# Energy Wars - Chrome vs. Firefox

## Which browser is more energy efficient?

João de Macedo, João Aloísio, Nelson Gonçalves
University of Minho, Portugal
{a76268,a77953,a78173}@alunos.uminho.pt

Rui Pereira
HASLab/INESC Tec, Portugal
ruipereira@di.uminho.pt

João Saraiva
University of Minho &
HASLab/INESC Tec, Portugal
saraiva@di.uminho.pt

## ABSTRACT

This paper presents a preliminary study on the energy consumption of two popular web browsers. In order to properly measure the energy consumption of both environments, we simulate the usage of various applications, which the goal to mimic typical user interactions and usage.

Our preliminary results show interesting findings based on observation, such as what type of interactions generate high peaks of energy consumption, and which browser is overall the most efficient. Our goal with this preliminary study is to show to users how very different the efficiency of web browsers can be, and may serve with advances in this area of study.

## KEYWORDS

Energy Efficiency, Web Browsers, Green Software

**ACM Reference Format:**
João de Macedo, João Aloísio, Nelson Gonçalves, Rui Pereira, and João Saraiva. 2020. Energy Wars - Chrome vs. Firefox: Which browser is more energy efficient? . In *35th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW '20), September 21–25, 2020, Virtual Event, Australia.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3417113.3423000

## 1 INTRODUCTION

With the advancement of time, technology has become more integrated in our lives and, with that, the use of the internet has become a tool in our everyday habits that we use at any moment, either to obtain some information or for simple leisure. The truth is, browsers have become almost unattachable from an internet experience, and where we depend upon them for everything web related. In addition, environmental concerns about green and sustainable IT have qualified energy consumption as one of the most important concerns in the design of electronic and computer systems, in particular due to the growing popularity of mobile devices. When we use notebooks to perform something online, we use web browsers whether for social life, personal research, watch videos, play games, work, etc. Herewith, we can say that web browsers are one of the most important and used internet tools [2, 17, 23, 26]. There's a great browser competition with that, because there's a great variety in that offer and every user can choose which one best suits their needs and demands.

Thus, with advances and impulses for easier use, convenience and omnipresence, the use of notebooks has been highly adhered to. Nevertheless, the battery efficiency and thus energy efficiency of a notebook is very important for a user. It is therefore important to better understand what is the lowest energy-consuming browser alternative, from the variety at hand but also without forgetting the performance of the browser having a balance between performance / energy, in order to reach the most efficient browser.

In this paper, we present our browser test-bed and preliminary study on two popular browsers: Google's Chrome and Mozilla's Firefox. Our test-bed uses Intel's RAPL (Running Average Power Limit) to precisely measure browsers energy consumption, and using Selenium[1] to generate simulation scripts for each browser. Thus, we analyze the energy efficiency of both browsers based on the same exact testing conditions

Thus, the aim is to present findings and information on the energy efficiency of two popular browsers, making it easier for users to choose their default web browser if energy efficiency is indeed a concern.

This remainder of this paper is organized as follows: Section 2 presents the detailed steps of our methodology to measure and compare energy consumed by different browsers. Section 3 contains the analysis and discussion on the obtained results, where we first present the results, and afterwards discuss the findings, details, and curiosities. Section 4 discusses the threats to validity of our study. Section 5 looks at related work, and finally, in Section 6 we present the conclusions of our work.

## 2 METHODOLOGY

As previously mentioned, the aim of this work is to understand which browser is the most appropriate to be used, if energy consumption is of concern to the user. Additionally, depending on the web application being used, or the type of task (be that watching videos, live streaming, or accessing social media), the choice of browser may vary. Thus, with this in mind, we construct the following research questions:

- **RQ1**: *Which web browser is the most energy efficient for browsing Youtube?* According with Alexa[2] Youtube is the most, popular video viewing site on the Internet. Understanding which browser is the most energy efficient can help heavily reduce the energy consumed during this common web browsing.

