DORA: Optimizing Smartphone Energy Efficiency and Web Browser Performance under Interference

Davesh Shingari*, Akhil Arunkumar†‡, Benjamin Gaudette†, Sarma Vrudhula†, Carole-Jean Wu†
*School of Electrical, Computer and Energy Engineering
†School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Email: {dshingar,akhil.arunkumar,bgaudett,vrudhula,carole-jean.wu}@asu.edu

Abstract—This paper proposes DORA — a dynamic frequency controller that maximizes the energy efficiency of smartphones subject to user satisfaction demands in the presence of memory interference stemmed from background processes and co-scheduled applications. The proposed algorithm predicts the optimal energy-efficient frequency setting at runtime using statically-trained performance, dynamic power, and leakage power models. The parameters of the models represent web page characteristics and dynamically varying architecture and system conditions. The algorithm is designed, implemented and extensively evaluated on a Google Nexus 5 smartphone using a variety of mobile web browsing workloads. The results show high prediction accuracies for the performance and power models of 97.5% and 96%, respectively. Overall, DORA improves the smartphone’s energy efficiency by an average of 16% compared to the default Android frequency governor, interactive, while maintaining the desired levels of user satisfaction (web page load time).

I. INTRODUCTION

Today’s smartphone is a high performance, parallel processing computer, with a general-purpose chip multiprocessor, graphics processing units, digital signal processing units, and custom hardware accelerators. The increasing degree of parallelism and heterogeneity offered by the general-purpose programmable cores mean that a significant number of applications (or Apps) can now be executed simultaneously, leading to an ever increasing demand on the shared resources.

With the rapid improvement in the cellular technologies from 2G/3G to LTE/5G, the network latency in round-trip time has decreased significantly in the past decade, from a few 100 milliseconds to 10s of milliseconds [1], [2]. Thus, beyond the 2G/3G cellular technologies, and with advancement in the browser software protocol stack, e.g., SPDY [3], the performance bottleneck of mobile Apps shift from network performance to the performance of mobile processors. And, as the technology and system management techniques for handheld display [4]–[9] and communication fabrics [10] offer higher performance while consuming less power, the performance and energy efficiency of processors on mobile SoCs become more critical to the overall smartphone energy efficiency [1], [11], [12].

The advent of new execution paradigms on today’s smartphones also have a significant impact on their performance and energy efficiency. Starting from Android 4.2.2 and iOS 9, mobile operating systems (OS) support multiprogramming features, such as screen sharing between multiple Apps. For instance, when users utilize a social networking App, such as Facebook, in the foreground, image recognition algorithms could be running at the background as part of the Facebook App, while a video conferencing App can be running simultaneously. Such execution scenarios give rises to increased contention in the memory subsystem which, if not properly managed, results in significant interference and therefore performance degradation. This performance loss can lead to quality of service (QoS) violations, which is particularly problematic for real-time, interactive applications, as these directly manifest as lower user satisfaction. For certain applications, some amount of performance degradation can be tolerated without sacrificing user satisfaction; however, there are many applications for which missing an absolute QoS target is intolerable.

The web browser is a key smartphone application whose performance has a direct impact on user satisfaction. As the computation complexity and memory requirement of web pages continue to increase, the need to improve web browser performance and to ensure user satisfaction has come to the fore-front. While a number of prior works [13]–[18] have proposed solutions to address this need, they have all been limited to scenarios where the web browsers are running in isolation. However, these solutions could lead to sub-optimal performance and user satisfaction as realistic workloads and user scenarios generally consist of background tasks and other co-scheduled applications.

We observe that web browsers can be highly susceptible to the impact of interference due to multiprogramming execution. To illustrate this, we conduct an experiment to study the impact of interference on the web browser QoS (web page load time). Figure 1 shows the real system measurements of the web page load time for Reddit at different frequencies, when it is co-scheduled with applications with different memory intensities. The vertical bars and dots at each processor frequency show the variations in load time depending on the memory intensity of the interfering application. The horizontal dotted lines represent a 2-, 3-, or 4-second deadline, corresponding to

†The first and the second authors contributed equally to this work.

1The data is collected with a Google Nexus 5 smartphone by rendering Reddit concurrently with the other interfering applications in a multiprogrammed manner.
Different levels of expected QoS or user satisfaction. The results indicate significant variation in the load times subject to the degree of memory interference. This variation owing to the state of the co-scheduled processes can result in possible violation of QoS requirements. For example, with a 3-second load time deadline, the frequency setting of 0.9 GHz allows the web page to meet the QoS target when the memory interference is low. However, the web page would miss its deadline with greater interference. This demonstrates the need for an effective approach to maximize smartphone energy efficiency while guaranteeing the performance QoS of user-facing, web browsing, in the presence of interference, is met.

To tackle the optimization problem, this paper proposes to use the dynamic voltage and frequency scaling (DVFS) feature to simultaneously manage the contention in the memory subsystem and the smartphone energy efficiency. We use a model-based approach to predict the web browser performance and the smartphone energy efficiency in the presence of memory interference. The paper designs a Dynamic quality Of service, memoRy interference-Aware frequency governor, named DORA, to make online frequency setting predictions using statically-trained performance and, dynamic and leakage power models. The models utilize web page characteristics, dynamically-varying architecture and system conditions to capture web page complexity and the impact of memory interference due to co-scheduled applications. DORA explores the web page load time and smartphone power consumption at different frequency settings and selects the setting that allows web browsing to meet a specified deadline while simultaneously maximizing the energy efficiency of the smartphone.

