Dirty Electrons: On the Carbon Intensity of Stored Energy

Robin Ohs*, Henry Janson*, Andreas Schmidt*, Luis Gerhorst†‡, Benedict Herzog† and Timo Hönig†
*Saarland University – Computer Science, Saarland Informatics Campus, Saarbrücken, Germany
†RUB – Ruhr-Universität Bochum, Bochum, Germany
‡FAU – Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany

***Abstract*—To reduce carbon emissions, system software must accurately trace and properly attribute these emissions to applications that cause energy consumption and hardware degradation. While the carbon intensity of the grid energy is readily available from online services, it is often overlooked that computing systems do not directly and immediately consume this energy. Instead, energy coming from the grid is first transformed by power supplies (PSUs) and then often stored in batteries.**

In this work, we investigate how PSUs and batteries affect the carbon intensity of the energy that actually powers the computer system. We analyze the extent to which specific hardware characteristics need to be taken into account in order to accurately attribute operational and embodied emissions to applications. To do this, we propose an analytical model for the carbon intensity of stored energy, which includes power supply and charging efficiency (which inflates operational emissions) and battery degradation (which causes embodied emissions). We conclude with a look at the future of system-level management of emissions caused by stored energy.

I. Introduction

With the decarbonization of important sectors such as transport and industry, the IT sector will become one of the largest producers of carbon emissions [1]. In order for manufacturers and software to be able to report, adapt to, and reduce carbon emissions that contribute to the global warming potential (GWP), one necessary prerequisite is to know the carbon intensity of the electricity currently used by the system. A typical approach for this problem is to query external providers such as [2], [3] for the current grid intensity. This can be done either directly, with the carbon-aware-sdk [4], or with the help of an operating system (OS) service such as carbond [5]. While this approach is a first step, two major sources of inefficiency are overlooked.

First, more than 15 years since the advent of smartphones, computing in general has long advanced beyond the *scope of power sockets*. Users heavily rely on mobile devices and Internet-of-Things nodes embedded into their environment. Most of these devices require the use of batteries to store energy for operation. However, it is often overlooked that these batteries have embodied emissions and conceptually store the carbon emissions that come from the electricity used to charge the batteries [6], [7] with certain inefficiencies that amplify the intensity. Secondly, the current used for operation or charging is direct current (DC), whereas the intensity specified by external providers refers to the alternating current (AC) of the grid, which must first be converted. The impact of those inefficiency sources raises the question of the *actual system carbon intensity of mobile systems* that is considered in carbon-aware decisions. This is because the intensity of the energy discharged from the battery could depend on different grid intensities during the charging process in the past.

In general, the carbon intensity of a system is computed using the current *operational* emissions O caused by the energy consumption of the hardware, together with a share of the *embodied* emissions M originating in hardware production. The embodied emissions per unit of work can be calculated from the expected lifespan of the component, which is for batteries usually given as recharge *cycle counts*. The *cycle count* describes the expected number of possible charging cycles before the battery degrades. A battery is considered degraded if the capacity of the battery drops below 80% of its maximum capacity [8], [9]. Besides degradation, batteries are imperfect components and subject to inefficiencies during charging and discharging (*charging efficiency*).

The usage of a battery therefore leads to both operational overheads and embodied emissions that are essential when calculating the carbon emissions and created GWP of a mobile computer system. In our analysis, we consider both scenarios: purely battery-powered operation and the operation with external power supply, which is also used for battery charging. Thereby, our model also incorporates the inefficiency of the power supply unit (PSU), necessary for the conversion of the AC current of the power grid into DC current used by the system.

The contributions of this paper are three-fold:

- We analyze and model the properties of PSUs and batteries that must be taken into account in order to accurately track the energy's actual carbon intensity.
- We propose policies to attribute emissions to applications that enable effective application- and system-level optimizations. This is important because, eventually, applications cause energy consumption and hardware degradation, even though both are hardware properties [10]–[13].
- We extend the publicly available system daemon carbond [5], with prototypical emission tracing and attribution on battery-powered Linux systems. Based upon this, we discuss necessary future steps (e.g., hardware extensions) to enable more accurate tracing of stored energy and its GWP.

II. System Model

Before we dive into the practical carbon attribution and technical implementation, we provide theoretical models for the energy and emissions inflicted by both PSUs and batteries.