[1] https://www.selenium.dev/
[2] https://www.alexa.com/topsites



*ASEW '20, September 21–25, 2020, Virtual Event, Australia*

ACM ISBN 978-1-4503-8128-4/20/09.
https://doi.org/10.1145/3417113.3423000

- **RQ2**: *Which web browser is the most energy efficient for live streaming on Twitch?* Twitch is a modern and widely used live streaming site, primarily focused on video game live streaming, also during the Covid-19, it was widely used by academics for course presentations. Once again, understanding which browser is the most efficient can help users choose a way to reduce their energy footprint.
- **RQ3**: *Which web browser is the most energy efficient for social media?* Social media has become something very common nowadays, and, yet again, knowing which web browser can help reduce energy consumption can give users more information for choosing a less energy footprint producing browser.
- **RQ4**: *Which web browser is the most energy efficient overall?* While there may be browsers more suited to certain tasks and web applications, understanding overall in a more general sense which is the most energy efficient one can further help users choose their used browser.

## 2.1 Design

Nowadays, browsers can almost seen as an integral part of a user's computing experience. This importance is linked to the rise of technology as most users who thoroughly use the internet do so through a browser. As such, we see companies like Google, Mozilla, Microsoft, and Apple develop and heavily maintain their own browsers. Users consider many factors when going into choosing their favored browser, such as speed, resource management, themes, plugin compatibility etc. What users do not currently consider, due to the lack of information, is the energy efficiency of the browsers. While many might assume speed directly equates energy efficiency, there are several research works showing this is in fact not always direct [1, 5, 19, 20].

According to statcounter [3] and w3schools [4], the most used browsers are *Google Chrome* and *Mozilla Firefox*, therefore, we decided to compare these two browsers. While we did initially intended to also explore *Safari* and *Microsoft Edge*, these two did not meet the compatibility requirements for our energy measurement framework (which will be discussed further on), due to not being available in Ubuntu. As such, they were not compared in this preliminary study.

The first step to obtain comparable and representative results regarding the energy consumed in each browser is to create execution scripts simulating real browser usage. These scripts simulate user actions on different websites, and the goal is to check different aspects of each application trying to mimic the typical user's usage.

Using the Selenium tool, an open-source automated testing framework for web applications across different browsers and platforms, we produced several usage scenario scripts for the browsers under test. To help answer **RQ4**, we wrote a global script where the browser would open and navigate various web pages simultaneously on different tab bars. This script performs the following steps: Log in into a Google account, go to Google sheets and upload one spreedsheet, navigate to Facebook, log into Facebook, navigate and open news on a sport news blog[5], navigate and open articles on a technology news site [6] and afterwards on a national news site[7], and finally watch a video [8] on YouTube, where the viewing begins with a video on 1080p and at some point switches to 4k. This script takes 30 minutes to complete. This sequence can be seen in Figure 1.

In order to help answer **RQ3**, another script was generated to simulate Facebook users to: log in to an account, search for a public sport page[9], and browse photos, videos and publications.

In order to further complement **RQ3**, another script performs the following steps on Instagram: log in to an account, search for one public page[10], navigate and watch public Instagram stories[11], watch several publications and browse photos.

To analyze browsers' energy consumption when on YouTube, and help answer **RQ1**, another Selenium script performs the following: navigate to YouTube[12], search for and watch a specific 4k video[13], afterwards return to the YouTube main page, search for and watch a 1080p video[14].

To analyze live streaming on Twitch and YouTube, the script would respectively navigate to the broadcasting channels of Electronic Sports League (ESL) for the popular competitive game Counter-Strike:GO, for Twitch[15] and YouTube[16].

Finally, in addition to the previous scripts, we also wrote a script to look at the energy consumption of Google Drive sheets which would: open a large spreadsheet, sort it, refresh the page and sort it again. These steps are all performed in a loop for 10 minutes.

## 2.2 Execution

To be able to analyze the energy consumption of browsers, we must first extract energy readings. For this, we used to Intel's RAPL [8], and the Java based RAPL framework jRapl [16] to monitor the energy consumption during each script's execution. This allows us to record precise energy consumption measurements from several hardware components (such as CPU and DRAM) as RAPL is a very reliable tool [10, 24] and used in many comparable studies [7, 15, 20], with a sampling rate of 10 measurements per second.

In order to obtain structured data and facilitate analysis, the jRAPL program was edited to write the measured data in a CSV file, i.e. as the script actions were monitored, CPU and DRAM energy measurements were collected: 10 per second, according to the RAPL frequency of 100ms sampling rate [8].