DORA is implemented and evaluated on a real smartphone—a Google Nexus 5. The evaluation results show that DORA’s prediction models are highly accurate. DORA is able to predict performance and power consumption in the presence of memory interference with an accuracy of 97.5% and 96% respectively. Furthermore, direct measurements on a real platform show that DORA can effectively select the frequency setting so that the web browser meets the specified load time deadline 82% of the time, while maximizing the total smartphone energy efficiency. For the other 18% of the workloads, the web browser cannot meet the deadline even when running at the highest possible frequency setting. For these workloads, DORA performs the same as the baseline interactive governor. Across all workloads, DORA increases the overall smartphone energy efficiency by as much as 35% and by an average of 16% compared to the existing interactive governor.

The key contributions of this work are as follows:

- This work offers an effective system architecture solution to provide performance guarantee for mobile web browsing in the presence of interference, by using the DVFS feature to simultaneously modulate memory interference and to maximize overall smartphone energy efficiency.
- The key parameters for performance and energy efficiency prediction concerning web browsers subject to memory interference are identified.
- The proposed frequency controller, DORA, is implemented as a light-weight user space frequency governor within the Android OS and is evaluated on a real system platform. The insights and the DORA frequency governor proposed in this paper are also applicable to other smartphone platforms with re-parametrization.
- The effect of memory interference on smartphone energy efficiency and QoS has not received much attention. We hope the insights and results presented in this paper can advance the state-of-the-art understanding and inspire additional innovative solutions.

II. BACKGROUND AND MOTIVATION

In this section, we discuss important characteristics of web pages which determine the page load time (Section II-A) and show that the performance degradation and energy consumption of web browsing can be significantly increased by co-scheduled workloads (Section II-B), leading to deadline violation. Our characterization results in Section II-C demonstrate that an optimal energy-efficient operating mode exists which maximizes device energy efficiency while meeting the QoS requirement of web browsing.

A. Web Page Execution Flow

The execution flow of a web browser can be abstracted into two components: networking and rendering. The networking component provides the necessary communication and security requirements to fetch a web page’s content from the internet while the rendering engine evaluates the fetched content and provides a viewable representation of the web page for the user. Since the networking component’s efficiency is dependent on network latencies, and thus out of the system architect’s control, we focus on studying the performance of the rendering engine in this work.

The rendering engine parses a web page’s HTML document. An HTML page provides the blueprint of a web page by specifying two important components — tags and attributes of the web page. The tags are used by the rendering engine to determine outline of the various blocks of a web page. The attributes are associated with the tags and describe the
characteristics of the blocks. These tags and attributes are used to create a hierarchical structure called the DOM tree, which defines the rendering order of the different blocks of a web page. The DOM tree, with the CSS attributes (which determine the visual properties and style information), completes the render tree. This render tree goes through a layout and a final paint stage to complete the load process.

Prior studies [18], [19] have shown that web page load time is a function of the complexity of a web page and is dominated by important web page features, such as the number of tags, attributes, and the amount of meta data utilized by the web page. Since these properties of web pages are available before a page is rendered, the web page load time can be pre-computed fairly accurately. However, existing approaches cannot accurately estimate the web page load time in the presence of other system dynamics.

B. Impact of Memory Interference on Web Browsing

To quantify the degree of performance degradation and the additional energy cost introduced by co-scheduled applications onto web browsing, we design experiments to quantitatively evaluate the performance and energy impact.

Increased web page load time: Figure 2(a) shows the measured web page load times for four common web pages when co-scheduled with an interfering application with varying memory intensities. The x-axis represents the degree of memory intensities of the co-scheduled application (categorized as low, medium and high), and the y-axis represents the web page load time.

Depending on the web page complexity and the co-scheduled application’s memory intensity, some web pages meet the hypothetical deadline of 3 seconds while others do not. For example, ESPN was able to meet the deadline regardless of the degree of interference, while others, such as AliExpress, were not able to meet the deadline at any memory intensity. For web pages such as Hao123 and Imgur, as the memory intensity of co-run processes increases, the web page load time increases, which manifests as QoS deadline violations.

Increased device energy consumption: In addition to the increase in the web page load time, when web browsing is co-run with other workloads, additional energy overhead is incurred. Figure 2(b) shows measured values of the additional energy cost when a web browser and other applications run concurrently versus when they run separately. That is, if $E_B$ and $E_O$ denote the energy consumption due to the web browser and the application running separately, then the energy consumption in the co-run case is $E_B + E_O + E_\Delta$, where $E_\Delta$ is the additional energy due to running them simultaneously. Part of the energy overhead $E_\Delta$ is due to the longer web page load time while the rest is due to the additional movement of data that could be cached in the memory hierarchy but is evicted early due to interference. In Figure 2(b), the x-axis represents four different web pages while the y-axis represents the percentage increase $(E_\Delta/(E_B + E_O + E_\Delta))$ in energy consumption. The bars represent the intensity of the co-run application. This additional energy cost is high, corresponding to as much as 29% increase in energy consumption.