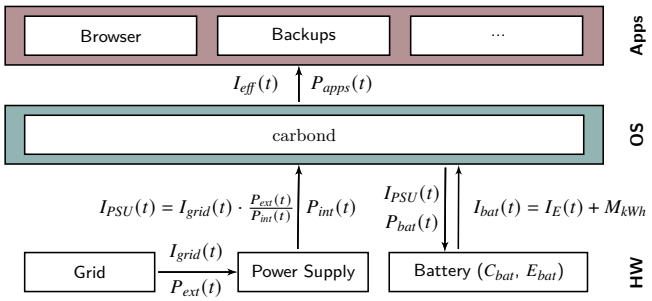

Figure 1. The operating system (OS) together with carbond [5] abstracts power supply hardware (HW) information and provides the effective carbon intensity I_{eff} to applications. I_{bat} is computed by carbond as the accumulated emissions divided by the accumulated battery charge (incorporating common inefficiencies) and a share of the embodied emissions from battery production.

We treat *operational and embodied emissions* differently and relate these to a *rate* (often kWh)—following the methodology of the *Software Carbon Intensity* (SCI) [14] ISO standard.

A. Power Supply Units

The power grid operates on AC, while electric devices operate on DC creating the need for hardware components to convert the energy. Those components are known as PSUs and responsible for drawing power from the grid, converting it, and delivering it to the system. These steps are illustrated in the lower left part of Fig. 1.

Due to heat generation, capacitor resistances, and other losses, PSUs have a certain efficiency that determines the AC to DC conversion ratio. Modern PSUs can achieve efficiencies from 80% up to 96%, depending on the current load in ratio to their maximum load. Internal PSUs may be certified according to *80 PLUS* [15]. For example, a 1000-Watt PSU rated at Platinum with a load of 50% = 500 W must at least have an efficiency of 94%. In contrast to that, battery-driven mobile systems typically rely on external PSUs which are typically not certified by *80 PLUS*. However, based on tests and regulations, it can be assumed that the average energy efficiency of external power supplies should be above 80% [16], [17].

This conversion ratio results in more energy being drawn from the grid than can ultimately be used or measured (e.g., with RAPL [18], [19]) internally by the system. Thus, when calculating the *actual system carbon intensity* of the electricity, it is important to consider the energy lost in the system resulting from this inefficiency. Otherwise, the outcome will be distorted (too low), and the use of less efficient components will not be penalized, despite the fact that they draw more energy and thus contribute to higher carbon emissions (higher GWP).

One potential solution is to compensate for the inefficiency by increasing the carbon intensity of the energy drawn from the PSU by an amount that accounts for the energy difference between the energy drawn from the grid and the energy drawn from the PSU. In this context to calculate the augmented intensity, we define P_{int} as the internal power drawn from the PSU by the system, while P_{ext} is the power drawn from the grid before conversion (cf. lower middle of Fig. 1). The efficiency of a PSU (Eff_{psu}) at time t can be calculated by dividing the load P_{int} on the PSU by the external power P_{ext} drawn by the PSU, as $Eff_{psu}(t) = \frac{P_{int}(t)}{P_{ext}(t)}$.

P_{int} is the sum of the power used by all running processes (P_{apps}) and the power used for charging the battery (P_{bat}), which could be negative during discharge or zero for non-battery systems, as shown below:

$$P_{int}(t) = P_{apps}(t) + P_{bat}(t)$$

Given the grid intensity (I_{grid}), it is possible to calculate the intensity after conversion (I_{PSU}) for time t as follows:

$$I_{PSU}(t) = \frac{I_{grid}(t)}{Eff_{psu}(t)} = I_{grid}(t) \cdot \frac{P_{ext}(t)}{P_{int}(t)}$$

Looking at Fig. 1, it becomes evident that this improved approach to differentiate between I_{PSU} and I_{grid} by considering the conversion efficiency for internal intensities does not solely apply to battery-driven systems. Rather, all systems that are operated with DC and connected to an AC source are affected.

B. Batteries

While battery physics are well-understood analytically, software systems often use overly simplistic models to predict their behavior. A comprehensive modeling of the carbon emissions stored in batteries and the flow of energy in a system can help identify inefficiencies and optimization hotspots to reduce the overall emitted emission [6], [20].

The following subsections identify inefficiencies that must be considered when calculating operational and embodied intensities for battery-powered systems. These inefficiencies, analogous to the PSU, augment the intensity of the electricity ultimately drawn from the battery. Furthermore, they are connected to the AC-DC conversion inefficiency, since a battery is charged using DC current.

1) Battery Degradation: Among computing hardware components, batteries are well-known for having a limited lifespan due to their performance degradation. Degradation is a chemical process over time when the capacity and power of a battery are getting smaller and internal resistances increase [21].