One of the requirements of RAPL (in addition to the CPU architecture and execution on a Linux based system), is the necessary execution as a superuser. The typical method of using RAPL is to directly instrument the source code with the measurement API. In this case, due to security related issues, the Selenium browser scripts could not be executed as superuser. To work around this,

[3] https://gs.statcounter.com/browser-market-share/desktop/worldwide
[4] https://www.w3schools.com/browsers
[5] https://blogvisaodemercado.pt/category/vm-extra/
[6] https://pplware.sapo.pt
[7] https://www.rtp.pt/noticias/mundo/covid-19-a-situacao-ao-minuto-do-novo-coronavirus-no-pais-e-no-mundo_e1220291
[8] https://www.youtube.com/watch?v=LXb3EKWsInQ
[9] https://www.facebook.com/diariodesportivo.ojogo/
[10] https://www.instagram.com/portugal/
[11] https://www.instagram.com/stories/highlights/17851326481891753/
[12] https://www.youtube.com/?gl=PT
[13] https://www.youtube.com/watch?v=LXb3EKWsInQ
[14] https://www.youtube.com/watch?v=-xiXnir2Isg
[15] https://www.twitch.tv/esl_csgo
[16] https://www.youtube.com/c/ESLCS

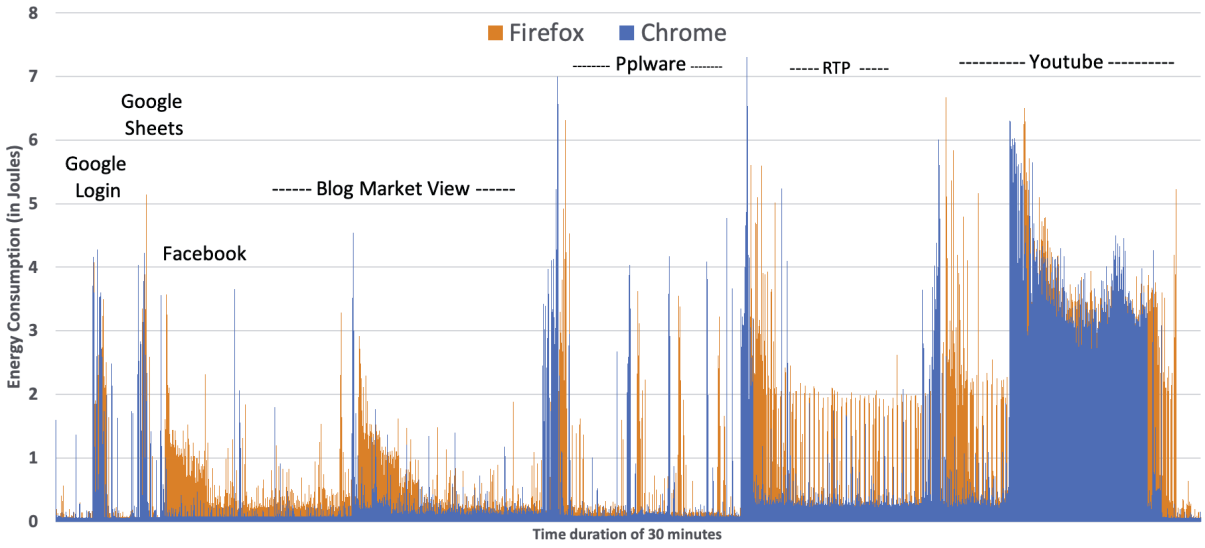


**Figure 1: Energy consumed by Chrome and Firefox.**

RAPL was executed on a separate thread to our browser testing script, allowing us to still collect energy readings.

In order to collect consistent data, and reduce effects from cold starts, warm ups, and cache effects, each script was executed 10 times. From these 10 measurements, we additionally removed the 20% highest and lowest values as to reduce outliers from such issues.

All measurements were performed in the same machine[17]: Linux Ubuntu Desktop 18.04 operating system, with 16GB of RAM, Intel® Core™ i7 8750H 2.2 GHz Maximum Boost Speed 4.1 GHz.

## 3 ANALYSIS AND DISCUSSION

This section will analyze and discuss the results of our study. The main focus is to understand the energy consumed by browsers and how their behavior, in terms of energy consumption, changes on different websites. Additionally, we will show how energy consumed by random-access memory (RAM) and central processing unit (CPU) can be different.