C. Optimal Operating Mode for Browsing with Interference

Modern processors are often equipped with dynamic voltage and frequency scaling (DVFS) to allow processors to operate at different performance and power regions. The availability of the DVFS control knob not only enables designs that...
predict and operate processors at an optimal energy-efficient setting [21]–[31] but also allows designs to control the degree of memory interference experienced in a multiprogrammed setting [32]–[35].

We measure the web page load time (performance) and evaluated the performance-per-watt (PPW) which represents the energy efficiency, for two workload combinations with varying processor frequency settings. Figure 3 shows the performance (web page load time) and the corresponding PPW of the web pages co-run with an interfering application from 729MHz to the maximum 2.2GHz settings. There is a frequency $f_E$ that maximizes the PPW. This is the unconstrained (i.e., with respect to a deadline) frequency setting that will result in the maximized battery lifetime. Now suppose that a deadline is imposed, and let the minimum frequency setting which ensures the web page meets the deadline, be $f_D$ (unknown). Then the optimal frequency setting $f_{\text{opt}}$ is given by

$$f_{\text{opt}} = \begin{cases} f_E & f_D \leq f_E \\ f_D & f_D > f_E \end{cases}$$

From Figure 3 we can see that for a web page like ESPN, the most energy efficient frequency setting $f_E$ would result in a large violation of the target load time. Therefore, in this case, $f_{\text{opt}}$ should be equal to $f_D$. In contrast, for a web page like MSN, $f_D$ that allows the web page to meet its QoS target would result in a significant energy cost, $f_E$, on the other hand, results in the optimal energy efficiency while allowing the web page to meet its QoS target. Scheduling the web page load process at the highest possible frequency is certainly an option that guarantees browsing QoS, but it results in a drastically lower energy efficiency, leading to 17% and 28% lower PPW than what the optimal PPW achieves with $f_{\text{opt}}$ for ESPN and MSN, respectively.

Furthermore, as we show in Section V-E, this optimal frequency setting could vary depending on the intensity of co-run applications. These experiments demonstrate the importance of an optimal smartphone frequency controller that can make dynamic predictions on the optimal frequency setting $f_{\text{opt}}$ by considering the memory interference, energy efficiency, and QoS targets simultaneously.

**Summary:** The real-system results presented so far have demonstrated processor DVFS as an effective control knob for managing the memory interference allowing web pages to meet its performance QoS deadline. It can also be used to significantly improve smartphone energy efficiency. With these insights, this paper aims to design an intelligent dynamic frequency prediction algorithm that, in the presence of background processes and co-scheduled workloads, predicts the $f_{\text{opt}}$ frequency setting at runtime to maximize the smartphone energy efficiency while enabling QoS-aware, satisfactory user web browsing experience.

**III. A NEW FREQUENCY GOVERNOR TO OPTIMIZE MOBILE WEB BROWSING WITH INTERFERENCE**

We propose a new frequency governor to dynamically set the PPW-optimal frequency subject to satisfying a given deadline for smartphones running a web browser. The algorithm is referred to as DORA which stands for Dynamic quality Of service, memorY interference-Aware frequency governor. The objective of DORA is to provide a high-quality web browsing experience for users while maximizing battery lifetime.

The pseudo-code of DORA implementation is presented in Algorithm 1. The first task is to accurately predict the range of core frequencies, $f_{i...n}$, that ensure the web pages complete rendering within the given deadline (lines 4-6 in Algorithm 1). The second task is to accurately identify a core frequency ($f_{\text{opt}}$) within $f_{i...n}$ such that the energy efficiency of the smartphone device is maximized (lines 7-15 in Algorithm 1).

The main components of DORA include models to predict the web browser performance and to predict the power consumption of the smartphone. The performance (web page load time) model includes the complexity of web pages, the degree of memory interference introduced by background processes and co-scheduled applications, and the core operating frequencies (Section III-A). The power model accounts for the dynamic power as a function of the core frequency and the leakage power as a function of core temperature (Section III-B). These models are used to estimate the device energy efficiency, performance-per-watt (PPW), at each frequency. Then, the PPW at each setting is used to set the frequency, $f_{\text{opt}}$ that maximizes the PPW.

To take into account the dynamic nature of interference from co-scheduled applications, DORA monitors the intensity of memory interference and determines $f_{\text{opt}}$ for the current time period and adjusts the core operating frequency to $f_{\text{opt}}$ periodically. Figure 4 shows this iterative process, as DORA executes in the background.
A. Web Page Load Time Prediction

We construct the web page load time and device power models using regression. A regression model is a hypothesized parametric relationship between the response or dependent variable, $y$, and a set of $N$ independent variables $X_1$, $X_2$, ..., $X_N$. The unknown model parameters are the coefficients to the polynomial combinations of $X_i$, which are estimated by minimizing the mean-square error between a set of observed values and model predicted values. Similar to Zhu et al. [18], we observe that five important parameters of web pages best represent web page complexity and hence the impact on web page load time—the number of DOM Tree node, class and href attributes, a and div tags. Therefore, we include these parameters in the web page load time prediction model.

Next, we identify the runtime architectural parameters that influence the web page load time. In order to account for memory interference on the performance of web browsing, the degree of interference in the shared memory is considered, i.e., the access rate in the shared L2 cache and DRAM.

Finally, the core and the memory bus frequencies also have a pronounced impact on the web page load time. Specifically, we note that on a typical SoC, a set of core frequencies map to a particular memory bus frequency. Therefore, we build piece-wise models for each set of core frequencies that share a single memory bus frequency. With the above insights, we construct a set of independent variables which demonstrate a strong correlation with web page load time (Table I).

