Characterization of Smartphone Governor Strategies

Sarbartha Banerjee(✉) and Lizy Kurian John

University of Texas at Austin, Austin, TX 78705, USA
{sarbartha,ljohn}@utexas.edu

Abstract. The voltage and frequency of the various components of a smartphone processor such as CPU cores, graphics, multimedia and display units can be independently controlled by their own dynamic voltage and frequency (DVFS) governors to fit the requirement of the workload. The dynamic change of the voltage and frequency performed by governors is targeted either towards achieving the optimal performance with the minimum energy consumption or choosing a mode which requires minimum supervision of workload and minimal change of DVFS modes (since changes in modes are accompanied by overheads of switching).

This paper explores the behaviour of different governors run on a wide variety of workloads and enlists the best strategy for different scenarios exemplifying the need for workload characterization. We also analyze the performance and power efficiency of workloads in a system having a common power source and study their behavior when multiple such blocks are operating together pushing the power source to its limit. Our results show that choosing the correct CPU governor alone is not sufficient but tuning the DVFS of different resources is necessary to achieve the best performance with minimum energy expenditure. We observe that the *powersave* governor does not always give the best energy efficiency. It was found to be sub-optimal for CPU intensive workloads due to increased execution time. Moreover, the *race-to-idle* strategy was found to be optimal for workloads in which one component is utilized for majority of the time. These results demonstrate the necessity for characterizing workloads and tuning the DVFS while distributing the power between the various components based on the workload's characteristics.

Keywords: SoC · Governor · Power budget · Race-to-idle · Pace-to-idle

1 Introduction

Getting desirable performance with optimum energy efficiency have become the major design criteria for modern smartphones. This is primarily because battery technology development has been much slower than processor development, with the form factor of the phones limiting the battery capacity and the stringent thermal limit of the device. To address this issue, all modern smartphones have multiple DVFS (Dynamic Voltage Frequency Scaling) modes to run different components in the most efficient mode. In typical DVFS, the frequency and the voltage of the processor is modified based on the component utilization. Tuning the frequency of the essential component not only saves

M. Aldinucci et al. (Eds.): Euro-Par 2018, LNCS 11014, pp. 120–134, 2018.
https://doi.org/10.1007/978-3-319-96983-1_9

power but also increases performance in certain scenarios. In smartphones, sometimes a single power source is shared among various components. There is a peak power limit of the power source in addition to thermal constraints. These constraints led to the development of new governor strategies which are not only focused on increasing performance but also tackling the workload in the most energy efficient way.

The availability of DVFS in different components and a high number of DVFS modes within a component makes the optimal choice very difficult. Moreover, providing the user with a satisfactory performance for prolonged period with high energy and thermal efficiency has become a new paradigm.

One simple heuristic for power management using DVFS is to run the job on the target system at the maximum possible frequency (maximum performance mode) and then throttle down to minimum or deep-sleep state as quickly as possible. This method is termed as *race-to-idle*. This method is simple, reduces latency and saves energy in certain use cases. The energy saving comes from the fact that the processing unit is active for the minimum amount of time and leakage power is saved in inactive modes. But, its validity and usefulness is yet to be conclusively established for smartphones as workloads tend to use different resources intermittently sometimes using multiple processing units at the same time. More complex methods can optimally switch the processor frequently to the optimum DVFS mode based on the workload performance requirement by polling the resource usage and trying to finish it in the most energy efficient manner. This is termed as *pace-to-idle*. But in such cases, some energy is wasted monitoring the workload continuously. Moreover, mapping a workload dynamically into heterogeneous clusters of multicore processors and various accelerators like the GPU cannot be done without efficient workload behavior characterization.

Furthermore, there are situations when the smartphone is running on low battery. Normally, the frequency of all the blocks are toned down to consume less energy. But the increasing leakage current raises the question if it really increases the energy efficiency when we need to run an application on the system at lower frequency?

Thus, understanding the workload behavior is essential while choosing the governor. At least if one can classify the workload and figure out the functional units needed, it will greatly help in choosing an appropriate governor for each resource. Also, most of the governors are designed for the CPU. But global decision of the various DVFS modes in an energy constrained system based on the workload improves the power efficiency and less temperature rise of the smartphone system-on-chip (SoC).

