Assessing the Energy Consumption of Software

Len Bioly, Kilian Holzinger*, Johannes Späth*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: len.bioly@tum.de, holzingk@net.in.tum.de, spaethj@net.in.tum.de

***Abstract*—To lower costs and help the environment, green software engineering is becoming more and more critical to lower software's energy consumption. Therefore, this paper reviews methods to measure the energy consumption of software, including hardware-based methods like Intel RAPL, AMD APM, and an experimental approach named SEFlab. Furthermore, it covers software-based estimations created with eLens, GreenOracle, and Silicon Labs Energy Profiler. Hardware-based methods can achieve precise measurements at runtime but suffer from a long setup time. It is also difficult to detect which current comes from which software. The software-based methods are helpful in development because they are easy to set up and can visualize the energy consumption fine-grained in the code. The paper also includes information about CloudSIM and proprietary methods of market-leading cloud distributors like Amazon to reflect the current research results in energy-efficient cloud computing.**

***Index Terms*—energy consumption, software measurement, runtime monitoring, software-based estimation, distributed systems**

1. Introduction

The quantity of information and communication technology (ICT) is rising yearly; therefore, energy consumption and costs are breaking record after record. The SMARTer 2030 Report forecasts that, in 2030, ICT will be responsible for 2% of all carbon emissions [1]. Because carbon dioxide will be the primary contributor to global warming, it is essential to lower the energy consumption of software to combat the climate crisis [2]. To achieve this objective, the developers must take exact energy measurements of their software.

There can be many methods for analysing software energy consumption; therefore, this paper outlines some measurement methods. It contains black-box testing methods measuring energy consumption at runtime and white-box software-based estimation methods integrated into development environments. This review details Intel Running Average Power Limit (RAPL), AMD APM, and SEFlab as runtime measurements, besides GreenOracle, eLens, and Silicon Labs Energy Profiler as software-based estimation methods. Furthermore, it analyzes the accuracy and precision and lines out the difficulties and concerns of the mentioned methods.

Cloud Computing has risen exponentially over the last few years [3]. More complex measuring structures are needed to provide a platform for researchers and developers to test and optimize their systems. Therefore, this paper reflects CloudSIM 7G, a state-of-the-art, robust simulation toolkit for cloud computing. To introduce an example of monitoring methods at major cloud service providers, AWS CloudWatch is detailed.

2. Related Work

A paper by Felix Rieger and Christoph Bockrish [4] concludes a summary of different studies. They reviewed existing research on green software design and assessment methods, such as Silicon Labs and SEFlab, which are contained in this paper.

Andreas Schuler and Gabriele Kotsis [5] analyzed event-based, utilization-based, code-analysis, and measurement-based methods for mobile platforms like Android or iOS, thereby reviewing individual system parts' energy consumption. Within their study, they categorized existing methods and stated problems that must be solved. In total, they reviewed 134 studies between 2011 and 2021, most of them applying the Android platform.

3. Energy Measurement Methods

The following section introduces the different methods for analyzing software energy consumption. The passage will describe the mode of operation and the functional framework of these individual methods and also state some of their limitations.

3.1. Intel RAPL

Intel RAPL allows the user access to sensors inside the CPU, allowing the CPU's and DRAM's accumulated power consumption to be distinguished. Intel introduced the method with their Sandy Bridge lineup [6], [7]. The information is stored in Machine-Specific Registers (MSRs) [1], [7], [8]. Rather than capturing physical measurements, these registers store architectural events from the cores, processor graphics, and I/O, which are processed with energy weights to estimate the active power consumption of the package [9]. The collected consumptions are displayed in Joules and are updated on average every millisecond, therefore, the granularity is 15.3 µJ for SandyBridge [6]–[8] and 61 µJ for Haswell and Skylake architectures [7].

Table 1 shows the different relevant storing registers. With Intel RAPL, CPU package power, the total consumption of the processor cores and the consumption of the

 doi: 10.2313/NET-2025-05-1_01

TABLE 1: List of available RAPL sensors, Table 1 in [1]

RAPL_PKG	Whole CPU package
RAPL_PP0	Processor cores only
RAPL_PP1	*A specific device in the uncore*
RAPL_DRAM	Memory Controller

DRAM controller can be measured. A significant disadvantage of Intel RAPL is that there is no possibility of measuring the power consumption of individual cores [1].