We evaluate three typical response surfaces: the linear, quadratic, and interaction models (Equations (2)–(4)).

\[ L = c_0 + \sum_{i=1}^{N} c_i X_i \]  
\[ L = c_0 + \sum_{i=1}^{N} c_i X_i + \sum_{i,j \in (1..N), i \neq j} c_{i,j} X_i X_j \]

where $L$ is the web page load time and $X_i$ represents the independent variables, and the $c_i$’s and $c_{i,j}$’s are the coefficient parameters to be determined. We provide additional details about the web page load time model in Section IV.

B. Dynamic and Leakage Power Prediction

DORA’s power model includes the dynamic power $P_{dyn}$ and the leakage power $P_{kg}$. $P_{kg}$ depends on the compute and memory resource utilization of the cores and the corresponding processor voltage and frequency settings, and $P_{kg}$ depends on the operating voltage and temperature.

**Dynamic Power Model:** Similar to the timing model, web page complexity is a good predictor for the dynamic power consumption. In addition, the degree of memory interference is also a key factor that significantly contributes to the smartphone power consumption. As interference at the L2 cache increases, additional data movement is required to fetch data into the L2 cache upon demand. Thus, the parameter of L2 cache MPKI is included in the dynamic power model. To consider the dynamic power contribution from the cores running background processes or co-scheduled applications, core utilization is used. Core utilization has been shown to have a linear relationship with $P_{dyn}$ for general compute-bound workloads. Finally, the core operating frequency has a direct effect on $P_{dyn}$. We again utilize the same three response surface models as in Equations (2)–(4); however, the dependent variable is now the dynamic power consumption.

**Leakage Power Model:** Due to the lack of cooling elements in most smartphones, high thermal levels often contribute to a significant portion of the total device power budget in the form of leakage power. Therefore, when predicting the total smartphone power consumption, we must include the effect of leakage power consumption such that the energy-efficient setting prediction for $f_{opt}$ considers this important, dynamic system runtime condition. We utilize the empirical model of [36] to capture the non-linear power dependence on temperature and voltage as exhibited by CMOS-based technologies:

\[ P_{kg} = k_1 v T^2 e^{\frac{\alpha + \beta}{T}} + k_2 e^{(\gamma v + \delta)} \]
TABLE II: Device Specification

<table>
<thead>
<tr>
<th></th>
<th>Google Nexus5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Android KitKat 4.4</td>
</tr>
<tr>
<td>Chipset</td>
<td>MSM8974 Snapdragon 800</td>
</tr>
<tr>
<td>Application Processor</td>
<td>Quad-core Krait</td>
</tr>
<tr>
<td>ISA</td>
<td>ARMv7</td>
</tr>
<tr>
<td>L1 I/D Caches</td>
<td>Private 16KB per core</td>
</tr>
<tr>
<td>L2 Unified Cache</td>
<td>Shared 2MB</td>
</tr>
<tr>
<td>GPU</td>
<td>Adreno 330</td>
</tr>
<tr>
<td>DSP</td>
<td>Hexagon DSP</td>
</tr>
<tr>
<td>Memory</td>
<td>LPDDR3 2GB</td>
</tr>
</tbody>
</table>

where $k_1, k_2, \alpha, \beta, \gamma,$ and $\delta$ are parameters that depend on circuit topology, and $v$ and $T$ are the operational voltage and temperature of the SoC respectively. The parameters of the leakage model are determined using non-linear numerical solutions and mean square error minimization. For SoC chipsets that implement multiple voltage and temperature domains, e.g., per-core thermal sensors, we construct a leakage model for each computational unit to increase the model accuracy.

IV. METHODOLOGY

A. Real Device Measurement Infrastructure

We perform all experiments on a Google Nexus 5 smartphone which has a Qualcomm MSM8974 Snapdragon 800 chipset [37]. The MSM8974 includes four Krait cores, each of which is equipped with private L1 instruction and data caches (16KB each). All four cores share a 2MB L2 last-level cache. The SoC consists of a 2GB Low Power DDR (LPDDR) memory which is shared between the application processor and the various accelerators. The chipset has 14 different frequency settings available, ranging from 300MHz to 2265MHz.

Table II summarizes the device specifications. The device runs the rooted Android OS. We also configure the kernel to enable performance profiling using perf [38]. We used a National Instruments Data Acquisition Unit (DAQ) to measure smartphone power consumption. Note, our smartphone power measurement and energy efficiency results include the power consumption of the entire smartphone (the display, the application processors, SSD, and all other active components on the device.) Thus, the energy efficiency improvement results directly translate to battery life improvement.

DORA is compared with the existing Android frequency governors, interactive and performance. The performance governor always operates the cores in the highest available frequency of 2.2GHz. The interactive governor on the other hand chooses a frequency setting based on the processor utilization. We use interactive as the baseline for our studies as it is the default option on most smartphones to-date.

B. Workload Characteristics

We use the 18 most visited web pages reported on “Alexa top 500 websites” [39] that load completely on an Android smartphone. These pages represent a wide variety of domains such as online shopping, sports, entertainment, news, and social media. They also vary widely in complexity resulting in load times in the range of hundred of milliseconds to 4 seconds, when running alone. The Firefox mobile web browser is used to load the pages. The source code of web pages is instrumented to enable the load time measurement. All web pages are stored in memory, eliminating any non-deterministic network fluctuation. Table III shows the web pages used in this paper and the classification. Similar experimental methodology is used in recent works [17], [18], [24], [34], [40]–[43].