### 3.1 Results

For a better understanding of the results obtained by our tests, we display some graphical visualizations of the energy consumed by browsers. Note that not all the generated graphics will be present within this paper, but are all accessible on our repository [18].

In Figure 1, we see the results of consumed energy/Joules (Y-axis) from our global usage script from start to finish (X-axis). The orange bars are the Firefox's results, while the blue bars are the results when using Chrome. Throughout the timeline, the website names, which were visited, are presented above each corresponding time slot. Additionally, the consumed Joules is the sum of the energy consumed by both DRAM and CPU at each moment of the script, as reported by RAPL. To have a better perception and comparison, Figure 2 represents the total energy consumed by each browser at the end of the entire global script.

Focusing on one website at a time, the bar graphs shown in Figure 3 show the mean amount of energy each browser consumed on each site, for DRAM, CPU, and the sum of both respectively. For each website was created a single script that simulates usage on the specific website. Each of the website measurements are based on the scripts described in Section 2.1.

Finally, in order to have a more detailed view of the results, we present in Figures 4 and 5 the previous results in a boxplot format. This representation is a standardized way of displaying the distribution of data based on a five number summary ("minimum", first quartile, median, third quartile, and "maximum"). It can tell about outliers and what their values are. It can also tell if data is symmetrical, how tightly data is grouped, and if and how data is skewed. While Figure 4 focuses specifically on YouTube for both browsers, and across CPU, DRAM, and their combination, Figure 5 details the combination of DRAM and CPU across both browsers for Instagram, Facebook, and Twitch respectively.

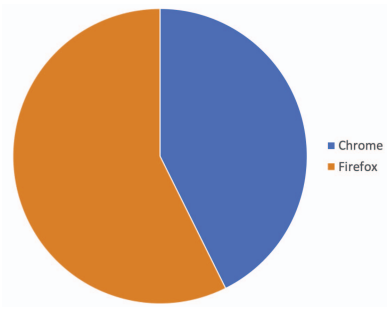


**Figure 2: Measurement of the total consumed energy by Chrome and Firefox based on the global usage script.**

[17] https://www.asus.com/pt/Laptops/ASUS-VivoBook-Pro-15-N580GD/
[18] https://github.com/Energywar/EnergyWar

