RESEARCH ARTICLE OPEN ACCESS

Response Time Analysis Using Linux Completely Fair Scheduler for Compute-Intensive Tasks

Pooja Tanaji Patil [1], Prof. Sunita Dhotre [2]
PG Scholar [1]
Department of Computer Engineering [1] & [2]
Bharati Vidyapeeth Deemed University College of engineering
Pune - India

ABSTRACT
Embedded systems are getting more numerous and complex day by day. Due to the need of portable devices to run the multiple application concurrently, power management is the biggest issue occurred in such systems. To maintain the performance of embedded system analysis of frequency change is an essential task. To reduce the energy consumed by systems the frequency of CPU has to minimize. Hence, the optimization of system can be achieved by estimating the response time of the completely fair scheduler (CFS) of Linux kernel. To achieve the better user experience the response time estimation is a huge threat.This paper deliberate the estimation of Response time by running Compute-intensive Task.

Keywords :— Embedded System, Power Management, Response Time, CFS, CPU Frequency

I. INTRODUCTION

Modern Linux operating system based portable devices such as Android, Apple's iPod, iPhone, Smartphones, tablets, etc. are experiencing considerable growth in performance and functionality to meet the multiplicity of user need. For such portable devices, the CPU frequency and software complexity is increasing day by day which demands the high power. But, the battery capacity did not increase significantly. Therefore the user experience is greatly affected due to limited battery capacity which is an unstable factor. So, power management is the biggest challenge faced by today's battery-limited devices. To address this problem several traditional Power Management schemes have been developed which provides more battery lifetime by managing the energy[1]. Dynamic Power Management executes workload to completion at the maximum CPU Speed and allows the rest of the system to perform in low power mode. The Dynamic Voltage and Frequency Scaling (DVFS) assumes that highest energy saving is possible by executing process at the lowest performance setting [2]. The proposed research focuses on designing a Scheduler driven DVFS Scheme by estimating frequency change analysis for Compute Intensive Task to minimize the Response time.

II. LITERATURE SURVEY

In the paper [3] author C. S. Wong, R. D. Kumari , and J. W. Lam has compared the two Linux kernel scheduler such as O (1) and Completely fair scheduler (CFS) in terms of fair sharing policy and interactive performance. In Linux kernel 2.6 O(1) scheduler is used while in 2.6.23 uses the CFS. O(1) replaced by CFS . Design goals of CFS are to provide the fair amount of CPU among all runnable tasks without immolating their interactive performance. Both schedulers share some characteristics in terms of fairness and interactive performance. Author has measured these design goals by using benchmarks that measure the system performance in terms of throughput. The results from the test conclude that the CFS is fairer than O(1) in the case of CPU bandwidth distribution and interactive performance.

In paper[2] author J. Wei, R.Ren, Juarez, F. Pescador provides the Energy based Fair Queuing scheduling algorithm(EFQ) which consume the energy on many devices. EFQ algorithm can achieve proportional sharing of power by consuming power on both CPU and I/O tasks based on their energy consumptions. This algorithm can achieve the power management scheme in battery limited mobile systems by providing proportional power sharing and efficient time-constrain compliance.EFQ algorithm achieves an energy-centric power management. Author focuses on energy-centric scheduling algorithm. Author also proposed that the EFQ can protect the sharing the power of specific application which is impossible for CFS. Author first improves the implementation of EFQ by using Pthread-based Test bench. Benchmark task are programmed into three type, real-time, interactive and batch also the performance of these tasks are retrieved under EFQ scheduling algorithm. Second, the power

consumption of each task is measured based on hardware metering system and based on obtained energy values are given as input to the Pthread-based test bench. Finally, the ability of EFQ of providing proportional power sharing is verified by calculating energy consumption caused by both CPU and I/O operations as total energy consumption of system. So, the CFS is extended by adding the new scheduling policy SCHED_EFQ. Four variables are included to the structure such as sched_entity and struct cfs_rq and initial weight, reserved share, energy packet size and warp parameters are added. Then the Linux nice values are calculated from 40 to 100. The nice values ranges from [-20 to 19] which is further modified to [-50 to 40].