We use a diverse set of workloads to serve as the co-run applications. The algorithms behind these workloads form the basic building blocks of current and future smartphone workloads, representing sensor data analysis, image processing, thermal prediction and management, video games, and medical applications [44]. We classify the algorithms based on their memory intensity (Table III). All the co-run applications are cross-compiled using the ARM-Android NDK toolchains and are statically assigned to a specific core. The applications are pushed to the device and launched using the Android debug bridge (adb) terminal.

We construct workloads to mimic multiprogrammed execution scenarios. Specifically, the Firefox browser is executed on two cores while a co-run application is executed on the third core of the application processor. We create workloads by combining a web page with an application from each memory intensity category shown in Table III. This results in a total of 54 workload combinations, i.e., 18 web pages, each co-scheduled with an application from the low, medium, and high intensity categories. 14 of the 18 web pages have been used to construct the models and thus the multiprogrammed

TABLE III: Web Page and Co-run Application Classification

<table>
<thead>
<tr>
<th>Intensity</th>
<th>Load Time</th>
<th>Web Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>&lt; 2 Sec</td>
<td>Amazon, Twitter, Youtube, 360, MSN, BBC, CNN, Reddit, Alibaba, eBay, Alipay, Instagram</td>
</tr>
<tr>
<td>High</td>
<td>&gt; 2 Sec</td>
<td>IMDB, ESPN, Hao123, Imgur, Aliexpress, Firefox</td>
</tr>
</tbody>
</table>

3 A recent characterization study has shown that the average thread-level parallelism for mobile Apps hovers around 2 [45]. Performance scaling with respect to the number of cores plateaus at 2 cores. Furthermore, as an increasing number of cores are integrated into SoCs from one generation to the next, we expect core-level computation resource contention is less of a performance problem as compared to the highly shared and contended memory subsystem.

6 The fourth core was switched off for all our experiments.
workloads formed with the 14 web pages are considered as the training set, resulting in 42 Webpage-Inclusive combinations\(^7\). The remaining 12 workloads form the test set and are referred to as the set of Webpage-Neutral workloads.

C. Model Parameters and DORA Configuration
Over 300 measurements of power and web page load times are taken by executing multiple workload combinations at different frequency settings, using the setup described in Section IV-A. The observations are used to determine the coefficients of the power and performance models using mean square error minimization.

For DORA’s decision making granularity, we evaluate three decision intervals of 50ms, 100ms, and 250ms. We observe that while 250ms is too slow to capture web page phases, 50ms and 100ms decision intervals perform similarly. Therefore, we choose the less intrusive 100ms decision interval for DORA.

V. Evaluation Results and Analysis
A. Performance and Power Model Evaluation
We evaluate three regression models — simple linear regression, linear regression with cross product terms (interaction) and quadratic, for the performance and power models.

**Performance Model Accuracy**: We observe that the interaction and quadratic models achieve the highest accuracy for web page load time prediction. Due to relative simplicity of the interaction model, we choose this to model the web page load time. The average error rate for this web page load time model is 2.5%. Figure 5(a) shows the cumulative distribution of prediction errors. Each (x, y) shows the fraction of web pages (y) which have errors lower than or equal to (x). About 87.5% of the web pages have less than 5% error with a maximum error of 10%.

**Power Model Accuracy**: In case of power consumption estimation, we observe that all three models (linear, interaction and quadratic) achieve a similar prediction accuracy. Since a linear model is simpler than the other two, we adopt it to predict the power consumption of the web page load process in the presence of interference. The average error rate for the power model is 4%. Figure 5(b) shows the cumulative distribution of prediction errors in the power model. For 75% of web pages, the model gives less than 5% error, and for 90% of web pages, it gives less than 10% error.

B. Sensitivity of \(f_{\text{opt}}\) to Model Errors
The value of the optimal frequency, \(f_{\text{opt}}\), is not highly sensitive to model errors. This is because the available processor frequencies are discretized (typically in steps of a few hundred MHz). Therefore, as long as the error rate of the estimated PPW for \(f_{\text{opt}}\) is lower than the PPW delta between \(f_{\text{opt}}\) and its adjacent frequency settings, \(f_{\text{opt} - 1}\) and \(f_{\text{opt} + 1}\), DORA’s prediction for \(f_{\text{opt}}\) would still be correct. Figure 6 illustrates this with an example with the PPW variation for loading a Youtube web page, co-run with a memory-intensive application. For this workload, the highest energy efficiency is attained at \(f_{\text{opt}} = 1.2\, \text{GHz}\). As noted in Figure 6, the neighboring frequency \(f_{\text{opt} - 1}\) would result in a load time difference of \(\Delta t = +20.3\%\), and power difference of \(\Delta P = -13.3\%\), compared to those at \(f_{\text{opt}}\). Similarly the neighboring frequency \(f_{\text{opt} + 1}\) would result in \(\Delta t = -20.8\%\) and \(\Delta P = +34.8\%\). Let \(t_e\) and \(P_e\) be the percent error in the performance and power models. Then the PPW in the presence of error is

\[
PPW = \frac{1}{P * t * (1 + \Delta t) * (1 + \Delta P)}
\]

where \(P, t\) are actual power and web page load time values.

