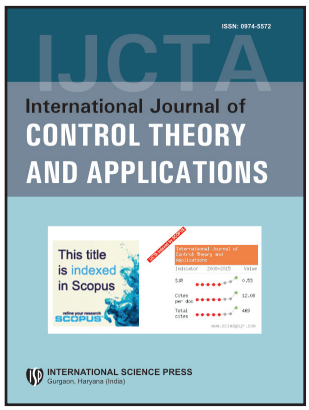

International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 9 • Number 44 • 2016

A Survey on Response Time Analysis Using Linux Kernel Completely Fair Scheduler for Data Intensive Tasks

Rucha Shankar Jamale[a], Sunita Dhotre[b] and Pooja Tanaji Patil[c]

[a]*M.Tech, Department of Computer Engineering, Bharati Vidyapeeth Deemed University, Pune, India. Email: rjrucha98@gmail.com*
[b]*Associate Professor, Department of Computer Engineering, Bharati Vidyapeeth Deemed University, Pune, India. Email: ssdhotre@bvucoep.edu.in*
[c]*M.Tech, Department of Computer Engineering, Bharati Vidyapeeth Deemed University, Pune, India. Email: poojapatil2829@gmail.com*

Abstract: Nowadays, pocket sized mobile devices with updated mobile operating system are used over a huge range. They provide numerous facilities and features which eventually ease our day to day requirements. However, while providing these variant attributes one major problem is Battery limitation.

This paper surveys earlier discussed solutions on mobile device's battery capacity as well as it also gives a short cogitation of analysis of response time of battery oriented mobile devices with the help of Linux Kernel Completely Fair Scheduler (CFS) and Dynamic Voltage & Frequency Scaling (DVFS) techniques.

Keywords: Mobile Devices, Battery Limitation, Response Time, Completely Fair Scheduler (CFS), Dynamic Voltage & Frequency Scaling (DVFS).

1. INTRODUCTION

IBM's BlueGene[1] and many other fastest supercomputers have Linux as their main operating system. Linux has also ported to various handheld devices such as Apple's iPod and iPhone. Main platforms such as smartphones, tablets, smartTVs, Android, Firefox OS, Mer, and Tizen use Linux platforms[2]. However, these portable devices have limited power capacity and while running multiple tasks at a time battery limitations arises.

Modern Mobile Operating systems[3] possess more features compared to the old cellular handheld phones. However, more features require more battery consumption. Unfortunately, battery lifetime[4] of modern mobile devices is fragile.

Modern mobile devices are observing huge growth regarding efficiency, performance, and multitasking to meet the user needs. Due to this many electronic consumers are motivated to transform their system regarding products quality of experience. Hence in overall, there should be an appropriate balance between modern

technology implementations and battery limitations[3][4]. A failure to this may lead significant degrade in the quality of experience.

Process scheduling [5][26] is a vital part of Multiprogramming operating systems. This kind of operating systems [5] permits multiple processes loaded into the executable memory at a time; these full process shares the CPU using time multiplexing.

The Linux scheduler [6][7] is a priority based scheduler which schedules tasks based upon their static and dynamic priorities. Linux uses dynamically allocated process priorities for nonreal time processes. The earlier versions of Linux[8], Linux pre 2.5 and Linux pre 2.5-2.6.23 used Multilevel feedback queue and O(1) scheduler respectively, while the later versions of Linux post 2.6.23 use Completely Fair Scheduler. The basic idea of CFS[7][8][9] is its even balance (fairness) in providing processor time to the processor.It means a fair amount of the processor is given to the processes. CFS represents this balance in fairness via the per-task wait runtime (nanosecond-unit) value.

Various open source communities use Linux as their principal operating system. Due to increasing Linux users, Linux kernels[27][28] CPU schedulers are enhanced with better performance and effectivity.

2. RELATED WORK

In this paper R. C. Garcia, J. M. Chung, S. W. Jo, T. Ha, and T. Kyong [10] proposed Response time, performance estimation scheme for smartphones by applying Dynamic Voltage & Frequency Scaling (DVFS)[11]scheme at CPU and CFS at Linux kernel.

DVFS techniques influence smartphone's overall time performance since DVFS works at CPU and change in operational frequency at CPU indirectly affects access speed and responsiveness for task execution. While CFS also has a significant impact on task execution at Linux Kernel level.