There is no fixed physical threshold at which a battery is considered degraded, rather this is a technical decision that varies on the battery and its use case. For instance, Apple iPhones will display a warning that *the battery is significantly degraded* when the maximum capacity falls below 80%, which is a widely adopted threshold [8], [9]. Degradation is non-linear and depends on external influences such as the operating temperature, the state of charge, and the load used for charging and discharging [8], [22]. A metric called *C-rate* is commonly used to classify the current load level on a battery. The *C-rate* states how fast a battery is charged or discharged in ratio to its maximum nominal capacity [8], [23]. For example, a 2 mAh battery can supply 2 mA for 1 hour (*C-rate* = 1) or 1 mA for 2 hours (*C-rate* = 0.5). Furthermore, the loads can also exceed the battery rating (*C-rate* > 1). Exceeding the rating places more stress on the battery, which can result in higher degradation potential compared to lower loads and less energy efficiency.

Although the mechanisms responsible for degradation are well-known, their interplay remains poorly understood [8], [21], [22]. Consequently, it is challenging to calculate the aforementioned effects precisely until they can be measured or predicted technically. For now, a charging-cycle–based approach may be employed as a provisional measure to incorporate the embodied emissions of a battery into the calculation of internal carbon intensities. This is supported by the average number of charging cycles a battery can undergo before the capacity is expected to fall below 80%, which is typically provided by the manufacturer. Modern Lithium-Ion batteries can withstand more than 1000 cycles [23]. With that, if the total emissions caused by the production of the battery are known, the attributable embodied GWP of a full battery cycle can be estimated. The production of Lithium-Ion batteries is, among other things, energy intensive. For instance, the battery of a *Fairphone 3* smartphone has a normalized embodied GWP of 133.33 $\mathrm{kgCO_2e/kWh}$ [24]. For comparison, we look at batteries of electric vehicles that embody normalized emissions of 61 to 106 $\mathrm{kgCO_2e/kWh}$, depending on the carbon intensity of the electricity used in manufacturing [25]. In summary, larger batteries benefit from scaling effects, such that the normalized production emissions are dependent upon the size of the battery, but the approximate range in which the embodied part is located is about 100 $\mathrm{kgCO_2e/kWh}$ [23]. One development, which will support the calculation of embodied emissions for batteries in the future, is the introduction of passports for batteries, for instance in the European Union, allowing for the tracing of battery footprints and life cycles [26].

A battery with normalized production emissions of Costs_{bat} $[\frac{\mathrm{kgCO_2e}}{\mathrm{kWh}}]$ and manufacturer specified average possible charging cycles before degradation of Cycles_{bat}, yields estimated emissions per withdrawn unit of energy of:

$$M_{kWh} = \frac{\mathrm{Costs}_{bat}}{\mathrm{Cycles}_{bat}}$$

2) Charging Efficiency: Lithium-Ion batteries are the most commonly used battery type for mobile systems, but suffer from inefficiencies during (dis)charging. Together with the AC-DC conversion inefficiency and the embodied emissions of the battery, a third source of emissions must therefore be taken into account, namely the efficiency of the battery during operation.

A model for battery efficiency is based on the *Coulombic Efficiency* (Eff_{CE}), which is close to 100% for LI batteries, and the *Energetic Efficiency* (Eff_{EE}), which is based on the *C-rate* used to charge and discharge the battery. Lowering the *C-rate* can increase the efficiency because less heat is generated [27]. For example, a *C-rate* above 0.5 lowers the efficiency to less than 97% [28]. According to [29], charging a battery with 1 C and discharging with 0.5 C, yields a total efficiency of 91.4%. These rates approximately match charging an iPhone with a fast-charger and running a computation-heavy task afterwards. Thus, the efficiency of the battery (Eff_{bat}) depends on the current *C-rate* (Cr_{bat}) at time t. Therefore:

$$Eff_{bat}(t) := Eff_{bat}(Cr_{bat}(t))$$

The *C-rate* dependence has important additional implications for attributing carbon to applications. If multiple applications run in parallel, their collective power consumption results in a higher *C-rate* and consequently to lower battery efficiency. Thus, the carbon footprint of each running app is augmented, creating an opportunity for the implementation of scheduling mechanisms that distribute the power consumption in order to maintain the battery efficiency at a high level.

