Messverfahren für Green Software

Bundesverband Green Software

Arne Tarara, Kollin Freise, David Kopp,
Andreas Weber, Michael Hagedorn

Version 1

23. Oktober 2025

Die zunehmende Dringlichkeit der Klimakrise, steigende regulatorische Anforderungen, Berichtspflichten und die gesellschaftliche Verantwortung aller Wirtschaftssektoren bei gleichzeitig steigendem Anteil des IT-Sektors am weltweiten Stromverbrauch machen es unerlässlich, die Umweltauswirkungen von Software zu betrachten. Nur was messbar ist kann verbessert werden. Dieses Whitepaper gibt einen praxisorientierten Überblick, über aktuelle Methoden und Standards zur Erfassung und Bewertung von Energieverbrauch und CO_2-Emissionen im Kontext von Green Software. Es beleuchtet Herausforderungen und Möglichkeiten von Messungen entlang des gesamten Software-Lebenszyklus und in verschiedenen Bereichen der Software-Entwicklung. Das Whitepaper beantwortet die zentralen Fragen, warum und wann gemessen wird, was messbar ist und wie die Messung erfolgen kann.

Dimensionen von Umweltauswirkungen durch Software

Die Messung im Kontext Green Software zielt darauf ab, Umweltauswirkungen von Software zu quantifizieren. Hierbei können zwei Arten von Umweltauswirkungen unterschieden werden:

1. Umweltauswirkungen, die sich aus dem zweckmäßigen Einsatz der Software ergeben.

Eine Software, welche Betreiber von Windparks unterstützt geeignete Flächen zu finden, hat beispielsweise andere Umweltauswirkungen als eine Software, welche bei der Vermarktung von Braunkohle unterstützt. Diese Art von Umweltauswirkungen wird häufig – bei positiven Auswirkungen – unter dem Begriff *Green-BY-IT* oder Software-*Handprint* zusammengefasst. Sie sind sehr individuell und werden für dieses Whitepaper nicht weiter betrachtet. Dennoch halten die Autoren es für wichtig, den Einsatzzweck von Software kritisch zu hinterfragen.

2. Umweltauswirkungen, die durch die Entwicklung und den Betrieb von Software entstehen.

Beispiele hierfür sind Energieverbrauch, Emissionen, Wasserverbrauch oder den Verbrauch von Ressourcen zur Herstellung der ausführenden Hardware. *Green-IN-IT* oder auch *Green Software* beschäftigt sich mit der Reduktion dieser Auswirkungen. Das Messen dieser Auswirkungen ist daher Fokus dieses Whitepapers.

Messen ist grundsätzlich kein neues Thema und wird in bestehenden Softwareprojekten sowohl zur Entwicklungszeit als auch in der Produktivumgebung bereits in vielen Unternehmen gelebt, Stichwort Observability.

Wenn man jedoch umweltfreundliche Software im Sinne von Green Software anstrebt, ist das Messen essentiell und geht über die klassischen Performance-Metriken hinaus.

Vielmehr sind Energieverbrauch und CO_2-Emissionen Schlüsselmetriken, die erhoben werden. Diese leiten sich teilweise aus Performance-Metriken ab, kommen jedoch zu großen Teilen aus Energie-Messungen, Machine Learning-Modellen oder Datenbanken. Eine wichtige Rolle spielen dabei vor allem CO_2-Emissionen, die entstehen, wenn Hardware nicht ausgelastet ist, sondern im Ruhezustand läuft oder sogar ungenutzt als Redundanz vorgehalten wird.

In diesem Whitepaper stellen wir zunächst bestehende Mess- und Bilanzierungsstandards als theoretische Grundlage vor. Anschließend geben wir einen Überblick über Ansätze zur praktischen Vermessung und Attributierung. Abschließend wird erläutert, wie sich die Messungen in den verschiedenen Phasen des Software-Lebenszyklus unterscheiden und welche Herausforderungen dabei jeweils auftreten.

Das Whitepaper gibt keine Empfehlungen für den Einsatz einzelner Tools. Wir als Bundesverband Green Software stellen jedoch eine umfangreiche Übersicht der verfügbaren Tools aus dem Green Software-Ökosystem in Form einer Tool-Landkarte zur Verfügung: https://landscape.bundesverband-green-software.de

Inhaltsverzeichnis

1. Warum wird gemessen

Die Erfassung der Umweltauswirkungen bei der Entwicklung und Bereitstellung von Software verfolgt stets das Ziel, die durch Software entstehenden Umweltauswirkungen zu quantifizieren. Typischerweise dient sie dazu, anhand der Messergebnisse Entscheidungen zu treffen, wie die Emissionen der Software reduziert werden können. Seltener erfolgt die Messwerterhebung, um Reporting-Pflichten wie ESG-Reportings, interne Audits oder CSRD-Berichte zu erfüllen. Die Unterscheidung zwischen diesen beiden Ansätzen findet man im Englischen als "Measurement for Action" und "Measurement for Reporting".

Ein weiterer wichtiger Grund für Messungen ist das Verständnis, ob überhaupt relevante CO_2-Emissionen vorliegen. Häufig verursachen Unternehmen zwar nur geringe Emissionen durch eigene Rechenzentren, jedoch hohe Emissionen durch die Nutzung von Cloud-Ressourcen (Upstream) oder durch die Nutzung durch Endanwender (Downstream).

Typische Beispiele hierfür sind Streaming- oder Werbe-Geschäftsmodelle, die vergleichsweise geringe Hardwareanforderungen haben, aber durch die Skalierung und Auslieferung an viele Endanwender hohe Emissionen verursachen.

2. Was wird gemessen

2.1 Umfang der Messung

Anders als bei klassischen Performance-Messungen ist es bei der Erfassung von Umweltauswirkungen wichtig, die gesamte Infrastruktur, welcher zur Funktionalität und Entwicklung der Software nötig ist, mit einzubeziehen.

Je nach Anwendungsfall müssen jedoch teilweise Vereinfachungen vorgenommen werden, etwa wenn Daten fehlen oder externe Dienste genutzt werden.

Je nach Anwendung sind eine oder mehrere der Komponenten für die Messung relevant:

- **Server-Infrastruktur**
- **ggf. Netzwerk-Infrastruktur**
- **ggf. Rechenzentrums-Overhead**
- **ggf. End-Nutzer-Geräte** (Desktop-PCs, Mobilgeräte, Monitore, Peripherie etc.)

Gemäß dem GHG Protocol werden hier sowohl die Komponenten betrachtet, die man selbst direkt besitzt (on-prem), als auch solche, die man mietet oder implizit benutzt, bzw. auslastet (Netzwerke, Endkunden-Geräte, Cloud-Ressourcen). Das GHG Protocol definiert diese impliziten Ressourcen als Scope 3. (Für Details zu GHG-Scopes sei hier auf die gute Erklärung bei der Green Software Foundation[1] verwiesen.)