The proposed scheme here is implemented on LG Optimus G smartphone Currently, all androids use CFS at Linux kernel level because CFS is default scheduler of Linux.CFS works under fairness mechanism. Also, CFS can apply priority mechanism on different task operations through nice value control. All the above leads to a significant influence on the response time of applications under execution.

Therefore the proposed system works as a response time estimator to analyze above effects under different load conditions.

A. CFS

Completely Fair Scheduler[7][8][9][29] deals with Ideal multi-tasking CPU which means CPU with 100% power and can execute each task at an equal speed, in parallel, each at 1/nr_running speed. The CFS tries to eliminate unfairness from the system. In a system, CFS keeps track of fair share of the CPU which is allocated to every process. Hence, CFS runs an equitable clock at a fraction of real CPU clock speed.

CFS is the default scheduler of Linux kernel; recently all Android smartphone use CFS scheduler. The ultimate goal of CFS is to provide the fair amount to all the tasks proportional to their weights. In CFS algorithm weight of each task is decided by each tasks nice value, when the nice value of a certain task is decreased by one, then the weight of the task is increased by 1.25 times.

CFS is a virtual runtime scheduler. The CFS algorithm uses Red-Black Tree, in this tree the tasks are sorted in a tree form from left to right according to the increasing order of their respective virtual run times. Meanwhile, CFS executes its task initiating from left most leaf moving towards the right.

B. DVFS Techniques

DVFS techniques[12][13] are widely applied in smartphones to reduce power consumption by changing CPU core frequency and system voltage, and eventually, this results in variance in response time in smartphones while executing a certain application.

DVFS schemes include different governors [14] like Ondemand governor, Performance governor, Conservative Governor, Powersave Governor and Interactive governor.

Ondemand governor [15] is the default governor of maximum Android-based smartphones. Ondemand governor was introduced in the Linux Kernel 2.6.10. Depending on the processor utilization it dynamically changes the processor frequency. The use of the processor is checked, and if the value exceeds the threshold, this governor set the frequency to the highest available value. If the utilization is less than the threshold, the governor steps down the frequency. The range of frequencies can be controlled by the governor and also the rate of checking the utilization of the system.

The proposed system [10] works on Optimus G Smartphone. The working here is particularly explained with respect to different load situations apparently, run in the background which helped to analyze Response Time Estimation and Response Time Performance.

The two most important terms which helped the result computation are Instantaneous event unit and Time measurement unit. Hence the proposed system effectively captured the variation in response time during a change in CPU frequency and applications running background.

C. S. Wong, I. K. T. Tan, R. D. Kumari, J. W. Lam, and W. Fun [16] explained in their paper that two schedulers are used for scheduling technique firstly O(1) and secondly CFS. O(1) scheduler was used for earlier versions of Linux kernel later it get replaced with CFS.

The main function of CFS is to maintain fairness between the executing processes. Moreover, at a particular extent, this characteristic of CFS is redeemed all over. However, fairness of CFS is not proved practically. Hence this paper has a centralize approach towards the scientifical demonstration of CFS and O(1) by resulting factual solutions with genuine computations.

O(1) scheduler[8] is an improvement over Linux foremost Scheduler O(n). O(1) had a significant refinement in Java virtual machines concerning in handling sizeable executing threads. In O(1) each processor has a run queue, and each run queue keeps track of all runnable tasks and programs analog with the CPU using two arrays Active array and Expired array. It is a priority based scheduler.

CFS [7] is the default scheduler of Linux Kernel. Its main function is to have a fair share amongst all the processor. Therefore it takes equal time for all processes; it is also responsible for CPU utilization and resource allocation. The main idea behind CFS is its working which is purely based on priority and timeslice mechanism. In CFS highest timeslice is received by a process having the highest priority. In CFS process management is done by using Red-Black Tree. Red Black Tree is a binary search tree and also known for its self balancing technique. Comparison of both the schedulers is carried out by Interactivity and Fairness test. This assessment proved that CFS has a benefit of fair CPU bandwidth distribution and interactivity performance. This is clarified by the enactment of execution of each task which is allocated with a fairly divided time slice. CFS is also prior in terms of an algorithm for evaluations of interactive tasks.

In this paper, J. Wei, E. Juarez, M. J. Garrido, and F. Pescador [17] implemented Energy based fair queuing (EFQ) scheduling algorithm. EFQ is used for maximizing the user experience in battery limited mobile systems. EFQ relies on energy oriented scheduling algorithms which support balanced energy usage

and effectual time restraint compliance. This paper shows how exactly EFQ is more flexible than Linux scheduler.