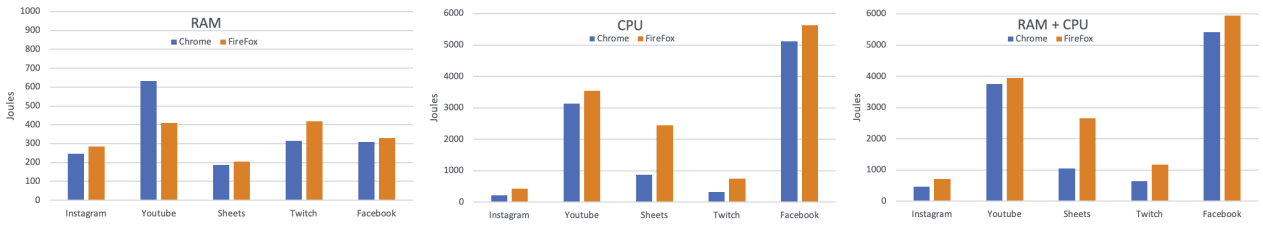


**Figure 3: Mean of consumed energy by RAM, CPU and Ram plus CPU, respectively.**

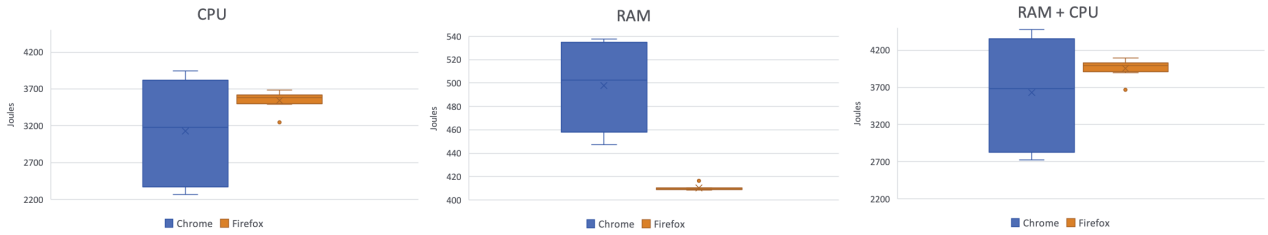


**Figure 4: Distribution of energy consumed on YouTube.**


**Figure 5: Distribution of energy consumed by RAM+CPU of Chrome and Firefox on different websites.**

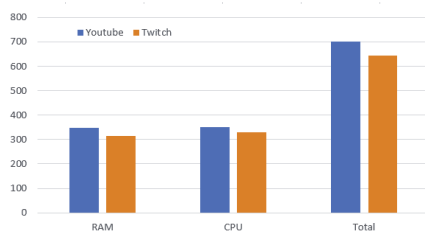


**Figure 6: Distribution of energy consumed in Chrome by Streaming Video websites.**

## 3.2 Discussion

Beginning with Figure 1 we can begin observing some important aspects. For Chrome, it seems to show that the biggest peaks of energy consumption occur when a new tab is opened. In the case of Firefox, it has a lower peak value of energy consumption, but the duration of energy consumption is longer. Additionally, we can also observe that it is slower than Chrome, whereas when Chrome finishes processing the global usage script, Firefox still has more processing to do.

This difference becomes more evident in Figure 2, where we can see that Firefox, globally, consumes more energy than Chrome. This leads us to conclude the answer for **RQ4** to seem to begin indicating that Chrome is the most energy efficient browser in an overall scenario.

Our previous conclusion, that Chrome seems to be the most energy efficient alternative, continues to be supported in Figure 3 where 14 out of 15 cases (across DRAM, CPU, and DRAM+CPU energy consumption) show Chrome to be more energy efficient. The aspect which stands out to show Firefox as more energy efficient is DRAM energy consumption when interacting with YouTube. This can be seen in more detail in Figure 4, with Chrome having a mean DRAM energy consumption value of approximately 500 Joules, with Firefox using approximately 410 joules. When looking back at **RQ1**, the results seem to indicate that on average Chrome

is more energy efficient for browsing YouTube (Figure 3). When carefully observing Figure 4 however, we can see that Firefox is more consistent in terms of energy consumption, while Chrome can in fact achieve higher energy consumption at times. In this case, a more thorough analysis in this scenario should be performed.

Another aspect that needs to be mentioned is the energy consumption in Sheets, where the difference between browsers is accentuated, and Chrome is again more efficient.

In Figure 5 we can see the clear distinction of the results between the two browsers, for our two social-media and live streaming sites, respectively. In our previous case (YouTube), we observed that some overlapping occurred between the energy measurements for Chrome and Firefox. This observation however does not occur for Instagram, Facebook, nor Twitch, where we see that Chrome is clearly more energy efficient in all 3 cases. This difference is even more drastic when visiting Twitch, where the energy consumed by Firefox is almost double. With this, we can conclude that our initial results show that for **RQ2** and **RQ3**, Chrome is definitely the more energy efficient browser for visiting such online platforms.

While Chrome seems to show very promising results, we also observe that it tends to have the biggest variation, as shown in Figure 4 and when browsing Facebook in Figure 5. This is opposite as to what occurs with Firefox, where the results are very consistent.

Finally, when observing live streaming on YouTube or Twitch on Chrome, shown in Figure 6, we see that both sites are very similar, but Twitch is shown to be more energy efficient.

## 4 THREATS TO VALIDITY

The goal of our study was to measure the energy consumption of browsers in different environments and compare the results to know which browser is more efficient. We present in this subsection some threats to the validity of our study.

*Conclusion Validity.* From our experiment, it is clear that *Chrome* and *Firefox*, while performing the same tasks, have a very different impact on energy consumption and time. We also see interesting cases where Chrome is the most efficient browser, yet not the most consistent. For a better comparison, we not only measured CPU energy consumption but also DRAM energy consumption.

This allowed us to not only look at processing energy consumption (CPU based), but also memory energy consumption (DRAM based), which is something well known to be a very important resource for browsers. Additionally, the way we select the scripts is how we felt is a typical user behavior in browser usage.