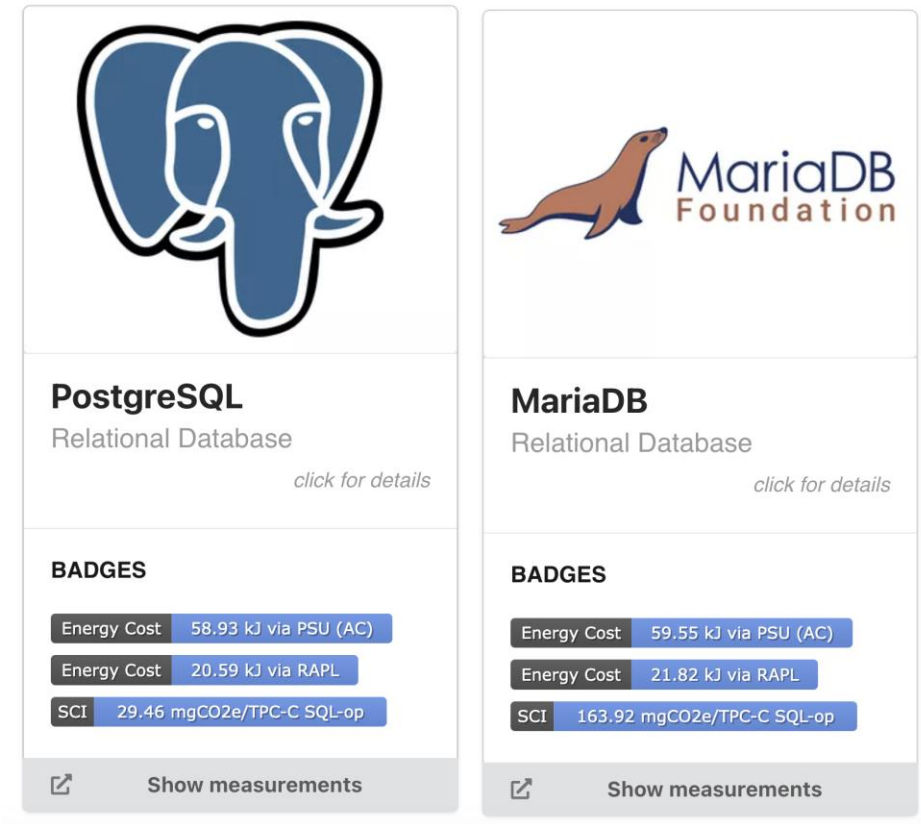
[1] https://learn.greensoftware.foundation/measurement/

2.2 Erhobene Daten

Für jede der unter 2.1 genannten Komponente werden im Folgenden vorrangig folgende Faktoren betrachtet:

- **Energieverbrauch** und zugehörige **Stromnetz-CO_2-Intensität** (Zur Ermittlung der CO_2-Emissionen durch Energieerzeugung)
- "Embodied Emissions" (**CO_2-Emissionen** durch die **Herstellung der Hardware)**

Nicht für jede Anwendung bzw. jede Infrastrukturkomponente lassen sich die tatsächlichen Energieverbräuche exakt messen. Teilweise benötigt es Abschätzungen mithilfe von Proxy Metriken, wie z. B. der Auslastung von Hardware, dem transferierten Datenvolumen, der Anzahl an Netzwerkanfragen, etc. Jedoch ist das Ziel immer, Einschätzungen zum Energieverbrauch zu erhalten und diese ggf. in CO_2-Emissionen umzurechnen.



Beispiele für den SCI Score von zwei Datenbanken für eine SQL-Benchmark (TPC-C)
Quelle: https://www.green-coding.io/products/energy-id/

Wir fokussieren uns in diesem Whitepaper auf Energieverbräuche und CO_2-Emissionen, da diese sowohl die wichtigsten Umwelteinflussfaktoren angesichts der aktuellen Klimakrise darstellen als auch die Faktoren sind, auf die Softwareentwickler:innen am stärksten Einfluss haben.

In einer vollständigen Betrachtung der Umweltauswirkungen von Software müssen ggf. noch weitere Aspekte betrachtet werden, wie etwa: Wasserverbrauch, Landnutzung, Nutzung von Ressourcen zur Herstellung der ausführenden Hardware (ADP), Elektroschrott-Erzeugung, etc.

Eine gute Übersicht bietet hier die Publikation "Methodology Paper on the Environmental Footprint of Digital Services" des eco:digit Projekts von Jens Gröger[2].

Für das vorliegende Whitepaper, das sich vorrangig an Softwareentwickler:innen und nicht an Infrastruktur-Anbieter richtet, werden daher nur die zuvor genannten vorrangigen Metriken im Detail betrachtet.

[2] https://ecodigit.de/en/home/news-and-activities/detail/methodenpapier-zur-oekobilanz-digitaler-dienstleistungen-veroeffentlicht

3. Wie wird gemessen

3.1 Welche Methodiken / Standards gibt es

ISO 14001 & GHG Protocol

Die ISO 14001[3] ist eine Norm zur Umweltbilanzierung. In dieser wird formal beschrieben, wie ein Produkt nach seinen Auswirkungen – Entnahmen aus und Abgabe von Einflüssen in die Umwelt – bilanziert werden kann.

Neben formalen Anforderungen an den Prozess wird eine Methodologie gefordert, welche anerkannt und standardisiert ist. Etabliert hat sich das GHG Protocol.

Im GHG Protocol für Software (ICT Sector Guidance[4]), wird beschrieben, welche formalen Anforderungen eine Messung des Energie- und CO_2-Verbrauchs einer Software erfüllen muss. Es werden Praxis-Beispiele geboten, wie eine Messung z.B. in virtualisierten Umgebungen erfolgen kann. Zu beachten ist jedoch, dass dieser Abschnitt inzwischen veraltet ist. Es wird beispielsweise von Auslieferung per CDs gesprochen und der Einbezug von KI und deren relevanten Anteilen an Energie und CO_2-Emissionen fehlt vollständig.