This article improves the working of EFQ and plays a vital role by maximizing the user experience in battery oriented mobile devices. The main work here deals with contributing traditional fair queuing algorithm regarding energy domain. The analysis is done by the help of test bench tool which is created based on Linux scheduler to verify the proposed algorithm here. Due to this new testbench EFQ properties are analyzed in an appropriate manner with ease and no flaws. Also, EFQ scheduler here is compared with Linux default scheduler to show its advantage on enhancing user experience in battery limited mobile devices.

In this paper, J. Wei, R. Ren, E. Juarez, and F. Pescador [18] explained the implementation of Energy base Fair Queuing (EFQ) Linux based scheduling algorithm. EFQ is an improvement over traditional fair queuing algorithm. The main characteristic of EFQ is proportional power share into the system.

This paper concentrates on improving the implementation of EFQ algorithm with the help of testbench Pthread in several ways.

1. MiBench an open source benchmark suite is also used to program the task under test. Three tasks are programmed here interactive, batch and real time and these tasks are tested under EFQ scheduling algorithm.
2. Hardware metering measures the power consumption of selected benchmarks and the obtained outputs in terms of energy values is given as an input to Pthread based testbench
3. The total power consumption also includes energy used by I/O operations so that the overall systems power sharing ability can be achieved to some greater extent.
4. The Linux Nice value table which maps the tasks priority wise in collaboration with its Kernels load weight; is redefined with the precise allocation of power share.

The respective paper follows several related work. The Venn diagram for References used in this paper is as shown in Figure 1 below:

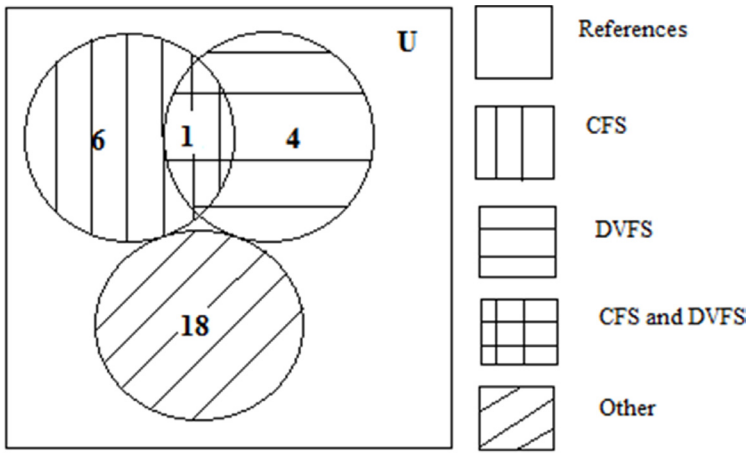

Figure 1: Venn diagram for References

3. PROPOSED SCHEME

Previously many research are done with Energy efficiency [19], Energy contingent, Energy fidelity but considering Operating System domain to deploy battery constraints are very scarce.

Devices are getting smaller in size with more amenities; hence it is crucial to maintaining a balance between battery capacity and different modern features of the portable devices, even more in handheld portable devices. The power management schemes were introduced to challenge battery limitation, and they have more impact on memory, CPU, Network Bandwidth and Performance.

The power management [20] scheme mainly focuses on two aspects Dynamic Power Management (DPM) and Dynamic Voltage and Frequency Scaling (DVFS). The DPM deals with executing the high workload at a maximum CPU speed while remaining workload at low power mode. The DVFS deals with executing processes at a low performance setting in terms of voltage and frequency.

The main motive is to design a scheduler driven DVFS scheme. To achieve this, already existing DVFS techniques are loaded into Linux Kernel module as shown in Figure 2 This aspect helps to reduce extra power usage by setting lowest value for processors frequency and voltage.

DVFS have its own set of different governors. Governors have a more controlled way for changing the CPU frequency.