To calculate the intensity of the energy in the battery (I_E) at time t during charging, it is necessary to know the amount of energy (E_{bat}) and the accumulated carbon emissions in the battery (C_{bat}) at time t. For E_{bat}, the load on the battery is multiplied with the efficiency of the battery Eff_{bat}. However, the formula for C_{bat} does not include the battery efficiency, as it is necessary to account for the emissions of the energy difference. Thus, the formulas for charging are given as:

$$E_{bat}(t) = E_0 + \int_0^t P_{bat}(t) \cdot Eff_{bat}(t)\,\mathrm{d}t$$
$$C_{bat}(t) = C_0 + \int_0^t P_{bat}(t) \cdot I_{PSU}(t)\,\mathrm{d}t$$

During discharge (P_{bat} is negative), the formulas change as the inefficiency reduces the usable power extracted from the battery and therefore more energy must be extracted to achieve the requirements of P_{bat}, which is illustrated as follows:

$$E_{bat}(t) = E_0 + \int_0^t \frac{P_{bat}(t)}{Eff_{bat}(t)}\,\mathrm{d}t$$
$$C_{bat}(t) = C_0 + \int_0^t \frac{P_{bat}(t)}{Eff_{bat}(t)} \cdot I_{eff}(t)\,\mathrm{d}t$$

For C_{bat} an artificial value I_{eff} instead of I_{PSU} is used, as there is no physical rule how the intensity of the energy that leaves the battery must be selected by a system service like carbond.

In theory, this value can be freely chosen as long as the sum of the amount of emissions remaining in the battery and those attributed to the applications remains constant (see later for pragmatic bounds).

Using the formulas above, the battery's energy intensity can then be calculated as:

$$I_E(t) = \frac{C_{bat}(t)}{E_{bat}(t)}$$

C. Discussion

Our analysis shows that the physical properties of PSUs and batteries must be taken into account when calculating internal power intensities. Relying only on external data providers such as [2], [3] for carbon-aware decisions is not sufficient, as they only provide a lower bound on the actual system intensity. In Fig. 2, we see that PSUs draw more energy from the grid than is ultimately delivered to the system. This increases the intensity of the energy used by the system and also the intensity of the energy used to charge the battery. Furthermore, batteries need to be considered not only for their embodied emissions, which can have a significant impact, as illustrated in Fig. 2 and related work [23], but also for their load-dependent inefficiencies during

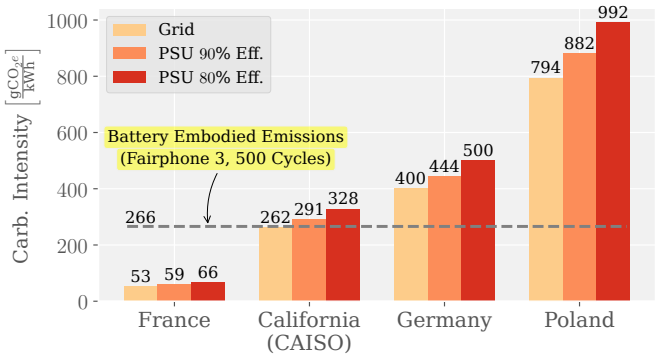

Figure 2. The original energy's intensity, the PSU's inefficiency as well as the battery's embodied emissions play a central role in the actual intensity of stored energy. The countries were selected as examples of best, mid and worst-case based on their average grid carbon emission in 2023 [2].

charging and discharging [30]. In the following section, we design an attribution policy that holds applications, which make heavy use of these carbon-intensive components, accountable.

III. Carbon Attribution

Attributing physical resources (e.g. energy or carbon emissions) to software entities (e.g. processes) requires a holistic view of the system. In particular, it is crucial to consider both the executed software and the resource demand of the hardware. Related work dealing with the attribution and management of energy consumption to software therefore has been situated at the operating-system level, where both resource demand and software can be monitored [10]–[13]. Analogous to these works, this chapter presents approaches and policies to attribute the carbon emissions caused by the utilization of a battery to the corresponding software—using the battery model from Sec. II.

A. Carbon Intensity of Stored Energy

The battery's intensity (I_{bat}) combines the intensity of the stored energy (I_E) and a portion of the battery's embodied emissions (M_{kWh}):

$$I_{bat}(t) = I_E(t) + M_{kWh}$$

Given a *Fairphone 3* battery with 500 cycles before degradation and 133.33 $\mathrm{kgCO_2e/kWh}$ production emissions [24], the following embodied emissions per unit of energy discharged can be calculated:

$$M_{kWh} = \frac{133.33\ \frac{\mathrm{kgCO_2e}}{\mathrm{kW\,h}}}{500} = 0.266\ \frac{\mathrm{kgCO_2e}}{\mathrm{kW\,h}} = 266\ \frac{\mathrm{gCO_2e}}{\mathrm{kW\,h}}$$

B. Effective Carbon Intensity

For attributing energy demand, sufficient measurement procedures are available [18], [19]. In contrast, attributing *carbon demand* is flexible, as it is a pure *accounting quantity*—a computer system (almost) never emits carbon by itself. Instead, it accounts carbon emissions, because all the figures we get are estimated far beyond our system, that is, embodied emissions in a Life-Cycle Analysis (LCA) and operational emissions from grid meta information. However, for a pragmatic attribution