*Internal Validity.* This category concerns itself with what factors may interfere with the results of our study. We faced a problem where in order to run RAPL, we were required to run as a super user. On the other hand, to execute the Selenium scripts with our browsers, we could not do so as a super user due to safety measures in the browsers. Our solution was to launch a thread which would monitor all the energy consumed during the scripts time-cycle. As to maintain a preciseness in measurements, we placed the system in a state running only the minimum needed processes, and shut down processes such as Bluetooth/WiFi and close unnecessary background programs. In addition, our measurement framework would open a CSV file and write the energy measurements within it. This can be seen as an overhead during measurements. Nevertheless, this overhead is a constant between every single browser/usage scenario. As we focused on comparing the browsers and not only looking at the direct measurement values, this consistent overhead can be considered non-important.

*Construct Validity.* Two browsers have been analyzed in different environments, with each script tested 10 times. The scripts are the same for each browser and all of them were tested the same way with the same rules. Although a few were not able to produce results such as the case of YouTube live streams in Firefox, where Firefox had incompatibility issues with HTML5, those were not considered in the discussion of this article. In terms of streams, we could observe in Figure 6 which stream websites are more efficient in Chrome.

*External Validity.* We concern ourselves with the generalization of the results. The obtained solutions were the best performing ones at the time we set up the study. Measurements in different systems, might produce slightly different resulting values if replicated. We believe these results can be further generalized, and other researchers can replicate our methodology for future work.

## 5 RELATED WORK

Over the last few years, the importance of energy efficiency of software has increased significantly. In reality, studies have arisen with several goals and areas with the vision of understanding how aspects of growth affect energy consumption in diversified software systems, such as Web Browsers.

There are several studies focused on web browsers. They are sophisticated and crucial software systems, and millions of users extend their browsers to customize their browsing experience, so, extensions constitute an important facet of web applications [12]. In modern Web applications, style formatting and layout calculation often account for a substantial amount of local Web page processing time. Web pages caching pay an important role in improving the web page load times. Smart style caching and layout caching are two novel techniques used by browsers to improve their performance [28]. Mobile page load times are an order of magnitude slower compared to non-mobile pages. It is not clear what causes the poor performance: the slower network, the slower computational speeds, or other reasons. Most Web optimizations are designed for non-mobile browsers and do not translate well to the mobile browser. Towards understanding mobile Web page load times, is performed an in-depth pairwise comparison of loading a page on mobile versus a non-mobile browser, and characterized the bottlenecks in the mobile browser vis-a-vis non-mobile browsers [17].

Focusing more on energy analysis, several studies intend to provide conclusions of the energy impact of different implementation decisions. Understand how the battery of the mobile device can last up to approximately an extra hour if the applications are developed with energy-aware practices [6]. Compare the energy efficiency of similar programs where the goal is to provide developers with techniques and tools to reason about the energy consumption of all products in a SPL, without having to produce, run and measure the energy in all of them [4]. In spite of similar functionality, mobile apps may present very different energy costs, due to the

choices made in their design and construction [11]. or even try to understand and relate time and energy efficiency between multiple alternatives [1, 14, 18, 20–22, 27].

Specifically on a mobile environment, there are works analyzing the energy efficiency of code blocks [5, 13], the energy impact of different virtual keyboards [25], monitoring how energy consumption evolves [9], how browser extensions affect browser performance [2], and analyzing the characteristics of web browsers which cause energy inefficiency in EAS enabled mobile devices [3].

Despite the interest in the area of energy consumption, we couldn't find any related work comparing the energy consumption behavior of different web browsers. With this study, we aim to be a basis for exploring the subject in more detail, such as comparing with other web browsers by providing evidence that there is a difference between web browsers performing multiple tasks in terms of energy consumption.

## 6 CONCLUSION

In this paper, we first present an initial analysis and comparison of the consumed energy by Google Chrome and Mozilla Firefox. It was possible to see which browser tends to be the most energy efficient.

We also observed how different websites have varied results in regards to energy consumption, and how these two different browsers act differently on certain websites. We were also able to see the energy consistency of browsers and discover that, while Chrome is the most energy efficient alternative of the two, it can also be the most energetically inconsistent and in certain scenarios, have the worse results.