Our study encompasses the analysis of various categories of governors for different kind of workloads to explain the optimal strategy in a mobile platform. The race-to-idle strategy has been shown to be effective for servers where the quality of service and latency of the requests are important. But for mobile devices, an acceptable quality of service is desirable within the bounds of power limit of the source must be provided while respecting the thermal limits making it an optimization problem.

Some power-hungry governors are good for performance while some relaxed governors might be power saving. With the availability of multiple DVFS modes, finite DVFS switching time and workload detection, researchers are coming up with improved governors that predict the pattern of the workload and choose the appropriate DVFS point. The analysis shown in the paper is a start point for any governor designer to make reasonable decisions for a governor.

The rest of the paper is organized in the following format: Sect. 2 provides background about the DVFS modes, the governors and their characteristics. Section 3 elaborates on the experimental setup. Section 4 explains the workloads and benchmarks used. Section 5 shows results of our experiments. Section 6 explains the benchmark characteristics and their behavior with different governors. Section 7 provides our observations from the experiments conducted and conclusion in Sect. 8.

2 Background

The smartphone system on a chip (SoC) comprises of multi-core CPUs, a GPU and multimedia units running on a separate DVFS point while sharing the same current source. With the demand for new aggressive power saving techniques, designers have added more voltage-frequency (VF) points to individual units and added governors for independent control of different units. Power can be saved if one enables the desired unit at the appropriate frequency. But switching the DVFS modes consumes energy and has non-zero latency. Too much switching is also not desirable. In addition, every unit can also be separately put in the different idle power modes like clock gating, retention or deep sleep. All these low power modes have different wake up latency and leakage current consumed.

2.1 Governors

In this section, we will first give a brief overview of the types of CPU governors present in the Linux kernel of an android smartphone today and then go over some of the common governors and frequency scaling points of other units in the SoC.

Performance Governor. This governor is a constant frequency governor which keeps the system in highest possible voltage and frequency irrespective of the workload. This is highly power hungry and the core latches itself to maximum frequency. Worth noting is that this governor works best when a series of compute intensive job is run in the system. Moreover, it also keeps the bus to DDR at its peak frequency. It doesn't waste extra time and power in DVFS switching. But keeping the processor in this frequency can cause thermal throttling and unnecessarily running it near the peak current of the supply. But once the processor run queue is empty, it goes back to the sleep state. It is considered as a 'race' governor which finishes the job as quickly and goes to idle.

Interactive and Ondemand Governor. The ondemand governor [2] switches the system in highest possible voltage and frequency whenever a job is scheduled and immediately ramps down to lower frequency when the resource utilization fades. The interactive governor find the optimal frequency based on the load average of the system. If the load average is more than a pre-specified value, it switches to higher frequencies. Similarly, if the load average is low, the ondemand ramps down immediately while the interactive waits for a certain hysteresis time. This works well when we have a sequence of compute intensive jobs interspersed with long delays. The immediate return to low frequency ensures that it spends minimum time in the highest DVFS mode. However, if the idle time between jobs is very low, this governor hops

between frequencies repeatedly. The Interactive governor adds a hysteresis timer on top of the ondemand governor to filter some of the switching. This governor can be considered as a pace' governor which will adapt the frequency based on the workload requirement.

Powersave Governor. Powersave governor is designed to save energy by running the CPU at the lowest possible operating frequency. This gives slow response but reduces average power in many situations and is often used when battery is low or during thermal throttling. It also gives good performance when the application is using another component of the SoC like the GPU with minimal CPU utilization but might falter in certain cases as the overall energy consumption may exceed others due to significantly higher runtime. It also fails to attain desirable QoS and provide poor user experience.

GPU Governors. Most of the chips have GPU as a proprietary unit, so the governors supported are specific to the hardware used in the experiment. Since our test setup had a Qualcomm Snapdragon processor, we will list down a couple of GPU governors.

Most of the fancy governors are largely pacing governors whose performance lie between the performance and the powersave governors. *Msm-adreno-tz* is one such governor which works like the interactive governor and tunes based on the GPUbusy data stating GPU utilization. It also has *performance* and *powersave* governors which are like the CPU counterparts working of GPU frequencies.

The optimization of the GPU governors can improve energy efficiency of the overall system as it is a high-power resource. Thus, the above options do tell us that battery power saving is not only limited to the CPUs but in every units of the SoC. Similar changes can be done to the DDR frequency and multimedia components.