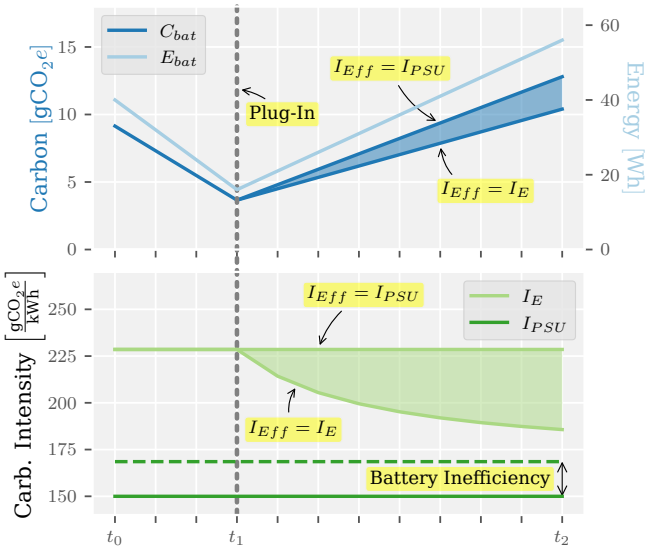

Figure 3. The choice of effective intensity (I_{eff}) given to applications has a significant impact on the carbon stored in the battery (C_{bat}) and hence on the development of the battery's intensity (I_E).

system, the flows of both carbon and energy within the system must be consistent, that is, no energy or carbon is lost without reason and proportions are maintained.

A well-designed attribution policy provides incentives for carbon optimizations by system and application designers. These optimizations include the following:

- *Energy Efficiency:* Save carbon emissions by saving energy, for instance, by running latency-insensitive tasks at a more energy-efficient CPU frequency.
- *Temporal Shifting:* Shift latency-insensitive application tasks to times at which low-carbon energy is directly available from the grid (i.e., the device is plugged in).
- *Geographical Shifting:* When applicable, offload tasks to other machines that have low-carbon energy available.

To better understand these flows of carbon and energy, we have to distinguish the following cases and pick the *effective intensity* I_{eff} provided to the application:

a) If the system is *unplugged*, the battery's accumulated energy and carbon is provided to the application $I_{eff} = I_{bat}$.
b) If the system is *plugged in*, I_{eff} is less obvious.

For most systems, being plugged-in means that the system is powered and the battery could be charging (parallel discharging and passing through grid power is unlikely). In Fig. 3, our system first runs on battery, which discharges from t_0 to t_1—reducing both the stored carbon and energy, while the intensity stays constant. At t_1, we plug into a power source that provided cleaner energy than stored in the battery. In the lower half of the diagram, we see that there is a gap between I_{PSU} and the smallest I_E, due to (dis)charging inefficiencies. The technical question is now: ''how quickly do we approach this optimal I_E''. The speed is determined by the choice of I_{eff}, where we have two pragmatic corner cases: a) I_{PSU}: we pass the clean energy through to the application, or b) I_E: we

pretend we still run on battery. The top graph shows how this choice determines the carbon trajectories, while the bottom graph shows the difference between the battery's intensity I_E staying constant and approaching the optimal intensity in a quick and pragmatic fashion (it could be approached faster by using I_{eff} beyond the space between I_E and I_{PSU}).

IV. Carbon Tracing

We first present our implementation and then discuss hardware changes that would improve support for accurate and efficient emissions tracing.

A. Linux-based Prototype

To implement battery emission tracing, we extend carbond with an interface for battery–emission-tracing. Thereby, we already have access to carbond's abstractions for grid carbon intensity and improve it by considering the PSU inefficiency to provide a more realistic system carbon intensity. To track the battery we use data retrieved from the Linux power supply device class abstraction [31]. This guarantees good support for as many different devices as possible. The Linux power supply class system is part of the Linux sysfs and provides data about power supplies. For batteries, it provides data about the charging state, the battery charge and the output voltage. Importantly, not all battery drivers support all these metrics, but in most cases, all the needed data can be derived.

Using carbon data of carbond and battery data from the Linux power supply class, it is now possible to create a model for the intensity of the battery energy I_E. By configuring production costs and expected charging cycles, carbond can calculate an embodied intensity for the battery, that together with I_E gives the battery intensity I_{bat}. In a next step, this allows the implementation of the attribution policies from Sec. III.