DORA would select the optimal energy-efficient frequency setting (\(f_{\text{opt}} = 1.2\, \text{GHz}\)) as long as the errors in the performance and power models are small enough such that DORA does not mistakenly choose either of the neighboring frequencies as the optimal setting. Analysis of this workload shows an error of 10.26% in power prediction and 1.32% error in load time prediction which can be easily tolerated by the discretization of frequency settings. This means DORA’s selection of \(f_{\text{opt}}\) is accurate in the presence of small errors in power and load time prediction. This holds true for most other workloads as well.

\(^7\)Webpage-Inclusive workloads are constructed by co-scheduling one training set web page with one distinct interfering application, since the most commonly-visited web pages are relatively stable while co-scheduled applications or background processes vary more frequently.
Deadline is a governor which ensures that web page load time target is met while disregarding energy efficiency, in energy efficiency when compared to EE constraint. On average, EE results in a greater energy efficiency than DORA, it also results in 21% of the workloads missing the QoS targets by a large margin, as seen in Figure 7(b). EE could result in web page load time as large as 6 seconds, which are simply unacceptable from a user satisfaction standpoint. Using the DL governor, the 3-second load time deadline is satisfied as long as it is feasible for a given workload combination. While this allows for more web pages to meet the QoS targets (compared to EE), it results in sub-optimal energy efficiency, as seen in Figure 7(a). The sub-optimal energy efficiency of DL and large QoS violations of EE highlight the importance of simultaneously considering both web page load time deadline and dynamic and leakage power consumption.

Importance of the Consideration of Memory Interference and Other Dynamic Conditions: The $f_{opt}$ setting depends on the QoS deadline specified by users or by the OS (Section V-G), the specific web page being loaded, and the dynamic system conditions. In particular, we observe that $f_{opt}$ changes significantly (often by more than one frequency setting away) when considering the varying degree of memory intensity of co-scheduled and background processes and the operating temperature of the smartphone (Section V-F). If memory interference is not considered in the performance prediction, a sub-optimal $f_{opt}$ is reached, leading to deadline missing for more than 64% of the multitasking workloads and energy efficiency degradation.

### C. Performance and Energy Efficiency Trends

Figures 7(a) and 7(b) show the average energy efficiency improvement and the distribution of web page load time achieved by DORA and other governors. On average, DORA improves the smartphone energy efficiency by 16% compared to the baseline interactive governor (18% and 10% for the Webpage-Inclusive and the Webpage-Neutral workloads, respectively). Frequency settings based on the load time predictions meet the QoS target of the 3-second deadline whenever possible with the available frequency settings. Although performance and interactive generally achieve faster web page load time than DORA (Figure 7(b)), this comes at the cost of lower energy efficiency (Figure 7(a)). Furthermore, DORA performs as well as a static offline optimal configuration, $Offline_{opt}$, and matches the energy efficiency improvement brought by $Offline_{opt}$.

DORA is also compared with two hypothetical governors — Deadline (DL) and Energy Efficient (EE). Deadline is a governor which ensures that web page load time target is met while disregarding energy efficiency, and Energy Efficient is a governor which ensures that energy efficiency is maximized while disregarding any QoS constraint. On average, EE results in a 19% improvement in energy efficiency when compared to interactive. Although $Offline_{opt}$ represents the single frequency setting that maximizes the energy efficiency achieved while loading the web page within the 3-second deadline. We obtained the PPW results for $Offline_{opt}$ by enumerating all possible frequency settings for ten randomly chosen workloads from the workloads constructed in this paper because the time taken to generate the PPW results for all possible frequency settings for all available workloads is prohibitively high.

D. The Adaptive Nature of DORA

DORA is designed to dynamically predict the web page load time and energy efficiency in order to determine $f_{opt}$ given in Equation 1. It does so by estimating $f_E$, the most energy-efficient frequency setting, and $f_D$, the lowest frequency that meets the deadline. Figure 8 shows the improvement in energy efficiency achieved by the different governors, for all evaluated workloads. For workloads 20 and beyond, DORA closely follows the energy efficiency trend of the EE.
governor. Thus, both DORA and EE result in an average improvement in energy efficiency of 24% when compared to the interactive governor. It is important to note that for these workloads, DORA’s predictions result in meeting the 3-second load time deadline while achieving the maximum possible energy efficiency.

For workloads 1 through 19, the EE governor violates the 3-second web page load time deadline. We denote these workloads as “$f_E < f_D$”. In such scenarios, since EE ignores the QoS, it continues to operate in lower but energy efficient frequency settings even when these settings do not meet the performance target. EE results in large violations in QoS.

For workloads where EE is not able meet the web page load time QoS target, DORA correctly identifies the frequency range that allows the web page to meet its QoS target, and then selects the most energy efficient frequency setting within that range. This frequency often coincides with the frequency chosen by the DL governor for these workloads. Figure 8 shows that DORA follows DL closely for these workloads. Generally, DORA satisfies the 3-second deadline as long as the deadline is met by the performance governor. For workloads where the web pages cannot load within the 3-second deadline at the higher frequency setting, DORA prioritizes for QoS and chooses the highest frequency setting to ensure that the web pages are loaded as fast as possible. 