2.2 DVFS Points

Owing to the need to save power and to provide flexibility to choose the appropriate mode to perform a task, hardware designers provide several DVFS points for different resources. Our testing platform is a Dragonboard 410c [14] platform consisting of a Qualcomm Snapdragon 410 processor having Quad-core ARM A53 processor with all four cores running at the same voltage & frequency. The cores can be independently put into low power mode but they cannot be run at different frequency. This Snapdragon processor supports the following eight different frequency points each having a different voltage.

- 1209 MHz
- 1152 MHz
- 1094 MHz
- 998 MHz
- 800 MHz
- 533 MHz
- 400 MHz
- 200 MHz

Apart from that the DDR memory also has different frequencies of 533, 400 or 200 MHz. Either it can be scaled independently or in tandem with the CPU frequency. Similarly, the GPU has its own independent DVFS modes but shares the same power rail as the CPU and others.

Choosing wrong DVFS points for individual components may prevent providing enough budget to the crucial component adversely affect performance. For instance, if there are a lot of I/O operation or if a multimedia application is running, keeping the CPU in performance mode will allocate a larger power budget from the current source to the CPU and the multimedia unit will simply perform poorer due to lack of power budget for this unit. In our test setup, we have observed a similar scenario by running Geekbench 3 by keeping the CPU at different frequencies. It is observed that the memory intensive tests that perform occasional computation perform poorly when the CPU is in its highest frequency as simple computations can be performed in lower frequency with same latency but without reaching the power limit of the device. Moreover, there can be thermal throttling forcing all units to tone down its activity. It is unique in smartphones as a lot of blocks share a single power source. Not only does it show poor performance but also consumes higher leakage and clock tree power when the processor fails to shut down when it is not required. Thus, choosing the correct DVFS point for each resource is essential for efficient power budget distribution for maximizing performance of the highest used resource.

2.3 Quality of Service

A governor should not only work towards energy efficiency but also provide user acceptable performance. The performance need not be the best but needs to comply to some standard. Researchers have collected user surveys to determine the level of user satisfaction for mobile devices. We compiled QoS data from prior research [7–10] and enlist them in the result section. Furthermore, we specify that the benchmark scores should be within 95% of the maximum possible score attained by the device.

3 Experimental Setup

We used Dragonboard 410c [14] for the analysis of energy consumption across various workloads and benchmarks. It contains a Qualcomm Snapdragon 410 consisting of Quad-core ARM Cortex A53 processors running Android 5.1.1. There are shunt registers provided on board [15] to check the incoming current to the processor. The reason of choice for this processor is its prevalence in value-tier market and the fact that it has a shared power source. Below are some of the specifications of this processor are listed in Table 1.

Table 1. Snapdragon 410c specification

CPU	4 x ARM Cortex A53 1.2 GHz
CPU arch	64 bit ARM V8 architecture
GPU	Qualcomm Adreno 306 400 MHz
DSP	Qualcomm Hexagon DSP
Memory	1 GB LPDDR3 533 MHz

The points across the shunt resistor (R77) on the board are tapped and a INA219 current sensor is connected to measure the current. The output of the current sensor is sampled using a microcontroller to get the data. A block diagram of the setup is shown in Fig. 1.

Some of the parameters of the hardware are tuned during the study of governor behavior. It includes *CPU governor, Governor tuning, DDR frequency, GPU frequency, Thermal throttler, Hotplugging setting.* All the parameters are tuned for every run and then the workload is run in the system. The android debug bridge [14] (ADB) is used for the measurements and various comparisons are performed.

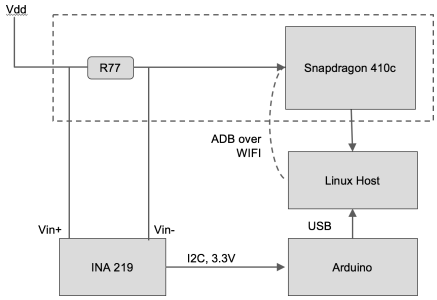

Fig. 1. Block diagram of the experimental setup

4 Applications and Benchmarks

A brief analysis of some of the experiments performed are described in this section. The results in term of scores and the normalized energy consumed in reported in Table 2. Linaro workload Automation suite [16] is used to run a host of applications explained in the Table 2 and standard benchmarks which includes the following:

- Antutu
- Geekbench
- BBench
- Nenamark
- Ebizzy
- Dhrystone
- Linpack
- Memcpy

Table 2. A description of the applications

Applaunch	Launches either the calculator, browser or google Maps application when no other application is running in the system
Multi_applaunch	Launches calculator, browser and maps application in a sequence on top of one another
Video	Playing a 720p video file in the native android video player
Audio	Plays an audio file in the native android audio player
Maps	Open google maps and perform a navigation task
Adobereader	Scrolls, zooms and searches a word after opening a pdf file
Facebook	Performs a series of tasks after logging in a facebook account including scrolling through the wall, like a friend's photo, post a status and comment on an existing post
Iozone	Performs a series of IO performance tasks

5 Results

First, we provide a distinction between *race-to-idle* and *pace-to-idle* governor strategies. The *performance* and *powersave* governors keep the CPU frequency at the max and the min operating point. This accounts for the least governor software overhead and no time wasted on voltage and frequency modulation. However, they cannot adapt to phase changes. Performance governor is a *race-to-idle* governor. On the other hand, the *pace-to-idle* strategies like the *interactive* and *ondemand* frequently changes DVFS points based on CPU utilization. These works better in the application workloads which interleaves different resources. A view of the number of switching is shown in Table 3. Antutu shows frequency toggles in performance mode because of thermal throttling pointing out the drawback of *race-to-idle* strategy in mobile devices. Interactive filters out some modulation using hysteresis as compared to ondemand governor and performs better in terms of performance and energy efficiency in most applications. Moreover, the performance must meet minimum standards which we compile from prior research and is enlisted in Table 4.

Table 3. DVFS mode switching of different governors

Benchmarks	Performance governor	Interactive governor	Ondemand governor	Powersave governor
Antutu	18	842	3809	0
Applaunch	0	1897	7463	0
Audio	0	43	112	0
Dhrystone	0	8	12	0
Geekbench	0	229	887	0
Homescreen	0	44	51	0
Linpack	0	31	83	0
Memcpy	0	10	12	0
Nenamark	0	1178	8383	0

Table 4. Quality of service of different user actions

Behavior	Quality	Application
Webpage load time	4 s	BBench, firefox
Online video loading time	2–10 s	Stream, Youtube
Facebook comment post	3 s	Facebook
Interactive tasks	100 ms	Applaunch, Adobereader
Video playback	30 fps–60 fps	Video/Game rendering
PDF rendering	1–10 s	Adobereader

Some of these are benchmarks like the Antutu, Geekbench, Dhrystone and Nenamark whose scores are directly reported in Table 5 when run with different governors. Antutu and Dhrystone primarily stresses the CPU. The interactive governor gives similar performance as performance governor but consumes more power because

it unnecessarily toggles the frequency. Nenamark is a graphics benchmark running OpenGL-ES 2.0. The powersave governor gives best frame rate as the GPU governor is tweaked to performance and bus frequency is changed while the CPU is in powersave mode. This shows that changing the DVFS modes for the critical component not only increases performance but also consumes less power. Applaunch of both single and multiple application works best when the CPU is in performance mode as the QoS is for the quickest application load time. Moreover, there is not much difference in response time whenever we are launching light application like the calculator. The effect is more pronounced when heavier or multiple applications are launched. Table 5 shows the performance of different applications and benchmarks. The values are marked in green for the acceptable QoS and red for unacceptable ones.

Table 5. Performance comparison among different CPU governors and green ones have acceptable QoS.

Workload	metric	Governors			
		performance	*interactive*	*ondemand*	*powersave*
Antutu	Score	19246	19038	19027	12201
Dhrystone	DMIPS	4053	4053	4052	2679
Applaunch calculator	Launch time (s)	0.71	0.74	0.79	0.89
Applaunch Browser	Launch time (s)	1.007	1.02	1.07	1.46
BBench	Runtime(s)	190.9	184.17	187.2	246.24
Adobereader	Runtime(s)	77.14	79.14	79.95	103.61
ebizzy	Total records/sec	2017	2011	1757	472
Nenamark	Frames per second	35.6	35.2	34.9	37.4
Memcpy	Bandwidth (in MB/s)	3114	3060	2970	588

6 Benchmark and Application Classification

A host of benchmarks and user workloads were run with different governors. The workloads are classified in this section into the following categories:

6.1 CPU Intensive Workloads

These are the workloads that are compute intensive and works best when the processors are at peak frequency. Race-to-idle scheme gives better performance and is often energy efficient as well by reducing the number of DVFS switches and also keeping the SoC active for the minimum amount of time. The pace-to-idle governors on the other hand, suffer from too many unnecessary DVFS modulations. Interactive and ondemand governor works good if the workload is continuously CPU demanding and behaves like performance in Dhrystone [1]. This can be viewed in the minimal number of DVFS modulations in Table 3. Figure 2 shows the average current of CPU intensive

benchmarks. Applaunch of calculator (simple application) and firefox works fastest in performance governor. Antutu benchmark shows that the performance has the highest power efficiency while meeting QoS. Powersave consumes the least power as its frequency is clipped to the lowest operating mode but it gives drastically poor performance. Thus, if we can categorize a phase of a workload as compute intensive, we can move to performance mode until the phase completes to get the maximum performance and minimal DVFS switching overhead.

6.2 Intermittent CPU Workloads with I/O Operation

Some of the workload we tested like the BBench which loads saved webpages by I/O operation and scrolls through the webpages which is CPU intensive works best in pace-to-idle type of governors. Since the CPU is only used intermittently, interactive governor is the most efficient as it lowers the frequency of the CPU while doing I/O operation. The lowering of CPU frequency also provides more power budget to the I/O unit and it can provide better response. Figure 3 shows that BBench is a heavy benchmark and consumes good amount of current throughout its execution. Facebook, Adobereader also are I/O intensive and saves CPU power during user interactions. Geekbench also has a lot of memory operations where interactive aces out. Performance governor scores better in the CPU intensive workloads of the Geekbench suite.

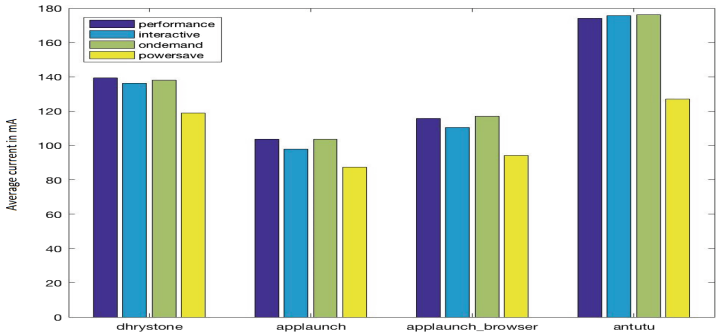

Fig. 2. Power comparison of CPU intensive benchmarks of different governors.

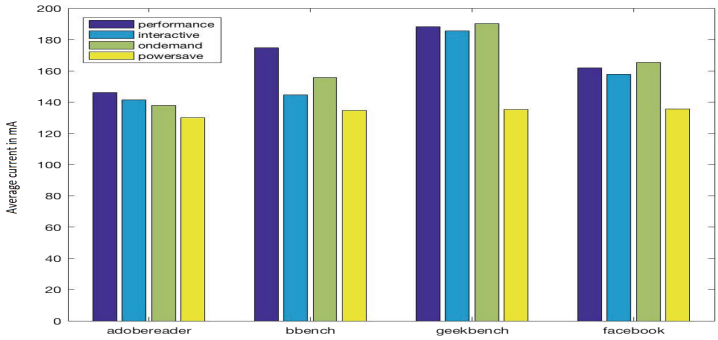

Fig. 3. Power comparison of user interactive applications of different governors.

We went in deeper into the Geekbench 3 workload and compared the *race-to-idle (performance)* and *pace-to-idle(interactive)* strategies closely. The performance governor aced in the compute intensive integer and floating point benchmarks. But due to constrained power budget, it performs poorly in most of the memory benchmarks. Since it clocked the CPU continuously at the highest frequency, it failed to provide enough power to the memory bus degrading performance. On the other hand, the interactive governor clocked the CPU at minimum operating point for simple operations and redirected the entire power to the memory bus giving better bandwidth as shown in Fig. 4(a).