Intel RAPL has severe security issues, as explored by Z. Zhang et al. [10]. Especially on Linux systems, unprivileged users can read the measurements offered by Intel RAPL through the "sysfs" interface. Furthermore, the same can be done on MacOS with specialized system calls.

They analyzed the memory power consumption using DRAM access procedures. With an AVX system call, they stored data in the DRAM and measured the energy consumption. They discovered that writing small segments consumes less power than writing larger segments. Then, they implemented a receiver and a sender, which can be placed, for example, one in the container and one in the management system. With some adjustments, they established a covert channel that can transmit 0 and 1 through those energy measurements while bypassing all the security implementations. On their testing systems, they achieved a bandwidth of 50 bps while maintaining an error rate below 2% on every system [10].

3.2. AMD Energy

In comparison, AMD Application Power Management (APM) can measure the energy consumed by each core and the total socket power. The socket power measurements differ from those of Intel's package power because it is not the sum of all cores but includes cache and other CPU internal parts. In contrast to Intel RAPL, it does not provide the consumed energy in Joules, but the average consumption over the last timeframe. On a system with AMD Opteron 6274, a timeframe was about 3.8 ms long and results in a granularity of 3.8 mW. Only the information of the last segment is stored in the registers. This approach is more accurate than Intel RAPL when measuring microscopic procedures because considering two timeframes instead of one does not have a tremendous impact. The disadvantage of AMD Energy is that no power measurement of the DRAM is possible [6].

3.3. SEFlab

Ferreira et al. [11] conducted further investigations to create the SEFlab, a hardware-based black-box measuring lab. It is especially suitable for processors produced before the introduction of Intel RAPL.

They tried to get exact measurements of both CPUs, memory, fans, mainboard, and HDD. With some additional testing, they could distinguish each wire to their consumer except the power for memory banks and fans because those are distributed directly on the motherboard. To address this problem, they measured the power consumption of the memory and fans and subtracted it from the measurements of the mainboard results. All these measurements were collected with a sampling frequency of 30kHz and then stored and processed in the data acquisition system (DAQ). The DAQ transmits the data to the measurement PC, where the data is visualized. Furthermore, they extracted a pulse via USB from the server and inserted it via a serial port. With that, they could get exact measurements when the software is executed on the server. Bram Visser did a validation analysis of SEFlab and discovered an error margin around 1% [11]. SEFlab is still an experimental approach that requires much work to adapt to other hardware.

3.4. eLens

A concept for estimating energy consumption is eLens. It combines *per-instruction energy modeling* and *program analysis* to trace executed paths. eLens bases the estimations on bytecode. The conversion process to bytecode is further introduced in Section 4 of this paper. eLens can be integrated into IDEs like Eclipse to estimate different application parts. Therefore, the developers do not need additional hardware if a SEEP described below is available. The algorithm can estimate the energy consumption per line of code, path, method, and application [12].

For eLens to work correctly, a software environment energy profile (SEEP) is needed. The SEEP contains the per-instruction energy costs of every hardware component in the target machine. The researchers of eLens hope that manufacturers will publish SEEPs of their products in their future. Since SEEPs are currently not available, the researchers generated their own SEEP with the LEAP setup [12].

The *Low Energy Aware Platform* (LEAP) is a testbed invented at the University of California, Los Angeles. It contains an Intel Atom CPU on a mini-ITX motherboard. With a Digital Acquisition Device (DAQ), they can sample at a rate of up to 10kHz. The system can be further adapted to measure all parts necessary for various application types. In cooperation with its nanosecond timestamp counter, it can report fine-grained energy consumption results while executing a task [13].

eLens contains three main parts, as shown in Figure 1. The Workload Generator converts the possible user interactions into path information, which can be processed further in the Analyzer and Source Code Annotator. The generator is needed, because it would be inaccurate if all paths were executed n times. The Analyzer then receives the paths from the workload generator and assigns each path's matching energy estimates extracted from the SEED. Afterwards, the Source Code Annotator visualizes the results from the Analyzer, which can then be integrated via the Eclipse plugin. For example, the single lines of code are highlighted in blue for lower consumption and then go up in steps to red representing enormous consumption [12].