In the paper [4] author has estimated the response time performance for smartphones. This response time estimation scheme is proposed by applying Dynamic Voltage and Frequency Scaling (DVFS) at CPU and Completely fair scheduler at Linux kernel.DVFS which controls scheduling reduces the power consumptions in smartphones through adaption of CPU core frequency level and system voltage. The change in CPU frequency ultimately changes the Response Time.In proposed Response Time Estimation Scheme there are two unit application architecture, first is Time measurement unit (TMU) which measures the response time of the second unit that is Instantaneous event unit (IEU). TMU launches the IEU and performs data parsing. IEU activate an update progress in separate threads.The effectiveness of proposed scheme is demonstrated by capturing various the changes in frequency levels based on executing various background applications for Smartphone.

The research paper [1] focuses on maximizing user experience in battery limited embedded system by using Energy-fair queuing which is class of energy-aware scheduling algorithm. In order to achieve the user-specified battery lifetime for embedded system, author proposed the energy instead of CPU should be managed. Author merges the traditional energy-efficient algorithm with EFQ to more maximize the user experience. EFQ algorithm manages the energy by scheduling each task based on their consumption of energy. This energy consumption controls the power of each task to avoid the energy starvation. For proportional time sharing relationship between CPU occupation time and energy consumption is considered. The maximization the user experience during its complete lifetime is simplified to one epoch. To achieve on epoch , first the applications which are preferred by user should executed with user-desired performance during whole epoch; after that, at the end of one epoch the remaining energy should be minimized to confirm that the rest of the task's performance is maximized. Based on this concept author has proposed the EFQ algorithm and it is tested on Linsched-based testbench which is an open-source Linux scheduler simulator.To frame the testbench author has modified the Linux Fair.c file and Linsched API to support new scheduling policies such as reserved share, Weight values and warp parameter. The new scheduling policy SCHED_EFQ is defined in sched.h file of Linux kernel and after that CFS code in Fair.c file is modified to implement the EFQ scheduling policy. Three types of Task are considered such as Real-time, Interactive and batch. Based on Setting of some parameters the maximum long-term power share and worst-case power share has computed. The other energy Efficient scheme such as DVFS in combination with the application self-adaption can be used to consume power and maximizing user experience.

III. RELATED WORK

The above observations and literature studies[3-7] indicate that CFS is not connected with the frequency scaling scheme of the CPU. As there is a possibility to enhance the response time by changing the frequency, CFS can be linked to the Dynamic Voltage and Frequency Scaling (DVFS) Algorithm. This leads to the need of design of a DVFS Scheme with an added scheduler governor wherein the responsive time of Compute-Intensive Task will be estimated to enhance the user experience. Scheduler-driven Frequency scaling scheme desires to exploits both the global information and per-task in the scheduler to improve the frequency selection scheme and achieves better responsiveness or performance and lesser energy consumption[5] so, to obtain the efficient performance from the users perspective, the proposed work utilizing the multi-threaded program executing in a multi-processor environment. The multithreaded program is implementing by compute-intensive task where the only CPU is utilized.

A. Completely Fair scheduler (CFS)

The latest Linux kernel scheduler is Completely Fair scheduler (CFS)[7][12] which was introduced in Linux Kernel 2.6.23 and extended in 2.6.24. CFS is “Desktop” process scheduler which was implemented by Ingo Molnar. Its core design can be summed up in single sentence: “CFS basically models an 'ideal, precise multitasking CPU' on real hardware [9][14].” It is impossible to get the ideal CPU in reality, but the CFS tries to imitate such ideal processor in system [16]. For scheduling the process CFS uses the process priority and timeslice [11][17]. timeslice is defined as the total amount of time taken by process to run and the process which is having the large timeslice is considered as higher priority process. The nice value given to each process according to user’s perspective determines the priority of process. The proportion of the time that any processor receives is determined by the