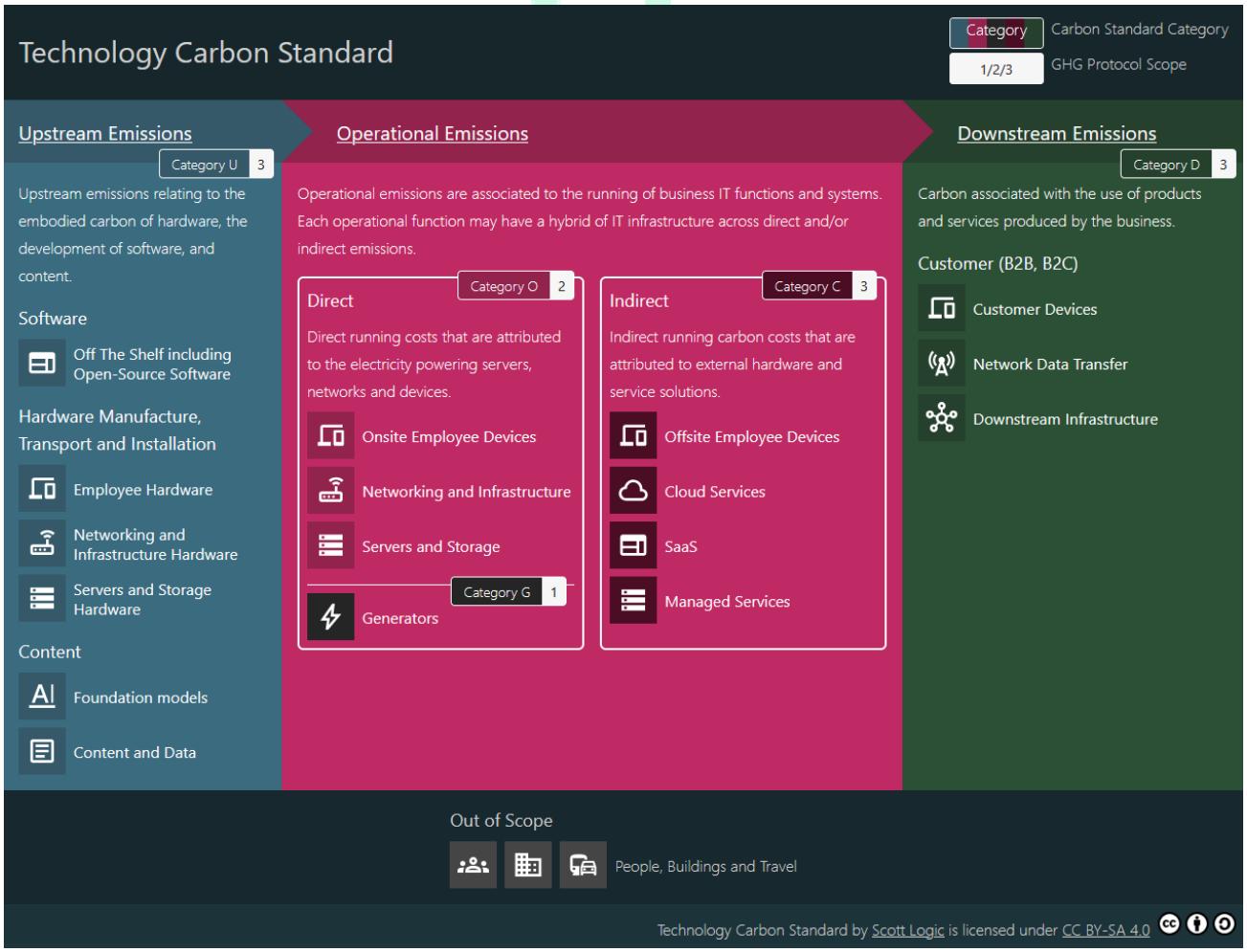
Der Leitfaden sollte nur genutzt werden, wenn dieser formal nötig ist, z.B. für spezielle Reportings. Ein Einblick wie dies z.B. für eine Cloud-Anwendung geschehen kann, gibt das Paper „Assessing the suitability of the Greenhouse Gas Protocol for calculation of emissions from public cloud computing workloads“[5].

[3] https://www.iso.org/standard/60857.html
[4] http://www.ghgprotocol.org/sites/default/files/ghgp/GHGP-ICTSG%20-%20ALL%20Chapters.pdf
[5] https://doi.org/10.1186/s13677-020-00185-8

Technology Carbon Standard

Der „Technology Carbon Standard"[6] stellt eine konkrete Anwendung des GHG Protocol für die Erhebung von IT-spezifischen CO_2-Emissionen dar, welche sehr aktuell in 2024 veröffentlicht wurde. Es handelt sich hierbei bislang nicht um einen offiziellen Standard, sondern um einen Vorschlag vom Unternehmen Scott Logic.

Er stellt die IT-bezogenen Emissionen im Betrieb sowie der vorgelagerten und nachgelagerten Emissionen übersichtlich dar (siehe folgendes Schaubild).



https://www.techcarbonstandard.org/

[6] https://www.techcarbonstandard.org/

SCI (ISO/IEC 21031:2024) [7]

Standard der Green Software Foundation[8] zur Bestimmung der sog. **S**oftware **C**arbon **I**ntensity. Die Idee dahinter ist, die innerhalb einer festgelegten Systemgrenze entstehenden Emissionen für einen konkreten, realen Anwendungsfall zu erfassen.

Der Standard bietet ein loses spezifiziertes Framework, um einen CO_2-Wert für eine Software in einem konkreten Anwendungsfall / Business-Case zu erhalten.

Zuerst werden die Systemgrenzen für den Anwendungsfall festgelegt, für welche der SCI berechnet werden soll.

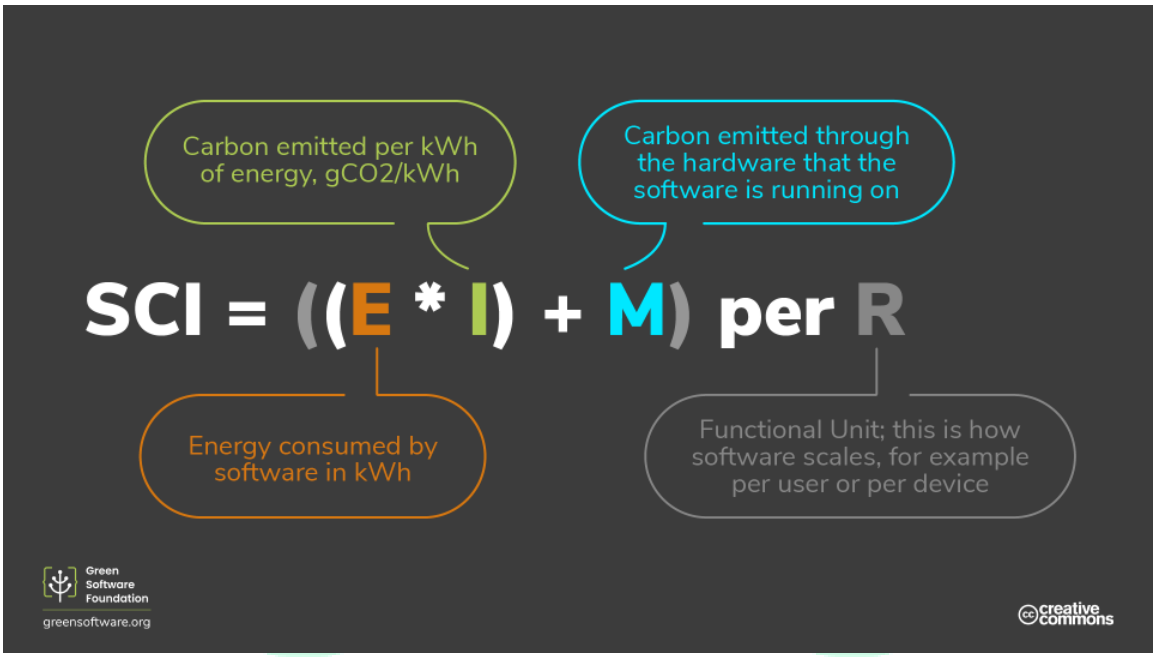
Ein Beispiel für die Systemgrenze kann sein: Der ganze Server. Oder auch nur: Der Laptop des Kunden wenn dieser auf die SaaS-Anwendung zugreift. Oder: Nur die Backup-Lösung für unsere Server.