With this analysis, we can begin to conclude that Google Chrome is a better browser than Mozilla Firefox in terms of energy efficiency, however, Mozilla Firefox is more consistent.

For future work, we are extending this study to explore more web sites and web browsers, such as Safari and Edge - it is a matter of time for Edge be compatible with Ubuntu - , and also consider GPU energy consumption. Additionally, by extending this initial study, we will obtain a more robust testing bed which will allow us to properly provide a thorough and complete statistical analysis between the web platforms and browsers, and thus present finalize conclusions for our research questions.

## ACKNOWLEDGEMENTS

This work is financed by National Funds through the Portuguese funding agency, FCT -Fundação para a Ciência e a Tecnologia within project UIDB/50014/2020.

## REFERENCES

[1] Sarah Abdulsalam, Ziliang Zong, Qijun Gu, and Meikang Qiu. 2015. Using the Greenup, Powerup, and Speedup metrics to evaluate software energy efficiency. In *Proc. of the 6th Int. Green and Sustainable Computing Conference*. IEEE, 1–8.

[2] Kevin Borgolte and Nick Feamster. 2020. Understanding the Performance Costs and Benefits of Privacy-Focused Browser Extensions. In *Proceedings of The Web Conference 2020 (WWW '20)*. Association for Computing Machinery, New York, NY, USA, 2275–2286. https://doi.org/10.1145/3366423.3380292

[3] Yonghun Choi, Seonghoon Park, and Hojung Cha. 2019. Optimizing Energy Efficiency of Browsers in Energy-Aware Scheduling-Enabled Mobile Devices. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom '19)*. Association for Computing Machinery, New York, NY, USA, Article 48, 16 pages. https://doi.org/10.1145/3300061.3345449

[4] Marco Couto, Paulo Borba, Jácome Cunha, João P. Fernandes, Rui Pereira, and João Saraiva. 2017. Products go Green: Worst-Case Energy Consumption in Software Product Lines. (2017).

[5] Marco Couto, Tiago Car cão, Jácome Cunha, João Paulo Fernandes, and João Saraiva. 2014. Detecting Anomalous Energy Consumption in Android Applications. In *Programming Languages (Lecture Notes in Computer Science)*, Fernando-Magno Quintão Pereira (Ed.), Vol. 8771. Springer Int. Publishing, 77–91.

[6] Luis Cruz and Rui Abreu. 2017. Performance-based Guidelines for Energy Efficient Mobile Applications. In *Proceedings of the 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft '17)*. IEEE Press, 46–57.

[7] Wellington de Oliveira Júnior, Renato Oliveira dos Santos, Fernando José Castor de Lima Filho, Benito Fernandes de Araújo Neto, and Gustavo Henrique Lima Pinto. 2019. Recommending energy-efficient Java collections. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 160–170.

[8] Martin Dimitrov, Carl Strickland, Seung-Woo Kim, Karthik Kumar, and Kshitij Doshi. 2015. Intel® Power Governor. https://software.intel.com/en-us/articles/intel-power-governor. Accessed: 2017-10-12.

[9] Fangwei Ding, Feng Xia, Wei Zhang, Xuhai Zhao, and Chengchuan Ma. 2012. Monitoring Energy Consumption of Smartphones. *CoRR* (2012).

[10] Marcus Hähnel, Björn Döbel, Marcus Völp, and Hermann Härtig. 2012. Measuring energy consumption for short code paths using RAPL. *SIGMETRICS Performance Evaluation Review* 40, 3 (2012), 13–17.

[11] R. Jabbarvand, A. Sadeghi, J. Garcia, S. Malek, and P. Ammann. 2015. EcoDroid: An Approach for Energy-based Ranking of Android Apps. In *Proc. of 4th Int. Workshop on Green and Sustainable Software (GREENS '15)*. IEEE Press, 8–14.

[12] Benjamin S. Lerner and Dan Grossman. 2010. Language Support for Extensible Web Browsers. In *Proceedings of the 2010 Workshop on Analysis and Programming Languages for Web Applications and Cloud Applications (APLWACA '10)*. Association for Computing Machinery, New York, NY, USA, 39–43. https://doi.org/10.1145/1810139.1810146

[13] Ding Li, Shuai Hao, William GJ Halfond, and Ramesh Govindan. 2013. Calculating source line level energy information for android applications. In *Proc. of the 2013 Int. Symposium on Software Testing and Analysis*. ACM, 78–89.