difference between the nice values of runnable process and the nice value of process itself. To decide the balance among multiple tasks CFS inaugurated the concept of "virtual runtime (vruntime)" [13]. Virtual runtime elucidate as the total amount of time provided to given task. The task which is having small virtual time means it has higher priority and will schedule first. The virtual runtime can be considered as a weighted time slice, which is represented by following equation- [11][12]

$$virtualruntime += \frac{(delta\_exec)\,(\text{default weight of process})}{se(load.weight)} \quad (1)$$

From the equation (1) of virtualruntime , delta_exec is the total amount of execution time of task, default weight of process means the unit value of weight and load.weight is weight of task/entity[19]. The weight of runnable processes is decided by their priority.

This scheduler also maintains the fairness for those processes which are waiting for I/O events to occur. Instead of maintaining these processes in run queue, the Completely Fair Scheduler maintains the time order Red-Black tree (RBTree) in a view to decide the task to schedule next on CPU.

B. DVFS(Dynamic Voltage and Frequency scaling)

The DVFS uses a disconnected set of governors namely Performance, Powersave, Interactive, Conservative and Ondemand[15]. Many CPU Frequency Scaling Governors exist which allows the drives to set the target frequency. Dynamic frequency Scaling[16] mechanism is applied for using the CPU efficiently.

C. Compute-Intensive Task-

Compute-Intensive is any task or application of computer which needs a lot of CPU/computation. These tasks are spends more time in executing the codes so also known as CPU bound processes in the operating system (OS). Linux scheduling policies attempt to achieve two goals such as fast response time and high throughput. So, in order to evaluate the performance measurements of scheduler the Compute-intensive tasks are implemented.

IV. PROPOSED SYSTEM

This research works on designing the scheduler-driven frequency scaling scheme to optimize the user experience from the perspective of Operating System. The analysis of change in frequency will be carried out by running Compute-Intensive Task which utilizes the system performance. Considering that the scheduler in the kernel plays a vital role in today's multi-core operating systems for estimating the performance. The proposed system aims to obtain the connectivity among the scheduler and DVFS scheme in order to optimize the Response time of the process and provide the better user experience.

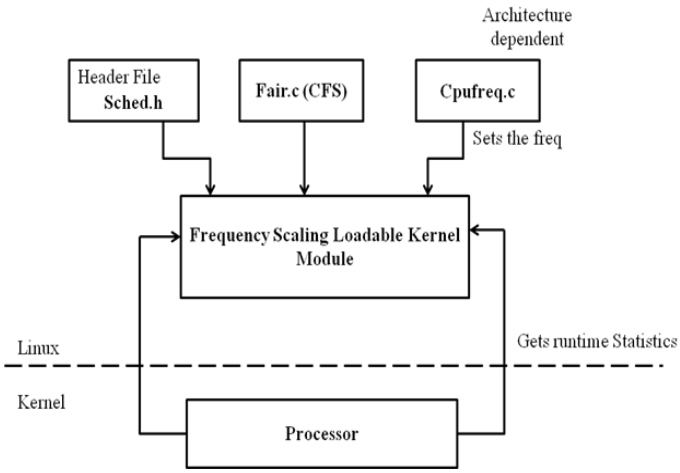

Fig. 1 System Level Implementation

As shown in fig. 1 , proposed research which aims to design the CFS enabled Frequency Scaling scheme. The Existing DVFS algorithm will be loaded in the kernel module along with the existing governors. The modification will be done in CFS header file sched.c. Then the Compute-Intensive task will be executed by setting different governors for analysis of change in CPU frequency and to estimate the Response time.

VI. CONCLUSION

Completely Fair scheduler has disconnected design from frequency scaling algorithm, so CFS could not controls the CPU frequency. Proposed research work will achieve the connection among CFS and Dynamic Frequency scaling scheme. Our work will focus on optimizing the user experience by analyzing the Response time for scheduler-driven frequency scaling scheme with the help of Compute-intensive Task and will compare the results with existing frequency scaling algorithm.