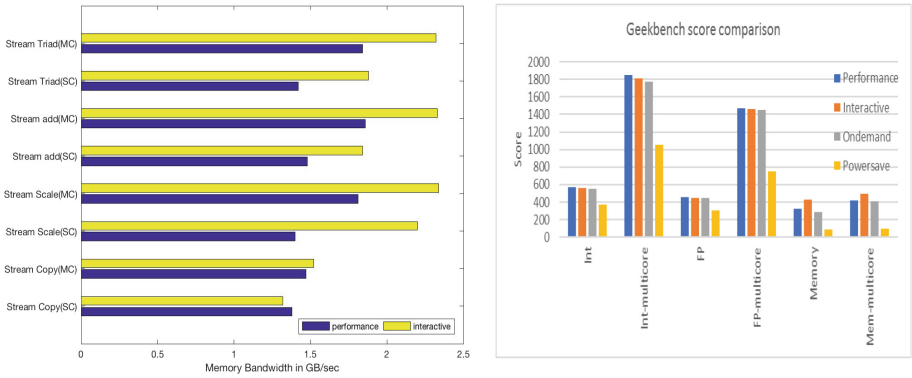

Fig. 4. (a) Geekbench 3 memory perf comparison between performance and interactive governor (b) Geekbench 3 score comparison of different types of workloads.

6.3 Application Requiring Other Blocks in the SoC

There are other applications like playing a video which requires the multimedia unit to be active. The CPU can stay in the powersave mode while providing power budget to the DDR and the multimedia unit to perform. Moreover, playing games require the GPU to be in higher performance mode to render better user experience.

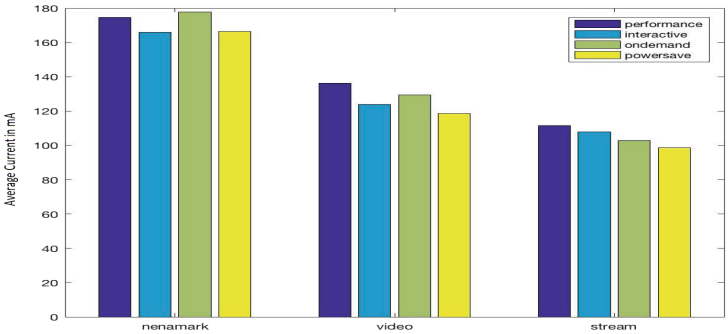

Fig. 5. Power comparison of governors running applications using other resource

In Fig. 5, the GPU is in performance mode in the Nenamark benchmark and the powersave not only consumes least power but also provides the highest fps as more power budget is allocated to the accelerator. Thus, application-wise characterization is useful to provide better power efficiency. Running in powersave ensures that the device remains cooler for a longer period which is the normal usage pattern of video and audio playback applications.

7 Observations

After running various types of workloads on all the different kinds of governors, it is seen that choosing the correct governor in a battery-operated system-on-a-chip is a multi-variable problem where one should consider the balance of activities in the various units. Governors should also minimize thermal throttling hardware to attain best efficiency and performance in addition to better chip life.

As mentioned in the introduction that a race-to-idle scheme works well for servers because more importance is given on the performance, this scheme is thought to be a poor fit for battery-operated devices. But our observation states that in compute intensive workloads, the race-to-idle scheme performs better not only in performance but also it gives better energy efficiency in some cases. In multicore devices, normally there is provision of switching off each core into several idle low power states. So, if a governor finishes the pending work in the minimum time and goes to idle, it will save operating power. This strategy will work even better with technology shrinking as the leakage current becomes comparable with the dynamic current. With more application-specific units are put in the SoC, having a global governor controlling the DVFS modes of every component based on workload characterization will be the desired solution as we observe by classifying the workload and showing that CPU powersave coupled with resource *race-to-idle* works better in scenarios which use accelerators. The race-to-idle scheme also makes sharing of power source easier as the units are active for the minimum amount of time. Last but not the least is the fact that race-to-idle schemes give better performance most of the time.

The pace-to-idle strategy also performs well in multiple scenarios where multiple resources are used together or in a sequential manner. For instance, the BBench workload loads a set of heavy webpages from the memory making it a memory intensive workload followed by the execution of contents in the webpages, which is compute-intensive. In these scenarios, the pace-to-idle strategies work best as all the units like the memory-bus and the CPUs are appropriately scaled whenever it is needed. It also performs better in applications like Facebook which requires user interaction where the CPU frequency can be opportunistically modulated to reduce power and temperature of the device. It also helps in thermal distribution as the cores get heated up when it is constantly at higher frequency reducing reliability and performance by engaging the thermal throttler. We tested the Adobereader applications in which we opened a document, scrolled through it and adjusted the zooming. Since there were enough idle times between these operations, interactive was the most efficient. The responsiveness of the interactive governor was like the performance

governor but it performed poorly while searching a word in the file ending up consuming more power. But mere scrolling through the text would have been more efficient in the interactive governor.