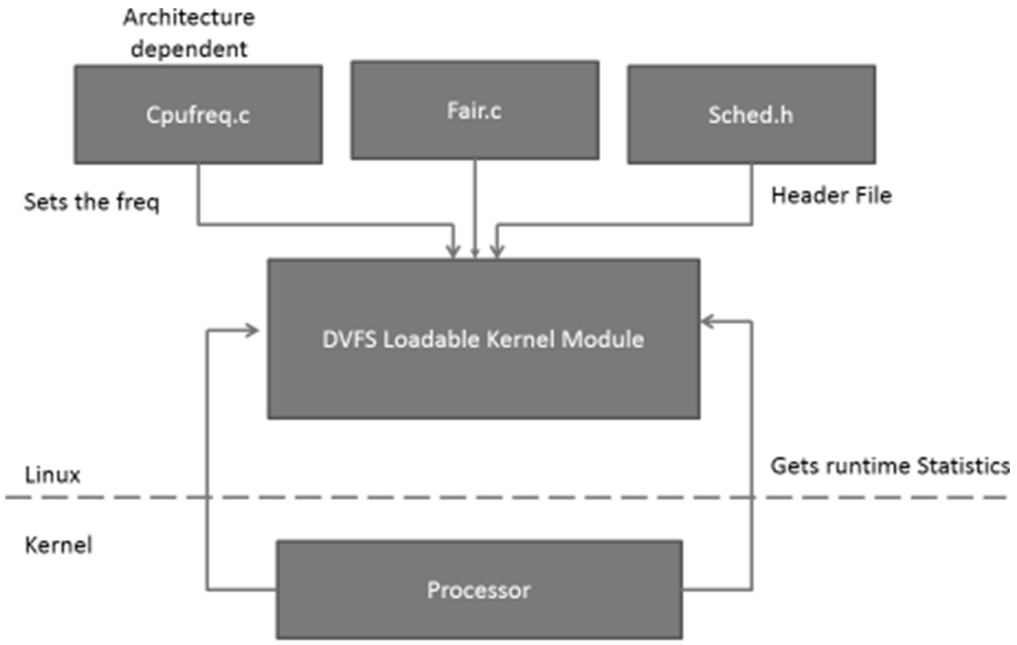

Figure 2: Proposed System Architecture

The proposed system focuses on estimation of response time [21][22] analysis by designing scheduler driven DVFS scheme. Response Time Analysis of Linux Kernel Completely Fair Scheduler for Data Intensive Task is carried out by analysis of frequency change by the help of DVFS properties invoking in Linux kernel with the help of Data Intensive Task. To optimize the user experience the Completion time or Response time of a Process is the main focus of the work. For the given frequency limits the utility of CPU Scheduling Algorithm will be explored.

Frequency analysis is done by the help of Data-intensive task. Data-intensive tasks are used to describe applications that are I/O bound or with a need to process large volumes of data. This kind of claims most of their processing time[23][24][25] to I/O and movement and manipulation of data. Data-intensive platforms use parallel computing approach combining multiple processors and disks to large computing clusters connected using high-speed communications switches and networks.

The response time analysis determines the schedulability of real-time systems on a fixed priority basis. The main objective of this study is to identify the points of interest with respect to frequency change within the Linux kernel for the response-time analysis.

4. CONCLUSION

Modern handheld devices have several advanced inbuilt features due to this, devices possess common battery limitation problem. Above paper reviewed various solutions to overcome this problem. One of the solutions is proposed in this paper with respect to Operating System track which is to invoke DVFS techniques in Linux scheduler CFS in collaboration with frequency change. This aspect directly works on kernel level approach.

Acknowledgment

The authors would like to thank all the Bharati Vidyapeeth College of Engineering staff members for their valuable inputs and support.

REFERENCES

[1] A. Gara *et. al*., "Overview of the Blue Gene/L system architecture, " in *IBM Journal of Research and Development*, Vol. 49, No. 2.3, pp. 195-212, March 2005.

[2] F. Lin and W. Ye, "Operating System Battle in the Ecosystem of Smartphone Industry, " *2009 International Symposium on Information Engineering and Electronic Commerce*, Ternopil, 2009, pp. 617-621.

[3] Peter Loscocco, N. S. A. "Integrating flexible support for security policies into the Linux operating system." *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference. Boston: USENIX Association*. 2001.

[4] Cuervo, Eduardo, et. al., "MAUI: making smartphones last longer with code offload." *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010.

[5] Jensen, E. Douglas, C. Douglas Locke, and Hideyuki Tokuda. "A Time-Driven Scheduling Model for Real-Time Operating Systems." *RTSS*. Vol. 85. 1985.

[6] Wong, Chee Siang, et. al., "Towards achieving fairness in the Linux scheduler." *ACM SIGOPS Operating Systems Review* 42.5 (2008): 34-43.

