ISSN (Print): 0974-6846
ISSN (Online): 0974-5645
Indian Journal of Science and Technology, *Vol 12(29), DOI: 10.17485/ijst/2019/v12i29/146973, August 2019*

Performance Analysis of Real-Time System

Kiran Wamanacharya* and Prashanth V. Joshi

School of Electronics and Communication Engineering, REVA University, Yelahanka, Bengaluru – 560064, Karnataka, India; gitkiran@gmail.com, prashanthvjoshi@reva.edu.in

Abstract

Objective: Performance analysis of a Real-time system is very important aspect where the performance bottlenecks can be identified via detailed early analysis. The objective is to display these Performance parameters and analyze. **Methods/ Statistical Analysis:** Details like CPU Utilization, Memory Utilization and Heat dissipation add on to the list of critical measurement of the system. In this study, the procedure to display the mentioned aspects in a graphical user interface when an application is being executed on the Raspberry Pi controller and storing of these parameters in a data-base for efficient analysis is mentioned. **Findings:** With the approach mentioned here the required parameters analyzed for a given embedded firmware which helps in analyzing the performance of the real time embedded system targeted to the Rasperry Pi platform. **Application:** This model can be used for the programs that have been developed on Raspberry PI for efficient analysis and optimization.

Keywords: LINUX, Performance, Profiler, Women Employees

1. Introduction

The analysis of the performance parameters has been done from the time of its necessity to Optimize a given system[1] also known as Profiling. The dynamic profiling is discussed in which the Application is being executed[2]. Various parameters determine the performance, some of the parameters considered are:

- *CPU Utilization*: Finding CPU utilization is one of the important tasks, it is the overall work handled by a Central Processing Unit. Variation can be there according to the type and amount of tasks getting executed as some tasks require lot of CPU time while others require less. To obtain the CPU Utilization there are various approaches, one such approach is discussed further in this paper
- *Memory Usage*: For a given real time system, the memory plays a vital role as it stores program and the data being used during the course of program execution. This parameter always challenges the designers. The Memory Utilization metric is an average utilization metric derived from the percentage of available memory in use at a given moment, averaged over the sampling interval. Conditions like overutilization of the memory which in turn indicate decreased performance of the corresponding application processes, and memory leaks can be identified using this parameter.
- *Monitoring the Core Temperature*: One of the important parameter for understanding the heat dissipation of the controller. The parameter will be useful to design the Heat sinks if needed. Also, closely monitor if the controllers are being operated in the prescribed temperature limits, else these will be non-functional. These parameters along with DMA Access and Cache Miss/Hit will be monitored for the analysis in this Paper.

2. Proposed Model

The module comprises:

1. Raspberry Pi with Raspbian Operating system.
2. A Camera for image capturing.
3. Host system.

*Author for correspondence

The embedded firmware is targeted to the Raspberry Pi platform. As a case study we have analyzed the embedded firmware developed for camera image processing.

The various performance parameters are extracted by following methodology.

3. CPU Utilization

"sudoiostat–c[2] is the system command used to get the current CPU utilization. The results are detailed in Figure 1. The output is read as below (in percentage):

- *% user*: CPU usage that has occurred while executing the given application.
- *% nice*: CPU usage while executing the user application with nice level priority.
- *% system*: CPU usage while executing at the system (kernel) level.
- *% iowait*: CPU idle time during which the system had to complete the pending disk I/O request.
- *% steal*: time spent involuntarily wait by the CPU while the hypervisor was servicing another virtual processor.
- *% idle*: time of which the CPU is idle without executing anything.

09:03:29	IST	CPU	%user	%nice	%syste
09:03:31	IST	all	2.77	0.00	0.8
09:03:33	IST	all	2.89	0.00	1.6
09:03:35	IST	all	1.26	0.00	0.5
Average:		all	2.31	0.00	1.01

Figure 1. CPU utilization result for the command "sudoiostat –c".

The command "Top" can also be used for the same.

4. Memory Usage

The command "free[3] is used for getting the Memory usage status. The result of the usage of the same is detailed in Figure 2. In the result, number that follows "–/+ buffers/cache:" is the amount of free physical memory which is available in the system (kilobytes). The same number also gives the information about Random Access Memory that is currently not being used by the Kernel for disk caching. The 3rd column in the 'Mem:' line signifies the amount of physical memory that is currently been unallocated by the kernel for any reason.

RAM_usage.py

Details

	total	used	free	shared	buffers	cached
Mem:	1832	1002	829	128	47	458
-/+ buffers/cache:		496	1335			
Swap:	1943	0	1943			

Figure 2. Memory usage result for the command "free –m".

5. Getting the Core Temperature

In the Raspberry PI terminal the command "/opt/vc/bin/vcgencmd measure_temp[4] has been used to get the core temperature in Degrees Celsius. The obtained results in shown in Figure 3. The same can be updated further to signal an over temperature.

```
pi@raspberrypi:~ $ /opt/vc/bin/vcgencmd measure_temp
temp=56.9'C
```

Figure 3. Core temperature using vcgencmdmeasure_temp.

6. Results and Discussion

The results (Figure 1-3) are obtained when there is no application program getting executed. Only the program for executing the commands is active. Similar results can be obtained while application programs like complex image processing are under execution.

The results shows that various parameters can be analyzed by using the proposed methodology and thus the performance of embedded system can be easily analyzed. Also, if there is a similar system available, then these metrics help in deciding which system performs well and is efficient. Not only for the above mentioned reasons, other important aspect is that performance parameters are measured to help the designer to optimize the system by[5] acting as a feedback mechanism during the early phase of the design.

7. Conclusion

LINUX provides elaborate commands together with Raspberry Pi which itself is a powerful device, one can have many Parameters listed and the same can be analyzed. We executed an Image processing Application and switch to other applications to understand the change in Performance parameters. This has an enormous potential for the following and many more fields:

1. Hardware Prototyping: A test code can be executed and all the critical parameters can be obtained.
2. Efficient Hardware design: Debugging with the parameters obtained can be efficient.
3. Effective scheduling of the processes in OS.
4. Caching mechanism can be improved.
5. Efficient memory handling.

8. References

1. Janjusic T, Kartsaklis C. Glprof: A Gprofinspired Callgraph-oriented Per-Object Disseminating Memory Access Multi-Cache profiler, Procedia Computer Science. 2015; 51: 1363–72. https://doi.org/10.1016/j.procs.2015.05.324.
2. How to Monitor Linux Systems Performance with Iostat Command. Date accessed: 12/02/2018. https://www.linuxtechi.com/monitor-linux-systems-performance-iostat-command/.
3. IBM. Date accessed: 23/08/2019. https://en.wikipedia.org/wiki/IBM.
4. Monitor the Core Temperature of your Raspberry. Date accessed: 03/07/2017. https://medium.com/@kevalpatel2106/monitor-the-core-temperature-of-your-raspberry-pi-3ddfdf82989f.
5. Thiele L. Performance Analysis of Distributed Embedded Systems. In: International Conference on Embedded Software; 2007. p. 10. https://doi.org/10.1145/1289927.1289933.