B. Hardware Support

We identify multiple instances where extended hardware information could improve our implementation's accuracy. While in the general energy sector, meta information about energy storage is shared [30], for computing, this is only rarely the case. First, current hardware limits us in that precise efficiency curves are often not available both for batteries and PSUs. Ideally, hardware would make this information directly available to the OS. This traditionally requires a separate connection (usually USB) because power-supply standards like ATX [32] do not foresee such information transfers. However, with USB4 being now frequently used for power delivery, this information could be transferred without additional connections. Second, the majority of mobile systems are capable of charging even when powered off. In this situation carbond can not track the carbon intensity of the power used for charging but only interpolate between available points after the next time the service is run. For a more realistic tracing while powered off, it would be sufficient if the hardware would log the power consumption over time, which would allow the energy and carbon to be processed without interpolation. In summary, both accurate efficiency information and tracing while powered off pose unique but solvable challenges to our implementation.

V. Implications

Our analysis shows how PSUs and batteries significantly impact the energy's actual carbon intensity. For example, embodied emissions from a *Fairphone 3* battery are 266 gCO_2e/kWh, while the carbon intensity for the grid might range from 53 to 411 gCO_2e/kWh (I_E for France/USA in 2023 [2]). Battery embodied intensities can not be mitigated by charging with emission-free electricity and they do not even account for environmental hazards [33], [34] not expressible in gCO_2e.

While there exist some techniques that may reduce battery-related inefficiencies and embodied emissions, their expected impact is limited. In order to reduce the impact of the battery, either the production emissions of the battery have to be reduced or the number of cycles before degradation has to be increased. While the first could be achieved by using renewable energy for the production, the second is a complicated topic with much research underway. One example are new *charging protocols*, where the use of pulse current charging instead of constant current charging shows a significant increase in possible cycles before degradation, which leads to a reduction of M_{kWh} and I_{bat} [9]. However, even when this actually doubles the battery's lifespan, the component's impact on overall carbon footprint is still significant.

Recently, *Carbon Explorer* [35] showed that batteries are essential in order to reduce the overall carbon footprint of data centers. Initially, this may appear to conflict with our analysis, which shows that batteries have a devastating effect on the energy's effective carbon intensity. However, this is only because the alternatives described in [35] are much worse. First, power supply from renewable energies is inherently intermittent, even with significant investments. Second, deploying more servers to support aggressive carbon-aware scheduling implies embodied emissions that are much higher than the emissions from batteries. With this in mind, our work highlights that battery use, while necessary, must be limited as much as possible. This is enabled by our attribution policy, which incentives applications to be frugal with their battery-usage.

In Sec. III-B, we opened the design space for carbon attribution strategies when it comes to batteries. While the concrete choice of strategy will depend on concrete system parameters (PSU, battery) and usage patterns (charging behaviour), it also opens up a *fairness* discussion. The first, about *Instantaneous Fairness*, is that at any point in time, all applications should have the same intensity I_{eff} (provided they do give the same guarantees to the OS when it comes to scheduling). The second, is *Long-Term Fairness*, where it is currently unclear how this can even be defined. However, intuitively, we want to incentivize applications that use the OS in a more sustainable way and we do not want applications that are agnostic or even try to trick the system to get low carbon footprint attributed. We believe formalizing both the definition as well as the system model is an interesting research task.

In future work, we will analyze how our conclusions affect software design in detail, and deduce application- and system-level changes to reduce carbon emissions. For example, battery

charging during periods of abundant clean energy in the grid must be reevaluated and it must be analyzed how improved job scheduling can reduce battery-degrading load spikes.

VI. Conclusion

We have presented the main sources of PSU- and battery-related carbon emissions that are relevant to current mobile computing systems. We prototype a system to track battery-related emissions on Linux that enables practical future research and analysis by system- and application-developers. This is mandatory, as our work shows that the physical properties of PSUs and especially batteries have a significant impact on a system's carbon footprint. Our analysis reveals that battery-preserving optimizations are much more important than previously anticipated, and even that the grid's carbon intensity might only be of secondary concern for some systems.

Acknowledgements

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project numbers 389792660 (TRR 248 – CPEC[1]); 465958100 (''NEON''); 502228341 (''Memento''); 502615015 (''ResPECT''); and 539710462 (''DOSS'').

[1] https://perspicuous-computing.science

References

[1] C. Freitag, M. Berners-Lee, K. Widdicks, B. Knowles, G. S. Blair and A. Friday, ''The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations,'' *Patterns*, vol. 2, no. 9, 2021. DOI: 10.1016/j.patter.2021.100340.

[2] Electricity Maps. ''Live 24/7 CO2 emissions of electricity consumption.'' (2024), [Online]. Available: https://app.electricitymaps.com/map (visited on 16/09/2024).