Moreover, for video playback or a GPU-intensive game, the multimedia or GPU is used and changing the CPU governor doesn't make any difference in performance. But changing the governor of the corresponding block improves both performance and energy efficiency. Based on the above trends, we conclude that characterization of workload would help design a high performing energy efficient governor.

8 Related Work

There has been considerable amount of research performed to enhance the native interactive(default) governor of the system. The related works are grouped into the following categories:

8.1 Race-to-Idle vs Pace-to-Idle Schemes

Some works suggested pace-to-idle strategy is the better strategy [3] due to the intermittent CPU usage pattern of the workloads and waiting for I/O interrupts. With the constrained system power/thermal budget, it is not feasible to make all resources available simultaneously. But on the other hand, transistors are shrinking in size and leakage is comparable to the dynamic current. The reduction in resource active time by race-to-idle schemes are making it attractive. Moreover, Race-to-idle schemes give better performance. Coupled with all these benefits, the race-to-idle is becoming popular. Albers and Antoniadis [4] have proposed that the race-to-idle strategies provide better energy efficiency provided the system has multi-level and deep sleep states which is common in smartphones. It causes the minimum DVFS transitions causing less halts and power wastage. While Hoffman [3] claims that pace-to-idle betters than race-to-idle in smartphones due to the intermittent use of a specific resource diluting the effectiveness and energy efficiency of the resource. But workload characterization can help improve the effectiveness of the resource by deploying race-to-idle strategy for the required resource making it fully available when necessary thereby improving energy efficiency.

8.2 Governor Design Based on Runtime Phase Behavior and QoS Deadline

Isci et al. and others [5, 22] has used runtime phase behavior to perform dynamic DVFS management of a device. The phase behavior was identified from the branch predictor. On the other hand, some of the DVFS governors were designed keeping in mind the idea of meeting a quality-of-service(QoS) deadline while running the processor at the optimum frequency to achieve the highest energy efficiency [19, 20]. These policies explore the search space to figure out the most energy efficient DVFS mode. Though these works [6, 11] are promising but their applicability is restricted to limited applications like web browsing and video playback. Moreover, it doesn't

consider system-wide power budget. A single power source can be shared among multiple resources like multi-core CPUs, GPUs and other accelerators. Redirecting the power to the most useful resource is important when the current consumed is near the limit of the source.

8.3 Power Sharing Among Different Resources

Paul et al. and others [18, 21] has evaluated the need of cooperative boosting between CPUs and GPUs in a AMD APU processor. This is critical in smartphones when multiple resources are sharing a power source or when the device is thermally limited. However, this work is focused on desktop CPUs. A smartphone CPU like the one used in this work has more resources sharing a current source and the QoS metrics are quite different. We evaluated the different smartphone governors on similar lines with higher granularity in the type of resource.

8.4 Reducing DVFS Switch Time

Another line of radically different effort is put to reduce the DVFS switch time. The pace-to-idle can be made even more aggressive in the switching time is reduced. New PLLs and voltage regulators have better response time to quickly switch the modes with minimum current spikes which improves the overall energy efficiency. Several researchers have proposed elegant methods [12, 13] to reduce the switch time. But still the DVFS switching time is of the order of several micro-seconds as it involves changing the voltage. Workload characterization and identification of phases based on usage pattern can reduce the number of DVFS mode changes and will increase efficiency but the algorithms can be made more aggressive when the DVFS switching time reduces.