Beispiele für Anwendungsfälle könnten z. B. sein:

- Ein Business-Case ("Berechnung der Kreditwürdigkeit eines Kunden")
- Eine konkrete Funktionalität einer Software ("Versenden eines Newsletters")
- Der 24h-Betrieb einer Software.

Für diesen Anwendungsfall muss dann der Energieverbrauch, die Stromnetz-CO_2-Intensität und auch der Embodied Carbon Wert ermittelt werden.

[7] https://www.iso.org/standard/86612.html
[8] https://greensoftware.foundation/



(E) Energieverbrauch (Kilowattstunden) für verschiedene Komponenten:
Bsp. CPU/GPUs, Datenspeicher, Arbeitsspeicher, Netzwerk

(I) Emissionsfaktor des Stromnetz

(M) Embodied Carbon (gebundene Emissionen)
Bsp. Herstellung von Server, Mobilgeräte und Laptops

(R) Funktionale Einheit, Geschäftsmetrik, etc.

SCI Formel

Wie die Werte für *E* und *M* erhoben werden sollen wird nicht näher spezifiziert, lediglich dass bei der CO_2-Intensität des Stroms nicht mit Power Purchase Agreements (PPAs) bzw. *Market Based Intensity* gearbeitet werden darf. Stattdessen muss die sog. *Location Based Intensity* erhoben werden.

Die CO_2-Intensität des Stroms (I), d. h. die Höhe der Emissionen pro kWh soll dabei ortsbezogen und nicht marktbezogen einbezogen werden. Das verhindert das Grünrechnen durch den Einkauf von Ökostrom-Zertifikaten. Um die Vergleichbarkeit bspw. zwischen Softwareversionen zu gewährleisten, kann bspw. der jährliche Durchschnittswert für die CO_2-Intensität vom Strommix in Deutschland zurückgegriffen.

Hilfreich ist der SCI vor allem bei der Betrachtung von Komponenten, die sich klar abgrenzen lassen und für die es definierte Use-Cases oder Lastprofile gibt, wie dies etwa bei Datenbanksystemen der Fall ist.

Die SCI bietet ein gutes Rahmenwerk, um bei gegebenen Systemgrenzen und Anwendungsfällen systematisch CO_2-Werte für Software zu erheben.

3.2 Überblick Mess- sowie Attribuierungs-Ansätze

Wie im vorhergehenden Kapitel erwähnt, sind, unabhängig von der verwendeten Methodik, folgende Kennzahlen zu erheben:

- Energie
- CO_2-Intensität des Stroms
- Embodied Emissions

Der Fokus liegt auf der Erhebung des Energieverbrauchs. Dieser lässt sich anschließend mit der CO_2-Intensität vom Stromnetz in CO_2-Emissionen umrechnen. Addiert mit den Embodied Emissions und letztlich der zeitlichen Attribuierung erhält man den CO_2-Fußabruck einer Software.

3.2.1 Energie

Die Erfassung des Energieverbrauchs unterscheidet sich je nach Komponente bzw. Infrastruktur-Ebene, so dass sich dieser Abschnitt aufteilt in Server, Cloud, Rechenzentrums-Overhead, Netzwerk-Infrastruktur und End-Nutzer-Geräte.

3.2.1.1 Server

Sofern physischer bzw. messtechnischer Zugriff auf die Infrastruktur besteht, auf der die Software betrieben oder entwickelt wird, ist eine direkte Messung zu bevorzugen. Diese kann je nach Gerät klassisch mit einem Energiemessgerät oder mit On-Board Sensoren auf der Hardware erfolgen.

Die Messauflösung sollte dabei dem Anwendungsfall entsprechend gewählt werden. Auf einem Server laufen typischerweise mehrere Anwendungen auf dem System, so dass nur die Summe der Energieverbräuche aller Anwendungen ermittelt werden. Hierfür eigenen sich klassische Steckdosen-Leisten mit Energy-Monitoring[9] und Auflösungen von

[9] https://gude-systems.com/en/power-distribution-units/metered-power-distribution-units/

1s, IPMI-Schnittstellen[10] (Beispielimplementierung [11]) auf dem Base-Management-Controller (BMC) oder auch Shelly-Plug[12] Systeme sehr gut.

Sofern eine Anwendung im Detail vermessen werden soll, um zum Beispiel energieintensive Komponenten oder sogar *Lines of Code* einer Anwendung zu identifizieren, bieten sich Energiemessgeräte mit hohen Auflösungen im Millisekunden-Bereich an. Dazu zählen Messgeräte wie MCP39F511N[13], oder On-Board Sensoren wie die RAPL-Schnittstelle [14] für Intel oder AMD Prozessoren sowie das *powermetrics* Interface [15] von MAC.

Server verbrauchen bereits im Leerlauf (Active Idle) signifikant Energie. Bei modernen Servern liegt die Leistungsaufnahme im Leerlauf bei 20-30 % der maximalen Leistungsaufnahme (siehe z. B. SPECpower Datenbank[16]). Soll der absolute Energieverbrauch einer Software erfasst werden, ist es nötig, auch den Idle-Energieverbrauch zu betrachten. Soll lediglich eine relative Betrachtung vorgenommen werden, kann hierauf verzichtet werden.

Wird ein Server exklusiv von einer Anwendung genutzt, lässt sich der Idle-Energieverbrauch einfach messen und der Bilanz der Anwendung zurechnen. In der Cloud (siehe nächsten Abschnitt) stellt dies aufgrund der geteilten Nutzung der Server durch mehrere Kunden oder Anwendungen eine Herausforderung dar. Hier gibt es meist keine Möglichkeit zu erfahren, wie viele andere Anwendungen mit welcher Ressourcenanforderung auf dem gleichen Server aktiv sind. Es ist somit unmöglich eine exakte Aufteilung des Idle-Energieverbrauchs zu berechnen.

---

[10] https://en.wikipedia.org/wiki/Intelligent_Platform_Management_Interface
[11] https://github.com/green-coding-solutions/green-metrics-tool/blob/main/metric_providers/psu/energy/ac/ipmi/machine/ipmi-get-machine-energy-stat.sh
[12] https://www.shelly.com/
[13] https://www.microchip.com/en-us/development-tool/ADM00706
[14] https://www.green-coding.io/case-studies/rapl-and-sgx/
[15] https://www.green-coding.io/blog/power-measurement-on-macos/
[16] https://www.spec.org/power_ssj2008/results/

3.2.1.2 Cloud

Typischerweise sind in geteilten Cloud-Infrastrukturen wie etwa AWS EC2 (Non-Bare-Metal), GCP und GitHub Actions die Sensor-Schnittstellen (u.a. RAPL) gesperrt. Hintergrund hierfür sind vor einigen Jahren bekanntgewordene Sicherheitslücken wie Platypus[17]. Entsprechend ist man in geteilten Cloud-Umgebungen auf Annäherungsmodelle angewiesen.

