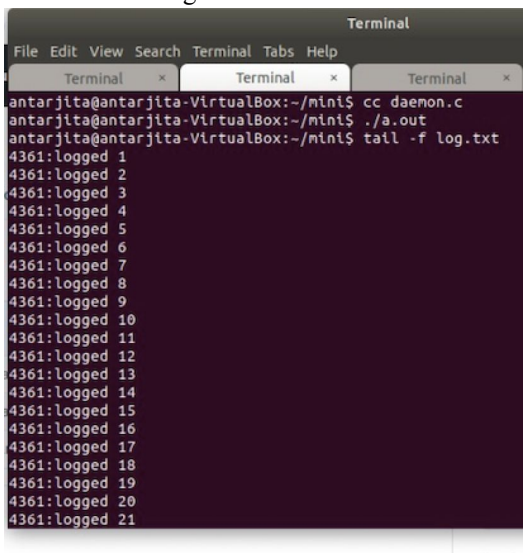
Our approach uses the /proc directory which is specific to the Unix OS.
Also a system with gcc compiler and python 3 installed is essential to run our code.

.

V. Test Cases

To run the background process code, simply run the following commands:-

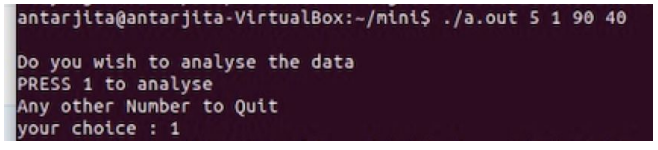
1. cc daemon.c
2. ./a.out
3. tail -f log.txt



Now we can see the proc directory copied onto the current directory as shown.

To run the user code simply type:-

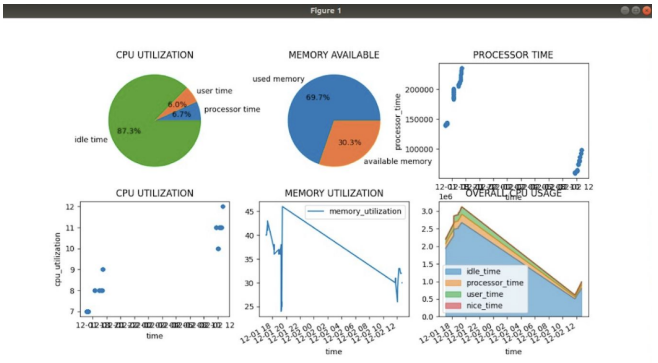
1. cc SystemResources.c
2. ./a.out <number of times the proc file should be polled> <delay in seconds>< cpu threshold> <memory threshold>



If any violations in the threshold are logged into the .csv file then a notification along with a beep sound is given.

A buffer of -10% is considered susceptible to warning.

Also a provision to view all the logs in graphical format is provided.



Further detailed analysis of predicting the CPU and Memory are forecasted using the ARIMA model.

VI. Conclusions

With this we conclude that System Resource monitoring is essential to improve overall efficiency. With our code you can be aware of % of CPU,MEMORY to be used for continuous and efficient work load balance. Also future status of CPU and Memory can be predicted so that the user can be well ahead in time to not overload their system.

VIII. Acknowledgment

Working on this project has been an enriching experience. We would like to thank Prof. Anand Subbarao and Prof. Kiran for the guidance and support provided by them in completing this project.

References

[1] Brooks, A. (n.d.). *Operating System and Process Monitoring Tools*. Retrieved December 4, 2020, from https://www.cse.wustl.edu/~jain/cse567-06/ftp/os_monitors/

[2] *proc(5) - Linux Manual Page*. https://man7.org/linux/man-pages/man5/proc.5.html. Accessed 4 Dec. 2020.

[3] *Creating a Daemon Process in C Language with an Example Program*. https://www.thegeekstuff.com/2012/02/c-daemon-process/. Accessed 4 Dec. 2020.

[4] "The Use of ARIMA Models for Reliability Forecasting and Analysis." *Computers & Industrial Engineering*, vol. 35, no. 1-2, Pergamon, Oct. 1998, pp. 213–16.