3.5. GreenOracle

GreenOracle is an energy estimation software for Android applications. It uses machine learning based on a big data approach; the creation of the dataset is further

 doi: 10.2313/NET-2025-05-1_01

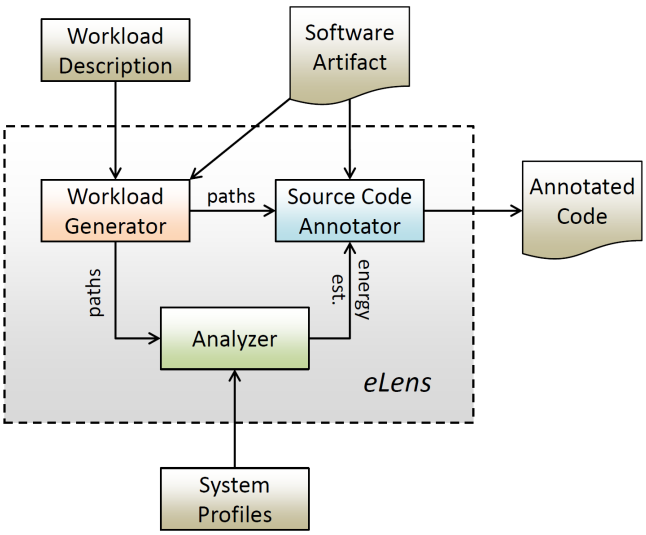

Figure 1: Structure of eLens, Fig. 1 in [12]

detailed in the accuracy and precision section of this paper. The software then calculates the energy consumption based on grouped system calls, processes with similar functions and energy consumption. With the fast execution and easy adaptation to new versions, app developers can assess energy-consuming parts of the code in progress. The results are promising, and it is universally applicable, while the achieved results are comparable to those of eLens. Nevertheless, GreenOracle is easy to use and does not require specialized hardware [14].

3.6. Silicon Labs Energy Profiler

In the world of "Internet of Things" (IoT), many household devices are connected to servers worldwide. Microcontrollers are needed to enable such functionality, which can manage and connect these devices while using only a little power. Silicon Labs provides a complete working environment with in-house software and hardware, which is available on the market. With their cooperating hardware the developer can assess the energy consumption of software since the 5th version of the Silicon Labs Studio. They manufacture 32-bit ARM Cortex cores, which can be programmed via the Silicon Labs SDK afterward [15].

The microcontroller EFM8 and EFM32 have a compatible debug interface which can be connected to a Silicon Labs starter kit containing a power supply. With the Advanced Energy Monitor (AEM) interface included in the SDK, the energy consumptions can be displayed within the IDE. The development environment has integrated the Silicon Labs Energy Profiler (SELP); therefore, multiple devices can be measured and compared simultaneously.

The visual user interface efficiently displays the live energy consumption with a waveform. The AEM interface provides a single-node and a multi-node view, where the user can see the animated live consumption. This presentation method allows the developer to efficiently assess all the needed information, make changes to the code, and measure again. Furthermore, the interactions between several devices can be measured and visualized. The system is modular overall and helps the developer getting complex measurements done in seconds, saving many work hours; e.g., the software allows the user to look into variable timeframes and modules, providing complete control and customization.

The IDE can manage the software autonomously, so the developer can start, pause, and end the measured software directly inside the development studio. Furthermore, recording the measurements is possible for later analysis, and the user can enable code correlation, whereby the software can assign the energy consumptions to each function because both code analysis and energy measurement run in parallel [16].

3.7. CloudSIM

Cloud SIM 7G is the 7th version of this open-source, java-based simulation tool. The system is not designed to assess the energy consumption of individual software but primarily to determine how to distribute the software most efficiently. The developers can simulate energy consumption of, for example, network components in geographically distributed systems while recreating the network traffic of their software. The developers of CloudSIM have integrated the CloudNetSIM++ software from OMNeT++ for this purpose. Using this, researchers can compare performances of, for example, star and mesh topology while connecting the data centers. It can manage TCP, UDP, and HTTP traffic. The system then outputs the calculated energy consumption per data center or hardware component type. With the extension ERouter, the consumption of switching and routing appliances can be assessed. It is an extensive system with an easy-to-use GUI (graphical user interface) that enables researchers and developers to simulate, optimize, and assess their software [17], [18].

3.8. AWS CloudWatch

AWS (Amazon Web Services), Google Cloud, and Microsoft Azure implement several proprietary measurement methods [19]–[21]. For example, AWS announced CloudWatch, allowing developers to see all system components' current hardware utilization. These approaches cannot measure absolute energy values but give the developers an indication of how energy-efficient those methods are compared to other software versions [19]. AWS CloudWatch works on all AWS EC2 systems, which all run Intel, AMD and proprietary AWS Graviton processors [22].