9 Conclusion

In this paper, we studied various governor strategies and their impact on performance and energy consumption while running various workloads for a smartphone. We conclude that a good governor must wisely choose the DVFS mode of not only the CPU, but also the various non-CPU components when the workload demands varied utilization of multiple blocks sharing a current source. System wide governors tuning the DVFS modes of different units of SoC will provide efficient utilization of the available current. Since, smartphone applications mostly use a specific component of the SoC, characterization of workloads to boost the frequency of the corresponding component to the required level gives better performance with increased energy efficiency. Analyzing phase behavior and usage pattern of the program can further help in selection of the optimum DVFS mode. It was observed that turning on the powersave mode does not necessarily save battery in many scenarios. The powersave governor led to increased energy consumption for CPU intensive workloads because of higher run time causing more leakage energy consumption. Hence, characterization of a workload and wise current distribution to the critical components is imperative in designing a governor giving it desirable performance but also yields high energy and thermal efficiency.

Acknowledgement. This work was partially supported by National Science Foundation (NSF) under grant numbers 1725743 and 1745813. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF or other sponsors.

References

1. Weicker, R.P.: Dhrystone: a synthetic systems programming benchmark. Commun. ACM **27** (10), 1013–1030 (1984)
2. Pallipadi, V., Starikovskiy, A.: The ondemand governor. In: Proceedings of the Linux Symposium, vol. 2, pp. 215–230 (2006)
3. Hoffmann, H.: Racing and pacing to idle: an evaluation of heuristics for energy-aware resource allocation. In: Proceedings of the Workshop on Power-Aware Computing and Systems, p. 13. ACM (2013)
4. Albers, S., Antoniadis, A.: Race to idle: new algorithms for speed scaling with a sleep state. ACM Trans. Algorithms (TALG) **10**(2), 9 (2014)
5. Isci, C., Contreras, G., Martonosi, M.: Live, runtime phase monitoring and prediction on real systems with application to dynamic power management. In: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society (2006)
6. Rao, K., Wang, J., Yalamanchili, S., Wardi, Y., Handong, Y.: Application-specific performance-aware energy optimization on android mobile devices. In: 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 169–180 (2017)
7. Halpern, M., Zhu, Y., Reddi, V.J.: Mobile CPU's rise to power: quantifying the impact of generational mobile cpu design trends on performance, energy, and user satisfaction. In: 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE (2016)
8. Shneiderman, B.: Designing the User Interface. Addison-Wesley, Boston (1992)
9. Zhu, Y., Halpern, M., Reddi, V.J.: Event-based scheduling for energy-efficient QoS (eQoS) in mobile web applications. In: 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA). IEEE (2015)
10. https://blog.kissmetrics.com/speed-is-a-killer/
11. Zhu, Y., Reddi, V.J.: Optimizing general-purpose cpus for energy-efficient mobile web computing. ACM Trans. Comput. Syst. **35**, 1 (2017)
12. Eyerman, S., Eeckhout, L.: Fine-grained DVFS using on-chip regulators. ACM Trans. Archit. Code Optim. (TACO) **8**(1), 1 (2011)
13. Kim, W., Gupta, M.S., Wei, G.Y., Brooks, D.: System level analysis of fast, per-core DVFS using on-chip switching regulators. In: 2008 IEEE 14th International Symposium on High Performance Computer Architecture, HPCA 2008, pp. 123–134. IEEE (2008)
14. Dragonboard 410c. https://developer.qualcomm.com/hardware/dragonboard-410c
15. Measuring power consumption for Dragonboard 410c. https://developer.qualcomm.com/download/db410c/power-measurement-appnote.pdf
16. Linaro workload automation. https://media.readthedocs.org/pdf/workload-automation/latest/workload-automation.pdf
17. Android debug bridge. https://developer.android.com/studio/command-line/adb.html
18. Paul, I., et al.: Cooperative boosting: needy versus greedy power management. In: ACM SIGARCH Computer Architecture News, vol. 41, no. 3. ACM (2013)

19. Shingari, D., et al.: DORA: optimizing smartphone energy efficiency and web browser performance under interference. In: 2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). IEEE (2018)
20. Gaudette, B., Wu, C.J., Vrudhula, S.: Improving smartphone user experience by balancing performance and energy with probabilistic QoS guarantee. In: 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE (2016)
21. Kim, Y., John, L., Paul, I., Manne, S., Schulte, M.: Performance boosting under reliability and power constraints. In: International Conference on Computer Aided Design (ICCAD), November 2013
22. Bircher, W.L., John, L.: Predictive power management for multi-core processors. In: Varbanescu, A.L., Molnos, A., van Nieuwpoort, R. (eds.) ISCA 2010. LNCS, vol. 6161, pp. 243–255. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24322-6_21