In der wissenschaftlichen Literatur[18] lässt sich eine Kategorisierung der Schätz-Methoden in drei Kategorien finden:

- Empirische Parametrisierung
- Funktionsregression
- Machine Learning

Empirische Parametrisierung ist die einfachste Variante. Ein bekanntes Beispiel hierfür ist *Cloud Carbon Footprint*[19]. Cloud Carbon Footprint arbeitet mit einer simplen linearen Annäherung unter Einbezug des Verbrauchs im Leerlauf und bei Vollauslastung, berücksichtigt dafür aber auch Heuristiken für Komponenten wie Festplatten, Arbeitsspeicher, Netzwerk usw.

Bei der Funktionsregression werden zur Annäherung des Energieverbrauchs reale Nutzungsdaten (CPU-Auslastung) und bekannte Messwerte (aus der SPECpower-Datenbank[20]) genutzt, so dass genauere Ergebnisse als bei einer linearen Annährung erzielt werden können. *Teads*[21] ist hierfür ein bekanntes Beispiel.

---

[17] https://www.spec.org/power_ssj2008/results/
[18] Lin, W. et al. (2020). A Taxonomy and Survey of Power Models and Power Modeling for Cloud Servers. https://doi.org/10.1145/3406208
[19] https://www.cloudcarbonfootprint.org/
[20] https://www.spec.org/power_ssj2008/results/
[21] https://engineering.teads.com/sustainability/carbon-footprint-estimator-for-aws-instances/

ML-Modelle sind auf Energie-Kennlinien von vorher gemessenen Servern trainiert und können so bei Eingabe von den Kenndaten des Systems (CPUs, Speicher, Auslastung, Hersteller etc.) einen genaueren Energiewert als die zwei anderen Annährungsmodellen ausgeben. Beispiele für Tools, die ML-Modelle nutzen, sind *Cloud Energy*[22] oder *PowerLetrics*[23].

Weitere Tools zur Abschätzung der Energieverbräuche in der Cloud finden Sie in unserer Tool-Landkarte[24].

Bei der Nutzung externer Dienste wie etwa Datenbanken in der Cloud, Logging-Diensten, oder SaaS bestehen in der Regel weder direkte Messmöglichkeiten noch indirekte Schätzverfahren.

Liegt der Fokus der Messung auf der Reduktion der Emissionen, so kann es sinnvoll sein, die Systemgrenze hier bewusst so zu ziehen, dass diese Komponenten nicht berücksichtigt werden, da sie – außer durch den Ersatz mit Alternativen – nicht optimiert werden können.

Sollen Energieverbrauch bzw. Emissionen dennoch zum Zweck der Bilanzierung oder zur Abschätzung des Einflusses eines Ersatzes mit Alternativen eingeordnet werden, so bieten sich eine Worst-Case Betrachtung (siehe unten) oder die Replikation und Vermessung einer lokalen Test-Umgebung an.

Vermessung einer lokalen Testumgebung: Hier ist nicht bekannt auf welcher Hardware die Datenbank genau läuft. Man könnte in Folge eine vergleichbare Applikation (PostgreSQL) lokal aufsetzen und mit den gleichen Daten und Anfragen durchmessen. Diese Werte können dann für eine Anwendungs-Analyse genutzt werden.

[22] https://github.com/green-coding-solutions/cloud-energy
[23] https://github.com/green-kernel/powerletrics
[24] https://landscape.bundesverband-green-software.de

Worst-Case Betrachtung: Man wählt das vermutlich größte System aus, das der Anbieter als Infrastruktur eingesetzt haben könnte. Dann schätzt man die typische Auslastung dieses Systems ab – meist zwischen 50 % und 100 %. Anschließend nutzt man die maximalen Stromverbräuche der vermuteten Hauptkomponenten, um den gesamten Energiebedarf abzuschätzen.

Solche Abschätzungen sind in vielen Fällen aufwendig. Der Bundesverband Green Software e.V. setzt sich für mehr Transparenz durch Anbieter solcher externen Dienste ein. Das würde den Vergleich von Diensten hinsichtlich ihrer Umweltauswirkungen deutlich vereinfachen und ermöglichen, diese bei der Auswahl des Dienstes zu berücksichtigen.

3.2.1.3 Rechenzentrums-Overhead

Der Rechenzentrums-Overhead ist in der Bilanzierung eine unerlässliche Komponente, da er nicht selten einen relevanten Anteil (20% und mehr) am Gesamtverbrauch ausmachen kann. Als Metrik für den Overhead dient die Power Usage Effectiveness (PUE). Ein PUE-Wert von 1.2 bedeutet, dass das Rechenzentrum einen Overhead von 20 % im Vergleich zu den Energieverbräuchen der IT-Komponenten verursacht.

Sofern eine Bilanzierung nötig ist, sei auf die Methodik bei "Methodology Paper on the Environmental Footprint of Digital Services" [25] des eco:digit Projekts von Jens Gröger verwiesen. Hier wird der Rechenzentrums-Overhead anteilig auf die jeweiligen Komponenten umgelegt.

Wenn das Ziel der Messung die Optimierung der Software ist oder die Messung in Umgebungen wie der Cloud erfolgt, kann der Rechenzentrums-Overhead als Konstante

[25] https://ecodigit.de/en/home/news-and-activities/detail/methodenpapier-zur-oekobilanz-digitaler-dienstleistungen-veroeffentlicht

angenommen und nicht berücksichtigt werden. Der Grund dafür ist, dass die Software keinen Einfluss auf den Overhead hat und dieser bei einer isolierten Betrachtung der Software daher nicht berücksichtigt werden muss.

3.2.1.4 Netzwerk-Infrastruktur

Für die Netzwerk-Infrastruktur werden typischerweise Schätzwerte aus Studien und akademischen Publikationen genutzt. Eine hilfreiche Metrik ist entweder die Bandbreite, die Zeit oder die Menge an Datentransfer. Die üblichste in der Industrie genutzte Methodik ist der Datentransfer. Hierbei wird ein Faktor mit der Dimension Energie pro Datenmenge genutzt. Konkret: kWh/GB.

Diese Methodik ist jedoch sehr umstritten, da der Energieverbrauch der Netzwerkkomponenten nicht mit der Steigerung an Datenverkehr korreliert. Die Methode ist jedoch nützlich, um die Relevanz von Netzwerkkommunikation deutlich zu machen und die Verantwortung entstehender Emissionen verschiedenen Parteien zuzuweisen (Attributional LCA). Eine Diskussion über die verschiedenen Ansätze und Vor-/Nachteile findet sich im Blog-Artikel „How to Measure and Act on Network Carbon Emissions in Green Software[26]“ von Arne Tarara und David Kopp.

3.2.1.3 End-Nutzer-Geräte

End-Nutzer-Geräte werden je nach Fragestellung unterschiedlich betrachtet. Um das Verhalten der Software auf einem Endgerät sowie die Interaktion mit der Server-Anwendung detailliert zu analysieren, ist es erforderlich, das Endgerät nach denselben Kriterien wie Server zu vermessen. Möchte man eine Bilanzierung z. B. nach dem GHG-Standard vornehmen, so ist bei den End-Nutzer-Geräten lediglich der Energie-Verbrauch zu erfassen sowie die Kosten der beanspruchten Netzwerk-Infrastruktur. Auch hier können die

[26] https://www.green-coding.io/blog/network-carbon-emissions-in-green-software/

Ansätze aus dem Bereich 3.2.1.1 Server und 3.2.1.4 Netzwerk-Infrastruktur übernommen werden.