[3] Watttime. ''Access to real-time, forecast, and historical data for electric grids around the world.'' (2024), [Online]. Available: https://watttime.org/ (visited on 16/09/2024).

[4] Green Software Foundation. ''Carbon-aware SDK.'' (2024), [Online]. Available: https://github.com/Green-Software-Foundation/carbon-aware-sdk (visited on 16/09/2024).

[5] A. Schmidt, G. Stock, R. Ohs, L. Gerhorst, B. Herzog and T. Hönig, ''carbond: An operating-system daemon for carbon awareness,'' in *Proceedings of the 2nd Workshop on Sustainable Computer Systems (HotCarbon '23)*, ACM, 2023, 2:1–2:6. DOI: 10.1145/3604930.3605707.

[6] P. Maistriaux, T. Pirson, M. Schramme, J. Louveaux and D. Bol, ''Modeling the carbon footprint of battery-powered IoT sensor nodes for environmental-monitoring applications,'' in *Proceedings of the 12th International Conference on the Internet of Things (IoT '22)*, ACM, 2022, pp. 9–16. DOI: 10.1145/3567445.3567448.

[7] T. Cheng, G. D. Abowd, H. Oh and J. Hester, ''Transient internet of things: Redesigning the lifetime of electronics for a more sustainable networked environment,'' in *Proceedings of the 2nd Workshop on Sustainable Computer Systems (HotCarbon '23)*, ACM, 2023, 18:1–18:8. DOI: 10.1145/3604930.3605723.

[8] Y. Sun, L. Kong, H. Abbas Khan and M. G. Pecht, ''Li-ion battery reliability – a case study of the Apple iPhone,'' *IEEE Access*, vol. 7, pp. 71 131–71 141, 2019. DOI: 10.1109/ACCESS.2019.2918401.

[9] J. Guo, Y. Xu, M. Exner *et al.*, ''Unravelling the mechanism of pulse current charging for enhancing the stability of commercial LiNi0.5Mn0.3Co0.2O2/Graphite Lithium-ion batteries,'' *Advanced Energy Materials*, vol. 14, no. 22, pp. 1–14, 2024. DOI: 10.1002/aenm.202400190.

[10] A. Roy, S. M. Rumble, R. Stutsman, P. Levis, D. Mazières and N. Zeldovich, ''Energy management in mobile devices with the cinder operating system,'' in *Proceedings of the 6th European Conference on Computer Systems (EuroSys '11)*, ACM, 2011, pp. 139–152. DOI: 10.1145/1966445.1966459.

[11] H. Zeng, C. S. Ellis, A. R. Lebeck and A. Vahdat, ''ECOSystem: Managing energy as a first class operating system resource,'' in *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '02)*, ACM, 2002, pp. 123–132. DOI: 10.1145/605397.605411.

[12] R. Neugebauer and D. McAuley, ''Energy is just another resource: Energy accounting and energy pricing in the Nemesis OS,'' in *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HOTOS '01)*, IEEE, 2001, pp. 67–72. DOI: 10.1109/HOTOS.2001.990063.

[13] H. Zeng, C. S. Ellis, A. R. Lebeck and A. Vahdat, ''Currentcy: A unifying abstraction for expressing energy management policies,'' in *Proceedings of the 2003 Annual Technical Conference (ATC '03)*, USENIX, 2003, pp. 43–56.

[14] International Organization for Standardization (ISO). ''ISO/IEC 21031:2024: Software Carbon Intensity (SCI) specification.'' (2024), [Online]. Available: https://www.iso.org/standard/86612.html (visited on 16/09/2024).

[15] CLEAResult. ''80 PLUS® Power Supply Certification Program.'' (2024), [Online]. Available: https://www.clearesult.com/80plus/ (visited on 16/09/2024).

[16] R. Stull. ''The latest EU efficiency regulations and how we got here.'' (2020), [Online]. Available: https://www.asme.org/topics-resources/content/the-latest-eu-efficiency-regulations-and-how-we-got-here (visited on 16/09/2024).

[17] Chargerlab. ''Review of new Apple 20W charger for iPhone 15 (A2940).'' (2023), [Online]. Available: https://www.chargerlab.com/review-of-new-apple-20w-charger-for-iphone-15-a2940/ (visited on 16/09/2024).

[18] M. Hähnel, B. Döbel, M. Völp and H. Härtig, ''Measuring energy consumption for short code paths using RAPL,'' *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 3, pp. 13–17, 2012. DOI: 10.1145/2425248.2425252.

[19] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen and Z. Ou, ''RAPL in action: Experiences in using RAPL for power measurements,'' *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 3, no. 2, 9:1–9:26, 2018. DOI: 10.1145/3177754.