4. Accuracy and Precision

The following paragraphs will detail the experimental testing setups and measurement results, assessing the accuracy and precision of Intel RAPL, SEFlab, eLens, and GreenOracle.

The measurements of Intel RAPL improved from Sandy Bridge-EP to Haswell-EP [8]. The testing group of Khan et al. [7] did extensive research on the accuracy of Intel RAPL power measurements. They established a testing setup with an Intel Core i7-4770 @ 3.40Ghz, a Haswell workstation CPU, and an Intel i5-6500 @ 3.20Ghz, a Skylake Desktop CPU. Their analysis used Microbenchmarks, which address only a specific part of

the CPU, and application-level benchmarks like Stream or ParFullCMS. The benchmark Stream showed a strong correlation (coefficient of 0.99) between the RAPL package power, and the power drawn from the wall socket. It should be noted that this is only feasible at constant temperatures. Especially when testing the Haswell CPU, they observed a significant impact on longer benchmarks when the temperature is rising. They measured a correlation of 0.93 between package power and temperature reading. Therefore, if measuring with RAPL technology, the developers should remember that all the tests should have comparable core temperatures. Overall they estimated a mean error of 4% for Sandy Bridge and 1.7% for Haswell CPUs [7].

The accuracy of the SEFlab cannot be distinguished precisely because of the lack of testing different hardware. However, in their testing lab, they could measure very accurate results on runtime. In future work, it will also be possible to get precise predictions when enough data is collected [11].

The researchers compared the accuracy of eLens by comparing the measurements of the ground trough (GT) metered with LEAP and eLens. First, they downloaded unmodified Android applications from the Google Play Store and afterwards converted the Dalvik bytecode to Java bytecode using the *dex2jar tool*. Some applications cannot be transformed and were excluded from their validation process. Furthermore, the code has to run on the LEAP Platform and during the path measuring process, 0.01% of all paths threw an exception. All the remaining applications were given to eLens as input. They compared the estimations of eLens with the measured GT, but different problems occurred during this process. The GT counted waiting times on human input; the LEAP could not determine which energy consumption was just background noise, and the LEAP had only a sampling rate of 10 kHz. The rate is just enough to measure functions that run longer than 10 ms. This lack resulted in many functions where the GT cannot be distinguished. Therefore, they did not compare the measurements at the line of code granularity. The average error for the whole program level was 8,8% but was consistently below 10%. At the method level, the average was 7.1% and also below 10% in any case. For the hardware components, RAM and WiFi, they got a maximal error of 12% and one measurement with GPS, where an error of 8.1% occurred. These are excellent results in comparison to other software estimation methods. For example, the *average-bytecode* strategy at the whole program level had an average error of 133% and the *no-path-sensitivity* analysis 267%. Primarily because not only the CPU consumption is assessed, but also other hardware components can be included in the calculation [12].

To get high accuracy, the developers of GreenOracle collected 24 different Android applications with a total of 984 versions from *F-Droid* [23], *GitHub*, and partly from a direct website. Then they used the *Green Miner* [24], a hardware-driven energy profiler consisting of a Raspberry Pi, a Galaxy Nexus phone, and an Arduino Uno. The Raspberry Pi executes the tests on the phone and stores the measured data, while the Arduino Uno collects the energy consumption of the phone. The predefined tests consist of standard usage of the app and the user inputs are emulated with Unix shell. They executed all tests in airplane mode. After repeating each test 10 times, they collected all system calls with the *trace* command in several independent tests. Finally, they grouped similar system calls and created a table of only 13 different system calls. With advanced machine learning, they achieved an average error of 5.96% using super vector machine regression (SV) because other regression types generated worse results. However, they recommend ridge regression because the worst case is much better than SV regression. That concludes with a mean error of 6.17% and a worst-case error of 13%. These results are comparable to eLens stated above [14].