Für die CO_2-Bilanzierung von Websites wird häufig auf Online-Rechner zurückgegriffen (z. B. websitecarbon.com). Es muss jedoch beachtet werden, dass bei solchen Rechnern mehrere Einschränkungen vorherrschen:

- Eingeschränkte Methodik (Abschätzung erfolgt meist nur anhand der Datenmenge)
- Messung nur von Ladevorgängen ohne Nutzerinteraktion (wichtige Aspekte fehlen, wie Cookies akzeptieren, Eingaben tätigen, Lazy Loading beim Scrollen, etc.)
- Aspekte wie Hintergrundaktivitäten, Animationen und auch die Aktivitäten auf Serverseite werden gar nicht oder nur unzureichend berücksichtigt

Für eine akkurate Bewertung der Umweltauswirkungen von End-Nutzer-Geräten sollte deshalb auch bei clientseitiger Software eine Erfassung realer Energieverbräuche von typischen Nutzungsszenarien vorgenommen werden, wie dies auch bei serverseitigen Softwareanwendungen der Fall ist.

Für eine Auswahl an Tools zur Erfassung der Umweltauswirkungen von Websites, siehe die Kategorie „Website“ in der Tool-Landkarte[27]. Tools mit Fokus „Desktop Application“ und „Mobile App“ lassen sich auch in der Tool-Landkarte finden.

3.2.2 CO_2

Nachdem der Energieverbrauch ermittelt wurde, gilt es nun, die tatsächlichen Umweltauswirkungen zu berechnen. Hierzu gehören insbesondere die CO_2-Emissionen.

[27] https://landscape.bundesverband-green-software.de/

3.2.2.1 CO2-Intensität des Stroms

Der Emissionsfaktor für genutzte Energie ist am einfachsten über vorhandene APIs zu ermitteln. Da der Emissionsfaktor vom Ort, als auch vom Zeitpunkt abhängt sollte er möglichst parallel zur Messung, bzw. wiederholt über die Mess-Dauer abgefragt werden. Ob der Einbezug des aktuellen Emissionsfaktors sinnvoll ist, hängt vom Ziel der Messung ab. Dienen die Messergebnisse dem Vergleich, ob sich die Software verbessert oder verschlechtert hat, würde der Einbezug die Resultate verfälschen – hier eignet sich etwa der globale Energiemix besser. Geht es hingegen um die Erfassung der tatsächlichen Emissionen, beispielsweise für die Nachhaltigkeitsberichterstattung, sollte der Emissionsfaktor für den jeweiligen Zeitpunkt und Ort berücksichtigt werden. Auch für Optimierungsszenarien, die auf den Emissionsfaktor eingehen (Carbon-Awareness), spielt dieser eine entscheidende Rolle.

Hier eine Übersicht von Anbietern:

Anbieter	Abdeckung	Kostenmodell	Besonderheiten / Quelle
Electricitymaps	International	Kostenloses Basispaket, kostenpflichtig für Forecasts	Echtzeit- und Prognosedaten zu CO_2-Intensität
WattTime	International	Kostenpflichtig	Fokus auf Echtzeit-Emissionsdaten und Automatisierung
Ember	International	Kostenfrei	Monatliche Bereitstellung von CO_2- und Stromdaten
ENTSO-E	Europa	Kostenfrei	Daten zum europäischen Stromnetz werden annähernd in Echtzeit bereitgestellt
Energy-Charts	Europa	Kostenfrei	Daten vom Fraunhofer ISE Eine API wird von bluehands[28] bereitstellt.
smard	Deutschland	Kostenfrei	Offizieller Provider: Bundesnetzagentur

Siehe auch die Kategorie „Databases - Grid Carbon Intensity API Service“ in unserer Tool-Landkarte[29].

[28] https://www.carbon-aware-computing.com/
[29] https://landscape.bundesverband-green-software.de/

3.2.2.2 Embodied Emissions – CO2-Emissionen durch die Herstellung

Die Emissionsdaten aus der Herstellung der Hardware lassen sich entweder direkt aus den Datenblättern der Hersteller oder aus Datenbanken öffentlicher und privater Anbieter beziehen. Letztere aggregieren die Daten und erzeugen Mittelwerte für die Komponenten von Herstellern, die diese nicht veröffentlichen.

Beispiele für Hardware-Produkte mit der veröffentlichen CO_2-Daten:

- Microsoft Surface Books[30]
- Dell Server[31]

Beispiele für Datenbanken:

- Boavizta[32] (NGO) – Open Source Datensatz, API[33] und Online-Rechner[34] für Endgeräte, Server, und Cloud-Instanzen
- resilio.tech[35] - Privater Anbieter mit Datenbank
- ADEME[36] - Französische öffentliche Datenbank

Siehe auch die Kategorie „Databases - IT Emission Inventory“ in unserer Tool-Landkarte[37].

[30] https://tco.exploresurface.com/sustainability/calculator
[31] https://www.delltechnologies.com/asset/en-us/products/servers/technical-support/Full_LCA_Dell_R740.pdf
[32] https://boavizta.org/
[33] https://api.boavizta.org/docs
[34] https://datavizta.boavizta.org/
[35] https://resilio-solutions.com/en
[36] https://data.europa.eu/data/datasets/5db1a0f46f444104866d1b43?locale=en
[37] https://landscape.bundesverband-green-software.de/

3.2.3 Zeitliche Attribuierung / Abschreibung

Für alle vorangehend besprochenen Werte (Energie, CO_2-Intensität des Stroms, Embodied Emissions) muss noch betrachtet werden, wie lange diese für die Software in Anspruch genommen werden.

Dabei ist stets die Zeit zu berücksichtigen, in der die Hardware ungenutzt oder lediglich reserviert ist.

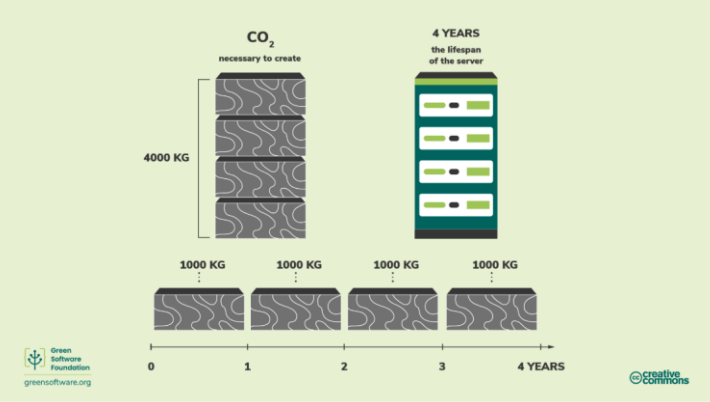
Für die Energie wird einfach die konkrete Leistungsaufnahme mit der Zeit multipliziert. Bei den Embodied Emissions wird die Nutzungsdauer der Hardware zugrunde gelegt, und der entsprechende Anteil wird anteilig pro Mess- bzw. Reservierungsdauer der Software zugerechnet.