This resiliency to variations highlights the fact that an optimal frequency governor must make its frequency decision considering runtime factors, such as memory interference, in order to guarantee the quality of web browsing in the most energy efficient way.

E. DORA with Different Web Page Complexity and Memory Interference Intensity

The complexity of web pages vary and so does the intensity of co-scheduled applications. To explore the behavior of DORA in such varied conditions, we take a closer look at one low complexity web page (Amazon) and a high complexity web page (IMDB). Figure 9 shows the behavior of DORA and the other governors for these two web pages when they are co-scheduled with low, medium and high intensity applications. The improvement in the PPW compared to the interactive governor is shown on the primary y-axis. The secondary y-axis shows the web page load time. Three groups of plots are shown, corresponding to low, medium and high memory intensities. The abscissa represents the frequencies determined by the governors performance, DL, EE, and DORA. To avoid confusion, note that the vertical bars represent the PPW and the line graph represents the web page load time.

**Frequency of operation:** Due to the fact that the low complexity web pages are relatively simple to load, $f_D$ typically hovers around 0.7-0.9 GHz. For example, Amazon in Figure 9 has its $f_D$ at 0.7 GHz. These low frequency points are often not the most energy efficient frequency settings. The $f_E$ point for Amazon occurs at 1.9 GHz when it is co-scheduled with a low intensity application. Therefore, the low complexity web pages typically fall under the “$f_E > f_D$” category. For this category of workloads, as discussed earlier, DORA’s behavior is similar to that of EE. DORA and EE both select the same operating frequencies for Amazon. As the complexity of the web pages increases, $f_D$ shifts to higher frequency range. For instance, IMDB, a high intensity web page has its $f_D$ at 1.9 GHz or 2.2 GHz, depending on the intensity of the co-scheduled application. This makes these high complexity web pages to typically fall under the “$f_E \leq f_D$” category and DORA follows $f_D$’s behavior. DORA and DL choose the same frequency for IMDB.

**Energy efficiency improvement:** DORA improves the smartphone energy efficiency by as much as 27% for Amazon. On the other hand, for workloads in the situation where “$f_E \leq f_D$”, it is expected that $f_{opt}$ should be closer to the highest frequency setting. This in turn means that the energy efficiency improvement achieved by DORA is more modest when compared to the interactive governor. DORA achieves a modest 1%-10% energy efficiency improvement for IMDB, in order to meet the web page load time deadline.

**Impact of memory interference:** The presence of memory interference influences the behavior of the smartphone SoC in multiple ways. First, as memory interference changes, the frequency of operation might change. For example, Figure 9 shows that for Amazon, $f_D$ changes from 0.7 GHz to 0.8 GHz as we move from a medium to a high intensity interfering application. We observe a similar behavior for IMDB as well where $f_D$ changes from 1.9 GHz to 2.2 GHz. Next, as expected, the web page load time is often degraded as memory interference increases. This is evident in the case of Amazon
and IMDB. This web page load time increase could eventually result in web pages missing the QoS targets.

F. Impact of Leakage Power

An important feature of DORA is its consideration of temperature as it influences the selection of $f_{\text{opt}}$. Prior work, such as [17], does not consider the leakage power component when optimizing for energy efficiency. This is likely to lead to a sub-optimal frequency setting. In order to evaluate the impact of leakage power on energy efficiency, we compare DORA with a configuration that does not take leakage power into account, DORA_no_lkg. That is, DORA makes the frequency selection decision using the dynamic power consumption component only. Figure 10(a) shows the energy efficiency of DORA and DORA_no_lkg for the web page Amazon when it is co-scheduled with a medium memory intensity application. DORA achieves 10% higher energy efficiency compared to the configuration that ignores the dynamic operating temperature of the smartphone.

We further explore the impact of temperature (and consequently leakage power) by evaluating the smartphone power consumption at different frequencies and $f_{\text{opt}}$ under room temperature and cooler ambient temperature. Ignoring the processor operating temperature when determining $f_{\text{opt}}$ leads to an energy efficiency sub-optimal solution, with 10% lower battery lifetime (Figure 10(a)). Figure 10(b) shows that there is a significant increase in the power consumption at higher frequencies at room temperature, compared to a lower ambient temperature condition. This increase in power consumption can be attributed to the additional leakage power due to the increase in device temperature. We observe that the maximum device temperature increases from 58°C to 65°C when operating at 1.9 GHz under the ambient room temperature. The temperature rise is significantly higher at higher frequencies than lower frequencies, making leakage power a significant contributor to device power at high frequencies. This increase in temperature and consequently leakage power result in the optimal operating point, $f_{\text{opt}}$, shifting from 1.9 to 1.7 GHz for this workload. DORA is able to predict this significant additional leakage power as shown in Equation 5 and identify $f_{\text{opt}}$ accurately, leading to higher energy efficiency.

G. DORA with Varying Performance QoS Deadline

DORA is designed to predict web page load time and smartphone power in order to operate a smartphone at the most energy efficient condition, $f_{\text{opt}}$. We have already seen that DORA either chooses $f_E$, the most energy-efficient frequency setting, or $f_D$, the lowest frequency that meets browser deadline. To highlight the behavior of DORA for varying deadline, we look at $f_{\text{opt}}$ chosen by DORA when MSN is being loaded with a high memory intensity application. Note, the models used by DORA do not need to be re-parameterized for using a different QoS deadline. Figure 11 shows $f_{\text{opt}}$ for different deadlines, from 1 to 10 seconds. For a demanding performance deadline, e.g., 1 to 2 seconds, DORA chooses the highest frequency in order to meet the QoS target. When the performance target is relaxed to 3-second, $f_{\text{opt}}$ becomes 1.7GHz, that is the most energy-efficient, deadline-meeting setting. When the performance target is further relaxed, $f_{\text{opt}}$ switches from $f_D$ to $f_E$, which is 1.19GHz for this workload.