5. Error Analysis

During the testing of SEFlab, they encountered some contradictions with similar experiments, for example, an experimental analysis by H. Chen, S. Wang, and W. Shi. [11], [25]. They did not assess these issues within the paper. Furthermore, they did not address the impact of rising temperatures of the SOC when running more extended tests. They stated in the paper that CPU usage is highly correlated with its power draw. Figures 2 and 3 show that the utilization is initially at 100%, but the power draw is not at its peak. They explained this by *apparent contradictions* in their benchmarks and the *lack of scalability* reported in [11], [26]. For mobile usage, many processors are designed to be highly efficient at idle but are inefficient at peak performance. This structure is necessary because, on mobile devices, the idle time is much longer than the time when the peak performance is needed [26]. Another point that may impact the findings is that energy density on the internal head-spreader (IHS) rose over several chip generations. For example, a Xeon CPU from 2004 has a TDP of 103W on a heat spreader with a size of 42.5 mm x 42.5 mm (1806.25 mm^2) [27]. In contrast, a last-generation Intel Core like the 14900K has a TDP of 125W and a turbo of up to 253W on a 45 mm x 37.5 mm (1687.5 mm^2) IHS [28]. That is an increase of 29.9% per mm^2 of energy density. This increase can cause a higher temperature range of the SOC and, therefore, different scalabilities because the resistance of the SOC rises with rising temperatures and consumes more energy for the same performance [29].

6. Conclusion and Future Work

As shown in the review, there are many ways to assess software energy consumption. Nevertheless, to this date, no method has perfect accuracy and is easy to adapt to all scenarios. The hardware-based methods described in the first part are more precise than software-based approaches regarding runtime measurements. However, these hardware-based methods are time-consuming, and it is challenging to distinguish between total system power usage and power usage caused by the software. The software-based estimation methods are easy to adapt but have some uncertainties but still are suitable for more applications. eLens and GreenOracle are not yet available to standard developers and still need more straightforward integration with state-of-the-art software development sys-

TABLE 2: Overview of methods mentioned in the paper

Method	Platform Compatibility	Error	
INTEL RAPL	Only Intel CPUs since Sandy-Bridge + DRAM (since Sandy-Bridge Server)	4% (Sandy-Bridge) 1.7% (Haswell)	[6], [7]
AMD APM	Only AMD CPUs since 15h generation	No measured values	[6]
SEFlab	AMD & Intel CPUs as above + additional hardware components	~1%	[11]
eLens	Android based CPU + RAM, GPS, WiFi	< 10%	[12]
GreenOracle	Android based CPU	< 13%	[14]
SL E. Profiler	SL Microcontroller	No measured values	[16]
CloudSIM	distributed systems	No energy values	[17], [18]
AWS CloudWatch	AWS EC2 instances	No energy values	[19]

Note: In this table, SL stands for Silicon Laboratories.

tems. CloudSIM provides the most advanced, freely available technology in this paper, but the system has many possible future improvements; ideas could include running individual software in simulation, which would be analyzed in a manner comparable to eLens or GreenOracle. Big cloud distributors have advanced proprietary monitoring tools to shorten costs but they do not provide absolute energy values.

In conclusion, much work is needed in this segment to overcome the current boundaries, but more advanced technologies are being developed every year. Platform-independent and easy-to-use methods are not available to date. Only Intel RAPL, Silicon Labs Energy Profiler, CloudSIM, and AWS CloudWatch are commonly used in the industry. Especially for network applications, Intel RAPL is used because this method can deliver precise measurements, and most test setups in universities are Intel x86-based. However, more software-based methods with hardware integrations like eLens are also coming, making analyzing energy consumption much easier.

References

[1] M. Hähnel, B. Döbel, M. Völp, and H. Härtig, "Measuring energy consumption for short code paths using rapl," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 3, p. 13–17, jan 2012.

[2] *Climate Change 2023: Synthesis Report (Full Volume) Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*, 07.

[3] Statista Research Department, "Umsatz mit cloud computing** weltweit von 2010 bis 2023 und prognose bis 2025." [Online]. Available: https://de.statista.com/statistik/daten/studie/195760/umfrage/umsatz-mit-cloud-computing-weltweit/

[4] F. Rieger and C. Bockisch, "Survey of approaches for assessing software energy consumption," in *Proceedings of the 2nd ACM SIGPLAN International Workshop on Comprehension of Complex Systems*, ser. CoCoS 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 19–24. [Online]. Available: https://doi.org/10.1145/3141842.3141846

[5] A. Schuler and G. Kotsis, "A systematic review on techniques and approaches to estimate mobile software energy consumption," *Sustainable Computing: Informatics and Systems*, vol. 41, p. 100919, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210537923000744

[6] D. Hackenberg, T. Ilsche, R. Schöne, D. Molka, M. Schmidt, and W. E. Nagel, "Power measurement techniques on standard compute nodes: A quantitative comparison," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013, pp. 194–204.

[7] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen, and Z. Ou, "Rapl in action: Experiences in using rapl for power measurements," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 3, no. 2, mar 2018.