[7] Pabla, Chandandeep Singh. "Completely fair scheduler." *Linux Journal* 2009.184 (2009).

[8] Wang, Shen, et. al., "Fairness and interactivity of three CPU schedulers in Linux." *Embedded and Real-Time Computing Systems and Applications, 2009. RTCSA'09. 15th IEEE International Conference on*. IEEE, 2009.

[9] Kumar, Avinesh. "Multiprocessing with the completely fair scheduler." *IBM developerWorks* (2008).

[10] R. C. Garcia, J. M. Chung, S. W. Jo, T. Ha, and T. Kyong, "Response time performance estimation in smartphones applying dynamic voltage & frequency scaling and completely fair scheduler, " *Proc. Int. Symp. Consum. Electron. ISCE*, Vol. 2, No. 2, pp. 1–2, 2014.

[11] Le Sueur, Etienne, and Gernot Heiser. "Dynamic voltage and frequency scaling: The laws of diminishing returns." *Proceedings of the 2010 international conference on Power aware computing and systems*. 2010.

[12] Choi, Kihwan, Ramakrishna Soma, and Massoud Pedram. "Dynamic voltage and frequency scaling based on workload decomposition." *Proceedings of the 2004 international symposium on Low power electronics and design*. ACM, 2004.

[13] Dhiman, Gaurav, and Tajana Simunic Rosing. "Dynamic voltage frequency scaling for multi-tasking systems using online learning." *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*. IEEE, 2007.

[14] D.Brodowski, "CPUFreq Governors, " https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt, Nov. 2013.

[15] Pallipadi, Venkatesh, and Alexey Starikovskiy. "The ondemand governor." *Proceedings of the Linux Symposium*. Vol. 2. No. 00216. sn, 2006.

[16] C. S. Wong, I. K. T. Tan, R. D. Kumari, J. W. Lam, and W. Fun, "Fairness and interactive performance of O(1) and CFS Linux kernel schedulers, " *Proc. - Int. Symp. Inf. Technol. 2008, ITSim*, Vol. 3, No. 1, 2008.

[17] J. Wei, E. Juarez, M. J. Garrido, and F. Pescador, "Maximizing the user experience with energy-based fair sharing in battery limited mobile systems, " *IEEE Trans. Consum. Electron.*, Vol. 59, No. 3, pp. 690–698, 2013.

[18] J. Wei, R. Ren, E. Juarez, and F. Pescador, "A linux implementation of the energy-based fair queuing scheduling algorithm for battery-limited mobile systems, " *IEEE Trans. Consum. Electron.*, Vol. 60, No. 2, pp. 267–275, 2014.

[19] Paul, Kolin, and Tapas Kumar Kundu. "Android on mobile devices: An energy perspective." *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*. IEEE, 2010.

[20] Carroll, Aaron, and Gernot Heiser. "An Analysis of Power Consumption in a Smartphone." *USENIX annual technical conference*. Vol. 14. 2010.

[21] Barabanov, Michael. *A linux-based real-time operating system*. Diss. New Mexico Institute of Mining and Technology, 1997.

[22] Dilipkumar, Vora Shivani, M. Tech, and S. S. Dhotre. "Runtime CPU Scheduler Customization Framework for Real Time Operating System."

[23] Kabugade, Rohan R., S. S. Dhotre, and S. H. Patil. "A Modified O (1) Algorithm for Real Time Task in Operating System."

[24] Kabugade, Rohan R., S. S. Dhotre, and S. H. Patil. "A Study of Modified O (1) Algorithm for Real Time Task in Operating System." *Sinhgad Institute of Management and Computer Application NCI2TM* (2014).

[25] Karande, Poonam, S. P. Dhotre, and Suhas Patil. "Task management for heterogeneous multi-core scheduling." *Int. J. Comput. Sci. Inf. Technol* 5.1 (2014): 636-639.

[26] Silberschatz, Abraham, et. al., *Operating system concepts*. Vol. 4. Reading: Addison-wesley, 1998.

[27] Bovet, Daniel P., and Marco Cesati. *Understanding the Linux Kernel: from I/O ports to process management*. " O'Reilly Media, Inc.", 2005.

[28] R. Love, "Linux Kernel Development, " 2nd Edition, Noval Press, ISBN 0-672-32720-1, 2005.

[29] I.Molnar, "Modular Scheduler Core and Completely Fair Scheduler [CFS], " http://lwn.net/Articles/230501.