H. Overhead

From the implementation standpoint, DORA includes three key operations, namely, (1) periodically assessing hardware performance counters, (2) computing the optimal frequency point, $f_{\text{opt}}$, and (3) switching the core frequency to $f_{\text{opt}}$ if the newly computed $f_{\text{opt}}$ is different from the current setting. DORA is a lightweight controller. Its time and power overhead coming from the first two tasks mentioned above is less than 1%. This is because the first two steps are non-intrusive to the web page load process and occur in the background. Although most prior work assume a relatively small overhead for the frequency scaling operation mentioned above, we observe that this overhead is slightly higher than that of the first two tasks with a maximum of 3% of execution time. DORA monitors the variation in the runtime system performance conditions and decides to change the frequency setting only when the system performance conditions have changed significantly enough to alter $f_{\text{opt}}$. This results in the overhead of DORA to be dependent on the number of times frequency is scaled during the web page load process. This
overhead is negligible for workloads which enjoy a relatively stable phase behavior. For other workloads where DORA scales the processor frequency often, the energy efficiency improvement brought by DORA is high; thus, the overhead associated with the needed frequency scaling is considered to be worthwhile. Overall, the performance and energy efficiency results presented in this paper include the overhead incurred and still show a significant 18% energy efficiency improvement over the interactive governor.

VI. RELATED WORK

The study of smartphone web browser performance has been the subject of many recent works as it is one of the most widely used applications on mobile devices. Web browser performance optimization can be achieved through enhancements at multiple levels of the hardware-software stack. Many of the early works on browser optimization focused on improving browser specific tasks through software techniques such as task parallelization, browser rendering, and smarter browser caching, such as [13], [14], [16]. Butkeiwicz et al. [19] related the webpage complexity with important webpage primitives and characterized the impact of webpage complexities on performance. Thiagarajan et al. [15] presented a detailed breakdown of webpage energy consumption based on the different webpage primitives and proposed a set of webpage design recommendations to minimize energy consumption. Similarly, Bui et al. [13] used webpage primitives to design energy efficient web browsers.

Prior works have also suggested that micro-architectural techniques such as branch predictors, advanced cache and prefetcher management techniques can improve browser performance and, therefore, reduce its energy consumption significantly [40], [46]. Zhu et al. proposed hardware specializations to improve the performance and energy efficiency of mobile web browsing [47]. Another recent work by Fan et al. [48] demonstrated improved browser efficiency with asymmetric multiprocessors sharing the cache. These software and micro-architecture level works are orthogonal to our proposed DORA which performs energy efficiency optimization in the system level. Therefore, we expect the performance and energy efficiency gains from DORA to be additive.

Many other system level designs, aimed at improving the energy efficiency and QoS of mobile browsers have been reported. Lo et al. [42] considered the response time and the limits of human perception together to find opportunities to throttle frequencies while executing interactive applications on an Odroid SoC board. While their design is QoS-aware, it is not necessarily energy optimal as we demonstrate with the DL governor in this paper. Zhu et al. [18] developed models for webpage load time and energy consumption to design a deadline- and energy-aware governor but the effect of memory interference from background or co-scheduled processes is not considered. Another recent work by Gaudette et al. [24] developed probabilistic models to account for non-determinism in webpage load time and demonstrated improved QoS for the web browser.

Although many of above works optimize for QoS and energy efficiency for mobile web browsing, none of them explicitly consider the effect of memory interference. As we have shown in this paper, memory interference plays a key role and impacts both webpage load time and energy efficiency significantly for modern smartphones. This is the first work that designs an effective solution to provide performance QoS guarantee for mobile web browsing in the presence of background processes and other co-scheduled applications.

VII. CONCLUSION

We develop a framework capable of characterizing, profiling, and predicting the execution time and power consumption of a given webpage as the foreground application, subject to memory interference from co-scheduled applications and smartphone operating temperature. The proposed framework performs offline training for the performance and power models. This training is conducted on the most viewed webpages co-scheduled with interfering applications commonly seen on mobile devices. The models are proved to be sufficiently accurate in predicting webpage load times and power consumption with 2.5% and 4.0% average error respectively. Once the models are parameterized, DORA performs runtime, energy efficiency optimal control of the application processor to ensure that webpages are loaded within a targeted QoS level, by modulating the processor core frequencies to simultaneously control memory interference and performance/power states of the smartphone. This is done by reading the webpage characteristics along with the dynamically changing states of the co-scheduled processes and the smartphone—memory access intensity, core utilization, and temperature. The proposed design is implemented and evaluated on a Google Nexus 5 smartphone and proven to be capable of providing satisfactory performance for mobile web browsing despite the interference from memory intensive co-scheduled applications.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their useful feedback. This work is supported in part by the National Science Foundation under grants for the I/UCRC Center for Embedded Systems #1361926, CCF #1525462, and CCF #1652132.

REFERENCES