[20] M. Elgamal, D. Carmean, E. Ansari *et al.*, ''Carbon-efficient design optimization for computing systems,'' in *Proceedings of the 2nd Workshop on Sustainable Computer Systems (HotCarbon '23)*, ACM, 2023, 16:1–16:7. DOI: 10.1145/3604930.3605712.

[21] S. Barcellona and L. Piegari, ''Effect of current on cycle aging of lithium ion batteries,'' *Journal of Energy Storage*, vol. 29, pp. 1–7, 2020. DOI: 10.1016/j.est.2020.101310.

[22] J. S. Edge, S. O'Kane, R. Prosser *et al.*, ''Lithium ion battery degradation: What you need to know,'' *Physical Chemistry Chemical Physics*, vol. 23, no. 14, pp. 8200–8221, 2021. DOI: 10.1039/D1CP00359C.

[23] A. Mishra, A. Bisht and T. Zhu, ''Do batteries for peak shaving green the grid?'' In *Proceedings of the 14th International Green and Sustainable Computing Conference (IGSC '23)*, ACM, 2024, pp. 29–35. DOI: 10.1145/3634769.3634805.

[24] M. Proske, D. Sánchez, C. Clemm and S.-J. Baur, ''Life cycle assessment of the Fairphone 3,'' Fraunhofer IZM, Tech. Rep., 2020. [Online]. Available: https://www.fairphone.com/wp-content/uploads/2020/07/Fairphone_3_LCA.pdf (visited on 16/09/2024).

[25] E. Emilsson and L. Dahllöf. ''Lithium-ion vehicle battery production.'' (2019), [Online]. Available: https://www.ivl.se/download/18.34244ba71728fcb3f3faf9/1591706083170/C444.pdf (visited on 16/09/2024).

[26] The Battery Pass Project. ''Advancing the implementation of the battery passport in Europe and beyond.'' (2024), [Online]. Available: https://thebatterypass.eu/ (visited on 16/09/2024).

[27] S. Ma, M. Jiang, P. Tao *et al.*, ''Temperature effect and thermal impact in lithium-ion batteries: A review,'' *Progress in Natural Science: Materials International*, vol. 28, no. 6, pp. 653–666, 2018. DOI: 10.1016/j.pnsc.2018.11.002.

[28] I. Buchmann. ''BU-808c: Coulombic and energy efficiency with the battery.'' (2021), [Online]. Available: https://batteryuniversity.com/article/bu-808c-coulombic-and-energy-efficiency-with-the-battery (visited on 16/09/2024).

[29] M. Toman, R. Cipin, D. Cervinka, P. Vorel and P. Prochazka, ''Li-ion battery charging efficiency,'' *ECS Transactions*, vol. 74, no. 1, p. 37, Dec. 2016. DOI: 10.1149/07401.0037ecst.

[30] J. S. John. ''Finally, a way to tell how clean grid batteries actually are.'' (2024), [Online]. Available: https://www.canarymedia.com/articles/batteries/finally-a-way-to-tell-how-clean-grid-batteries-actually-are (visited on 16/09/2024).

[31] The Linux Kernel documentation. ''Linux power supply class.'' (2024), [Online]. Available: https://www.kernel.org/doc/html/v6.8/power/power_supply_class.html (visited on 16/09/2024).

[32] Intel. ''ATX Version 3 Multi Rail Desktop Platform Power Supply: DC Connectors.'' (2024), [Online]. Available: https://edc.intel.com/content/www/us/en/design/ipla/software-development-platforms/client/platforms/alder-lake-desktop/atx-version-3-0-multi-rail-desktop-platform-power-supply-design-guide/2.1a/dc-connectors/ (visited on 16/09/2024).

[33] X. Zeng, J. Li and L. Liu, ''Solving spent lithium-ion battery problems in China: Opportunities and challenges,'' *Renewable and Sustainable Energy Reviews*, vol. 52, pp. 1759–1767, 2015. DOI: 10.1016/j.rser.2015.08.014.

[34] M. A. Rajaeifar, P. Ghadimi, M. Raugei, Y. Wu and O. Heidrich, ''Challenges and recent developments in supply and value chains of electric vehicle batteries: A sustainability perspective,'' *Resources, Conservation and Recycling*, vol. 180, pp. 1–11, 2022. DOI: 10.1016/j.resconrec.2021.106144.

[35] B. Acun, B. Lee, F. Kazhamiaka *et al.*, ''Carbon Explorer: A holistic framework for designing carbon aware datacenters,'' in *Proceedings of the 28th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '23)*, ACM, 2023, pp. 118–132. DOI: 10.1145/3575693.3575754.