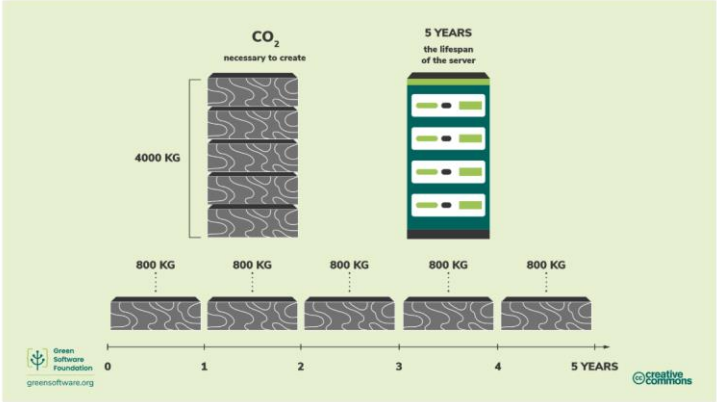
Sofern eine Hardware nach der geplanten Nutzungsdauer weiterverwendet wird, gibt es zwei Wege der Attribuierung:

Für die Bilanzierung z.B. in CSRD-Berichten empfiehlt es sich die Hardware auf 0 kgCO_2e abzuschreiben

Für eine kontinuierliche Software-Messung mit dem Ziel der Optimierung empfiehlt es sich, die Nutzungsdauer der Hardware kontinuierlich zu verlängern, sodass der pro Zeit berechnete Anteil der Embodied Emissions immer kleiner wird. Dies bildet am besten die Maßnahme ab, die Hardware länger zu nutzen, und zeigt die Reduzierung der CO_2-Emissionen durch Langlebigkeit.

Die Visualisierung der Green Software Foundation[38] beschreibt wie bei Hardware durch Verlängerung der Lebensdauer pro Jahr weniger Emissionen durch Embodied Emissions zugrunde gelegt werden.





Um das Konzept der zeitlichen Attribuierung besser nachzuvollziehen, sei ein komplettes Rechenbeispiel in der Publikation von Transparency for Software Climate Impact[39] von Arne Tarara, Max Schulze und Stefan Kruijer zu empfehlen.

Ein vollständig durchgerechnetes Beispiel findet sich in Anhang 1.

[38] https://greensoftware.foundation/
[39] https://publication2023.bits-und-baeume.org/redefining-progress/transparency-for-software-climate-impact/

4. Wann wird gemessen

4.1 Messen als Teil des Software Engineering Prozess

Das Messen der Energieeffizienz von IT-Infrastruktur und Software ist kein Selbstzweck. Wann gemessen wird, ist entscheidend und hängt wesentlich von der Zielsetzung der Messung ab.

Grundsätzlich kann die Messung von Energieverbräuchen / CO_2-Emissionen in allen Phasen des Softwareentwicklungsprozesses erfolgen:

- Analyse- und Designphase
- Entwicklungsphase und Testphase
- Deployment- und Betriebsphase

Die einzelnen Phasen mit ihren Herausforderungen werden im Folgenden vorgestellt.

Analyse- und Designphase

In frühen Projektphasen können prototypische Messungen helfen, verschiedene Technologien oder Architekturvarianten hinsichtlich ihrer Energieeffizienz zu vergleichen. Diese Erkenntnisse fließen dann in Architekturentscheidungen ein. Beispielsweise könnten KI-Modelle verschiedener Größe und Typs miteinander verglichen werden. Ziel ist es, möglichst frühzeitig eine fundierte Entscheidung über die am besten geeignete Technologie oder Architektur zu treffen. Durch frühzeitige Analysen lassen sich potenzielle Probleme identifizieren und beheben, bevor sie zu größeren Herausforderungen werden.
Herausforderungen beim Messen:

- Nur Schätzungen möglich
- Zielplattform und konkretes Nutzungsverhalten noch unbekannt
- Hauptverbraucher können nur schwer vorhergesagt werden

Entwicklungsphase und Testphase

Während der Softwareentwicklung und dem anschließenden Testen können Entwickler:innen (z. B. auf dem eigenen Rechner) erste Messungen durchführen, z. B. um die Auswirkungen von Codeänderungen auf die Performance und den Energieverbrauch einzelner Funktionen zu evaluieren. Diese Phasen eignen sich besonders gut, um zu analysieren, wie sich Energie- und damit CO_2-Emissionen in typischen Anwendungsszenarien über die Zeit verändert haben (Regressionstest). Testumgebungen ermöglichen es, reproduzierbare Workloads auszuführen und gezielt Messdaten zu sammeln. Ziel ist es Tendenzen frühzeitig zu erkennen.

Herausforderungen beim Messen:

- Testumgebung unterscheidet sich von der Produktionsumgebung
- Testumgebung muss realistische Szenarien abbilden
- Messungen verbrauchen Energie – wie regelmäßig muss gemessen werden?
- Vergleichbarkeit
    - Erstellung wiederholbarer Nutzungsszenarien
    - Nutzung der gleichen Konfiguration (Hardware, Datenbank-Füllgrad, Eingabe-Parameter, etc.)
    - Seiteneffekte vermeiden (Hintergrundprozesse, dynamische CPU-Frequenz, Energiesparmodus, etc.)

Tools, die sich für die Entwicklungsphase und Testphase eignen, werden in der Tool-Landkarte[40] insbesondere mit den Kategorien „Component“ und „Code“ abgebildet. Falls ein Monitoring der Testumgebung stattfinden soll, bieten sich auch Tools aus der

[40] https://landscape.bundesverband-green-software.de/

Kategorie „Infrastructure & Cluster Level“ an. Falls eine KI-Anwendung entwickelt wird, lohnt sich auch ein Blick in die Kategorie „Artificial Intelligence“.

Deployment- und Betriebsphase

Hier kann die DevOps-Infrastruktur sowie die Produktivumgebung überwacht werden. Daten aus dem laufenden Betrieb unterstützen die Wartung und Optimierung der Software. Sie liefern realistische Informationen über den tatsächlichen Energieverbrauch und können helfen die Hauptverbraucher im Gesamtsystem zu identifizieren. Ziel ist es getroffene Maßnahmen mit der Realität abzugleichen und Potentiale für Energie und CO_2-Einsparungen im Gesamtsystem zu finden.

Herausforderungen beim Messen:

- Monitoring-Tools verbrauchen selbst Energie
- Schwankende Systemlast und externe Faktoren
- Identifikation der Hauptverbraucher in komplexen Gesamtsystemen
- Ermittlung der CO_2-Emissionen einzelner Nutzerinteraktionen
- DevOps-Infrastruktur für Monitoring aufbauen und warten - Kosten
- In Cloudumgebungen und durch Virtualisierung werden genaue Messungen erschwert

Für die Deployment- und Betriebsphase bieten sich insbesondere Tools der Kategorie „Infrastructure & Cluster Level“ aus der Tool-Landkarte[41] an. Für eine akkurate Erfassung der CO_2-Emissionen sind die Tools aus der Kategorie „Databases“ mit ihren Unterkategorien „Grid Carbon Intensity API Service“ und „IT Emission Inventory“ hilfreich. Soll die Erfassung ganzheitlich und organisationsweit erfolgen, lohnt sich ein Blick auf die Tools der Kategorie „Data Aggregation“.