[14] Luís Gabriel Lima, Gilberto Melfe, Francisco Soares-Neto, Paulo Lieuthier, João Paulo Fernandes, and Fernando Castor. 2016. Haskell in Green Land: Analyzing the Energy Behavior of a Purely Functional Language. In *Proc. of the 23rd IEEE Int. Conf. on Software Analysis, Evolution, and Reengineering (SANER'2016)*. IEEE, 517–528.

[15] Luís Gabriel Lima, Francisco Soares-Neto, Paulo Lieuthier, Fernando Castor, Gilberto Melfe, and João Paulo Fernandes. 2019. On Haskell and energy efficiency. *Journal of Systems and Software* 149 (2019), 554–580.

[16] Kenan Liu, Gustavo Pinto, and Yu David Liu. 2015. Data-oriented characterization of application-level energy optimization. In *Fundamental Approaches to Software Engineering*. Springer, 316–331.

[17] Javad Nejati and Aruna Balasubramanian. 2016. An In-Depth Study of Mobile Browser Performance. In *Proceedings of the 25th International Conference on World Wide Web (WWW '16)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1305–1315. https://doi.org/10.1145/2872427.2883014

[18] Wellington Oliveira, Renato Oliveira, and Fernando Castor. 2017. A study on the energy consumption of Android app development approaches. In *Proceedings of the 14th International Conference on Mining Software Repositories*. IEEE Press, 42–52.

[19] Rui Pereira, Tiago Carção, Marco Couto, Jácome Cunha, João Paulo Fernandes, and João Saraiva. 2020. SPELLing out energy leaks: Aiding developers locate energy inefficient code. *Journal of Systems and Software* 161 (2020), 110463.

[20] Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, and João Saraiva. 2017. Energy Efficiency across Programming Languages: How Do Energy, Time, and Memory Relate?. In *Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering (SLE 2017)*. Association for Computing Machinery, New York, NY, USA, 256–267. https://doi.org/10.1145/3136014.3136031

[21] Rui Pereira, Marco Couto, João Saraiva, Jácome Cunha, and João Paulo Fernandes. 2016. The Influence of the Java Collection Framework on Overall Energy Consumption. In *Proc. of the 5th Int. Workshop on Green and Sustainable Software (GREENS '16)*. ACM, 15–21.

[22] Gustavo Pinto, Fernando Castor, and Yu David Liu. 2014. Understanding energy behaviors of thread management constructs. In *Proc. of the 2014 ACM Int. Conf. on Object Oriented Programming Systems Languages & Applications*. ACM, 345–360.

[23] Behnam Pourghassemi, Ardalan Amiri Sani, and Aparna Chandramowlishwaran. 2019. What-If Analysis of Page Load Time in Web Browsers Using Causal Profiling. *Proc. ACM Meas. Anal. Comput. Syst.* 3, 2, Article 27 (June 2019), 23 pages. https://doi.org/10.1145/3341617.3326142

[24] Efraim Rotem, Alon Naveh, Avinash Ananthakrishnan, Eliezer Weissmann, and Doron Rajwan. 2012. Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge. *IEEE Micro* 32, 2 (2012), 20–27.

[25] Rui Rua, Tiago Fraga, Marco Couto, and João Saraiva. 2020. Greenspecting Android Virtual Keyboards. In *2020 IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*.

[26] Deyu Tian and Yun Ma. 2019. Understanding Quality of Experiences on Different Mobile Browsers. In *Proceedings of the 11th Asia-Pacific Symposium on Internetware (Internetware '19)*. Association for Computing Machinery, New York, NY, USA, Article 1, 10 pages. https://doi.org/10.1145/3361242.3361249

[27] Anne E. Trefethen and Jeyarajan Thiyagalingam. 2013. Energy-aware software: Challenges, opportunities and strategies. *Journal of Computational Science* 4, 6 (2013), 444 – 449.

[28] Kaimin Zhang, Lu Wang, Aimin Pan, and Bin Benjamin Zhu. 2010. Smart Caching for Web Browsers. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. Association for Computing Machinery, New York, NY, USA, 491–500. https://doi.org/10.1145/1772690.1772741