[8] D. Hackenberg, R. Schöne, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer, "An energy efficiency feature survey of the intel haswell processor," in *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, 2015, pp. 896–904.

[9] E. Rotem, A. Naveh, A. Ananthakrishnan, E. Weissmann, and D. Rajwan, "Power-management architecture of the intel microarchitecture code-named sandy bridge," *IEEE Micro*, vol. 32, no. 2, pp. 20–27, 2012.

[10] Z. Zhang, S. Liang, F. Yao, and X. Gao, "Red alert for power leakage: Exploiting intel rapl-induced side channels," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, ser. ASIA CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 162–175. [Online]. Available: https://doi.org/10.1145/3433210.3437517

[11] M. A. Ferreira, E. Hoekstra, B. Merkus, B. Visser, and J. Visser, "Seflab: A lab for measuring software energy footprints," in *2013 2nd International Workshop on Green and Sustainable Software (GREENS)*, 2013, pp. 30–37.

[12] S. Hao, D. Li, W. G. J. Halfond, and R. Govindan, "Estimating mobile application energy consumption using program analysis," in *2013 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 92–101.

[13] P. A. Peterson, D. Singh, W. J. Kaiser, and P. L. Reiher, "Investigating energy and security trade-offs in the classroom with the atom {LEAP} testbed," in *4th Workshop on Cyber Security Experimentation and Test (CSET 11)*, 2011.

[14] S. A. Chowdhury and A. Hindle, "Greenoracle: Estimating software energy consumption with energy measurement corpora," in *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, 2016, pp. 49–60.

[15] Silicon Laboratories, "Silicon labs 32-bit microcontrollers." [Online]. Available: https://www.silabs.com/mcu/32-bit-microcontrollers

[16] ——, "Silicon labs energy profiler." [Online]. Available: https://docs.silabs.com/simplicity-studio-5-users-guide/1.0/using-the-tools/energy-profiler/

[17] R. Andreoli, J. Zhao, T. Cucinotta, and R. Buyya, "Cloudsim 7g: An integrated toolkit for modeling and simulation of future generation cloud computing environments," *arXiv preprint arXiv:2408.13386*, 2024.

[18] A. W. Malik, K. Bilal, K. Aziz, D. Kliazovich, N. Ghani, S. U. Khan, and R. Buyya, "Cloudnetsim++: A toolkit for data center simulations in omnet++," in *2014 11th Annual High Capacity Optical Networks and Emerging/Enabling Technologies (Photonics for Energy)*, 2014, pp. 104–108.

[19] I. Amazon Web Services, "Amazon cloudwatch." [Online]. Available: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/finding_metrics_with_cloudwatch.html

[20] G. LLC, "Google cloud observability." [Online]. Available: https://cloud.google.com/monitoring/docs/monitoring-overview

[21] M. Corporation, "Microsoft azure monitor." [Online]. Available: https://learn.microsoft.com/de-de/azure/azure-monitor/best-practices-data-collection

[22] I. Amazon Web Services, "Amazon cloudwatch." [Online]. Available: https://aws.amazon.com/de/cloudwatch/

[23] F.-D. Contributors, "F-droid: Free and open source android app repository." [Online]. Available: https://f-droid.org/

[24] A. Hindle, A. Wilson, K. Rasmussen, E. J. Barlow, J. C. Campbell, and S. Romansky, "Greenminer: a hardware based mining software repositories software energy consumption framework," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 12–21. [Online]. Available: https://doi.org/10.1145/2597073.2597097

[25] H. Chen, S. Wang, and W. Shi, "Where does the power go in a computer system: Experimental analysis and implications," in *2011 International Green Computing Conference and Workshops*, 2011, pp. 1–6.

[26] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[27] I. Corporation, "64-bit intel® xeon® processor 3.20 ghz, 1m cache, 800 mhz fsb." [Online]. Available: https://ark.intel.com/content/www/us/en/ark/products/28016/64-bit-intel-xeon-processor-3-20-ghz-1m-cache-800-mhz-fsb.html

[28] ——, "Intel® core™ i9 processor 14900k." [Online]. Available: https://ark.intel.com/content/www/us/en/ark/products/236773/intel-core-i9-processor-14900k-36m-cache-up-to-6-00-ghz.html

[29] R. Chu, R. Simons, M. Ellsworth, R. Schmidt, and V. Cozzolino, "Review of cooling technologies for computer products," *IEEE Transactions on Device and Materials Reliability*, vol. 4, no. 4, pp. 568–585, 2004.