[41] https://landscape.bundesverband-green-software.de/

4.2 Identifikation von Energie-Hotspots durch Messungen

In der Entwicklung/Test- und Deployment/Betrieb-Phase spielt für das Optimieren das Messen eine entscheidende Rolle. Insbesondere in diesen Phasen ist ein wichtiges Ziel, *Hotspots* im System zu identifizieren – also Systemteile, die besonders energieintensiv sind. Diese sollten möglichst früh erkannt und gezielt analysiert werden, um Optimierungspotenziale mit minimalem Aufwand und maximalem Nutzen zu realisieren.

Merkmale typischer Energie-Hotspots sind:

- Häufig genutzte Systembestandteile
- Hoher Energieverbrauch bei der Nutzung

Ein sinnvoller Ansatz ist dabei, mit einer Messung des Gesamtsystems (z. B. Cloud-Infrastruktur) zu beginnen, Hotspots zu identifizieren und wenn nötig anschließend auf die Detailanalyse einzelner Komponenten und Funktionen überzugehen, die ein hohes Einsparpotential bieten.

Beispiel: Eine Google-Suchanfrage, die auch nur um ein Hundertstel Prozent effizienter verarbeitet wird, spart im globalen Maßstab mehr CO_2 ein als eine einmal jährlich genutzte, rechenintensive Admin-Anwendung zu optimieren.

Um Hotspots zu identifizieren und gezielt zu optimieren, bieten sich zwei bewährte Ansätze an:

1. Messung in der Test- oder Produktivumgebung
2. Messung mittels Regressionstests

1. Messung in der Test- oder Produktivumgebung

Durch das Monitoring eines real eingesetzten Systems oder einer produktionsnahen Testumgebung kann ermittelt werden, welche Systemkomponenten wie viel Energie verbrauchen und somit CO_2 emittieren.

Diese Messungen liefern eine fundierte Grundlage, um gezielt dort zu optimieren, wo der größte Verbrauch entsteht – oder um sicherzustellen, dass sich bestehende Hotspots durch Updates nicht verschlechtert haben sowie getroffene Maßnahmen mit der Realität abzugleichen. Insbesondere bei großen Systemen ermöglicht diese Herangehensweise, die Bereiche mit dem größten Optimierungspotenzial zu identifizieren und den Überblick über den Energieverbrauch und die Emissionen des gesamten Systems zu behalten.

2. Messung mittels Regressionstests

Eine weitere Möglichkeit ist das frühzeitige Messen im Entwicklungsprozess mittels Regressionstests. Dabei wird regelmäßig der aktuelle Entwicklungsstand mit realitätsnahen Nutzungsszenarien durchlaufen und der Energieverbrauch gemessen (z.B. in der CI/CD Pipeline).

So lassen sich Veränderungen im Energieverbrauch einzelner Funktionen oder Komponenten frühzeitig erkennen, sodass bereits während der Entwicklung gegengesteuert werden kann, bevor ineffizienter Code in Produktion geht.

Anmerkung: Besonderheit bei KI-Projekten

Bei KI-Projekten gibt es eine weitere Phase, die Optimierungspotential bietet: das Training der Modelle. Die Trainingsphase kann bereits einen signifikanten Energieverbrauch haben. Entsprechend sollten Messung bereits beim Training erfolgen, um Optimierungspotentiale identifizieren zu können.

5. Fazit & Ausblick

Das Messen der Umweltauswirkungen von Software ist kein abstraktes Zukunftsthema, wie dieses Whitepaper zeigt, sondern heute bereits möglich und notwendig! Nur was gemessen wird, kann gezielt optimiert werden – und so der ökologische Fußabdruck von Software reduziert werden.

Dabei ist klar: Es gibt noch Herausforderungen – von fehlender Standardisierung bis hin zu praktischen Fragen der Umsetzung. Doch genau hier liegt die Chance, Erfahrungen zu sammeln, voneinander zu lernen und das Feld gemeinsam weiterzuentwickeln.

Ein Blick auf unsere Tool-Landkarte[42] zeigt, dass das Green-Software-Ökosystem rasant wächst und bereits viele Ansätze und Werkzeuge bietet. Erste Ansätze zur Zertifizierung – wie der *Blaue Engel für Software*[43] – eröffnen zudem die Möglichkeit, messbare Erfolge sichtbar zu machen und Vertrauen zu schaffen.

Jetzt ist der richtige Zeitpunkt:

Beginnen Sie mit dem Messen und werden Sie Teil der Green Software Bewegung.

[42] https://landscape.bundesverband-green-software.de/
[43] https://www.blauer-engel.de/de/produktwelt/software

Anhang 1

Komplettes Beispiel für eine SCI-Berechnung mit zeitlicher Attribuierung
Bestellung in einem Online-Shop mit Backup

R (Anwendungsfall): Eine Bestellung im Online-Shop.
=> Die Bestellung dauert 10 Minuten.
=> In den 10 Minuten fand eine Bestellung statt
=> Die Daten müssen 10 Jahre steuerlich aufbewahrt werden.
=> Es werden 100 MB an Daten übertragen

I (Stromnetzintensität): Server in Deutschland.
=> Jahresdurchschnitt Strommix in Deutschland = 334 g CO_2/kWh (electricitymaps.com)

E (Energie): Muss gemessen oder geschätzt werden.
Server: In diesem Beispiel wurde gemessen. Bei der Bestellung lag der durchschnittliche Stromverbrauch bei 30 W.
=> 30 W * 10 min = 0,005 kWh

Backup: Es handelt sich um eine SSD (1000 GB) mit laut Typenschild 10 W Leistungsaufnahme. Auf einer SSD können wir 10.000.000 Kunden ablegen
=> 10 W * 10 Jahre (4*365*24) / 10.000.000 Kunden = 0,03504 kWh

Netzwerk: 0,1 GB (100 MB) * 0,04106063 kWh / GB = 0,004106063 kWh
=> Wir nutzen hier das Energy Intensity Modell für Netzwerk-Energie-Verbrauch mit dem Koeffizienten 0,04106063 kWh/GB (Details: https://www.green-coding.io/co2-formulas/)

M (Embodied Emissions):
Server: 352 kg CO_2 (laut DataVizta; 4 Jahre Lebensdauer angenommen)
=> Zeitliche Attribuierung – 4 Jahre Laufzeit – Eine Bestellung dauert 10 Minuten
=> (352 kg CO_2) * (10 Minuten / (4*365*24*60 Minuten))*1000 = 1,674 g CO_2

Backup: 30 kg CO_2 (laut Boavizta - 4 Jahre Lebensdauer angenommen)
=> 30 kg CO_2 * (10/4) Jahre / 10.0000.000 Kunden = 0,0075 g CO_2

SCI
(0,005 kWh + 0,03504 kWh + 0,004106063 kWH) * 334 g CO_2/kWh + 1,674 gCO_2 + 0,0075 g CO_2 = 16,43 g CO_2 pro Bestellung