ACKNOWLEDGMENT

The proposed paper on "Response Time Analysis Using Linux Completely Fair Scheduler for Compute-Intensive Tasks" has been prepared by Pooja Tanaji Patil under the guidance of Prof. Sunita Dhotre.

Author would like to thank whole department, friends and parents for the valuable support and confidence in me.

REFERENCES

[1] 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.

[2] 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.

[3] C. S. Wong, R. D. Kumari, and J. W. Lam, "Fairness and Interactive Performance of O(1) and CFS Linux Kernel Schedulers," no. 1, 2008.

[4] 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.

[5] C. S. Wong, I. Tan, and R. Deena, "Towards Achieving Fairness in the Linux Scheduler," pp. 34–43.

[6] S. Wang, "Fairness and Interactivity of Three CPU Schedulers in Linux," pp. 2–7, 2009.

[7] S. M. Mostafa, H. Amano, and S. Kusakabe, "FAIRNESS AND HIGH PERFORMANCE FOR TASKS IN GENERAL PURPOSE MULTICORE SYSTEMS," vol. 29, no. December, pp. 74–86, 2016.

[8] J. Lozi, J. Funston, F. Gaud, V. Qu, and A. Fedorova, "The Linux Scheduler : a Decade of Wasted Cores."

[9] "Completely Fair Scheduler _ Linux Journal." http://www.linuxjournal.com/magazine/completely-fair-scheduler .

[10] "Tuning the Task Scheduler _ System Analysis and Tuning Guide _ openSUSE Leap 42." https://doc.opensuse.org/documentation/leap/tuning/html/book.sle.tuning/cha.tuning.taskscheduler.html.

[11] G. Cheng, "A Comparison of Two Linux Schedulers," Master thesis, pp. 1–89, 2012.

[12] Yigui Luo, Bolin Wu, "A Comparison on Interactivity of Three Linux Schedulers in Embedded System", Communications, Computers and Signal Processing(PacRim), 2011 IEEE Pacific Rim Conference, pp. 494-498, August 2011.

[13] Wei-feng MA, WANG Jia-hai, " Analysis of the Linux 2.6 Kernel Scheduler" 2010 IEEE International conference on computer Design and Applications, pp.71-74, 2010

[14] Prajakta Pawar, SS Dhotre, Suhas Patil, "CFS for Addressing CPU Resources in Multi-Core Processors with AA Tree", International Journal of Computer Science and Information Technologies, Vol. 5 (1), 913-917, 2014

[15] R. Ge, R. Vogt, J. Majumder, and A. Alam, "Effects of Dynamic Voltage and Frequency Scaling on a K20 GPU."

[16] "Power Management & DVFS." http://www.arteris.com/power-management-dvfs .

[17] P. T. Patil, "A Study on Evolution of Storage Infrastructure," vol. 6, no. 7, pp. 501–506, 2016.

[18] Poonam Karande, SS Dhotre, Suhas Patil, "Illustration of Task Scheduling in Heterogeneous Quad-Core Processors", International Journal of Engineering and Technology Research, Vol 03, Issue 08, Pages:1389-1393, May 2014

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

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

[21] P. Malviya, "A Study Paper on Storage Area Network Problem-Solving Issues," vol. 4, no. 4, pp. 151–156, 2016.

[22] Pawar, Supriya Haribhau. "A Study on Big Data Security and Data Storage Infrastructure." International Journal 6.7 (2016).

[22] A. Silberschatz, P.B. Galvin, G. Gagne, "Operating System Concepts," 7th Edition, John Wiley & Sons Inc.,2005

[23] Richard Petersen, "The Complete Reference" Linux, Second Edition, Tata McGraw Hill.

[24] Daniel P. Bovet & Marco Cesati". Understanding the Linux Kernel, OReilly October 2000