FAKULTÄT FÜR
INFORMATIK

Otto-von-Guericke-Universität Magdeburg

Fakultät für Informatik

Institut für Technische und Betriebliche Informationssysteme

Thema:

Energieeffizienz in der Webentwicklung
Untersuchung des Stromverbrauchs einer Webanwendung bei unterschiedlicher Lastverteilung

Bachelorarbeit

Themensteller: Prof. Dr. Hans-Knud Arndt

Vorgelegt von: Florian Kleinert

Abgabetermin: 03.05.2022

Hinweis zur Formulierung:

In dieser Arbeit wird aus Gründen der besseren Lesbarkeit in der Regel das generische Maskulinum verwendet. Entsprechende Begriffe gelten im Sinne der Gleichbehandlung grundsätzlich für alle Geschlechter. Die verkürzte Sprachform beinhaltet keine Wertung.

Abstract

Nahezu alle Produkte, Systeme und Tätigkeiten hinterlassen einen ökologischen Fußabdruck. Aus diesem Grund wurde Nachhaltigkeit zu einer zentralen Anforderung der heutigen Zeit. Zwar gilt dies für Softwareprodukte bisher nur in einem geringeren Ausmaß. Nichtsdestotrotz lenken aktuelle Entwicklungen, wie beispielsweise die Einführung des Blauen Engels als Label für energie- und ressourceneffiziente Software, kontinuierlich mehr Aufmerksamkeit auf diese Thematik. Ein Schwerpunkt von Untersuchungen zu dieser Thematik liegt in der Energieeffizienz dieser Anwendungen. Da Software in unterschiedlichen Bereichen in vielen verschiedenen Architekturen zum Einsatz kommt, sind vielfältige Einflussfaktoren auf den jeweiligen Stromverbrauch denkbar. Einen speziellen Anwendungsfall bieten hierbei Client-Server-Architekturen. Nicht nur die Implementierungsweise der Algorithmen kann hier Einfluss auf den Energiebedarf ausüben, sondern auch der Ort an dem die Berechnungen ausgeführt werden. In dieser Ausarbeitung wurde daher untersucht, welchen Einfluss eine unterschiedliche Lastverteilung auf die Energieeffizienz dieser ausüben kann. Die erhaltenen Resultate legen nahe, dass teils gravierende Unterschiede durch verschiedene Berechnungsorte entstehen. Wobei diese Differenzen auf multiple Faktoren zurückzuführen sind. Wissen darüber ermöglicht nicht nur eine genauere Bewertung der Energieeffizienz von Webanwendungen, sondern kann auch als ein Grundsatz nachhaltiger Webentwicklung dienen und Entwickler dabei unterstützen den ökologischen Fußabdruck ihrer Anwendungen zu reduzieren.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Bachelorarbeit unterstützt, motiviert und an mich geglaubt haben.

Besonderer Dank gilt hierbei zunächst Prof. Dr. rer. pol. habil. Hans-Knud Arndt für die Betreuung der Arbeit sowie die Möglichkeit, sie in dieser Form zu absolvieren. Ebenso möchte ich Regiocom SE, allen voran Lars Plagemann, aber auch allen anderen Beteiligten für die Bereitstellung des Arbeitsplatzes, der Messtechnik und der Unterstützung während dieser Zeit, die in dieser Form nicht selbstverständlich war, danken.
Abschließend möchte ich mich auch bei meinen Freunden sowie meiner Familie für ihre Unterstützung während des gesamten Studiums bedanken.

Inhaltsverzeichnis

Abkürzungsverzeichnis

CNS	Consumption Near Sweet Spot
CPU	Central Processing Unit
DEA	Datenerfassung und Auswertung
DRAM	Dynamic Random Access Memory
IKT	Informations- und Kommunikationstechnologie
LT	Lasttreiber
PUE	Power Usage Effectiveness
SNMP	Simple Network Management Protocol
SUT	System Under Test

Abbildungsverzeichnis

Tabellenverzeichnis

Diagrammverzeichnis

Formelverzeichnis

1. Einleitung

1.1 Motivation

Durch den globalen Klimawandel entwickelt sich bereits seit längerer Zeit ein weltweit steigendes Interesse an Nachhaltigkeit und Energieeffizienz in allen Bereichen der Wirtschaft.

Dies rückte auch die Entwicklung nachhaltiger IKT-Anwendungen immer mehr in den Vordergrund, da diese einen nicht vernachlässigbaren Einfluss auf die Umwelt haben.[1] Der größte Fokus liegt dabei neben dem Einsparen von Kosten auf der Reduzierung des Energieverbrauchs.[2] Es ist daher unumgänglich auch den Einfluss der Software zu betrachten. Zwar wird Strom zunächst für Hardware benötigt, ausgelöst wird dieser Verbrauch allerdings durch die zugehörige Software.[3] Nichtsdestotrotz wird erst seit wenigen Jahren versucht, den Fokus von Green IT auf den Einfluss jener, nicht physischer Komponenten zu lenken. Obwohl durch die laufende Forschung Hardware immer effizienter wurde und somit gleiche Algorithmen mit weniger Stromverbrauch bewältigen konnten, steigt der Energieverbrauch des gesamten IKT-Sektors Jahr für Jahr weiter an.[4] Innerhalb dieses Sektors wird ein Großteil des existierenden Verbrauchs durch auf dem Internet basierende Dienste verursacht.[5] Beispielsweise konnte durch Untersuchungen gezeigt werden, dass allein die transportierte Datenmenge sich innerhalb von fünf Jahren um den Faktor 4,5 erhöht hat.[6] Der verzeichnete Anstieg lässt sich durch den Rebound-Effekt erklären. Dieser besagt, dass erhöhte Ressourceneffizienz zu einer erhöhten Nachfrage und letztendlich zu weniger Einsparungen als erwartet führt.[7] Vereinfacht gesagt bedeutet dies, dass

[1] Vgl. Hilty, L.M., Arnfalk, P., Erdmann, L., Goodman, J., Lehmann, M., Wäger, P.A., „The relevance of information and communication technologies for environmental sustainability – A prospective simulation study“, Jg. 21, Nr. 11, 2006, S. 1618–1629.
[2] Vgl. Molla, A., „GITAM: A Model for the Adoption of Green IT“, ACIS 2008 proceedings, 2008, S. 64.
[3] Vgl. Gröger, J., Köhler, A., Naumann, S., Filler, A., Guldner, A., Kern, E., Hilty, L., Maksimov, Y., „Entwicklung und Anwendung von Bewertungsgrundlagen für ressourceneffiziente Software unter Berücksichtigung bestehender Methodik - Abschlussbericht“, 2018, 156 Seiten.
[4] Vgl. Morley, J., Widdicks, K., Hazas, M., „Digitalisation, energy and data demand: The impact of Internet traffic on overall and peak electricity consumption“, in Energy Research & Social Science, Jg. 38, 2018, S. 128–137.
[5] Vgl. Stobbe, L., Proske, M., Zedel, H., Hintemann, R., Clausen, J., Beucker, S., „Entwicklung des IKT-bedingten Strombedarfs in Deutschland“, Berlin 2015.
[6] Vgl. Bundesnetzagentur. Tätigkeitsbericht Telekommunikation 2016/2017.
[7] Vgl. Hilty, L., Lohmann, W., Behrendt, S., Evers-Wölk, M., Fichter, K., Hintemann, R., „Grüne Software: Ermittlung und Erschließung von Umweltschutzpotenzialen der Informations- und Kommunikationstechnik (Green IT)“, UBA TEXTE, 2015, 68 Seiten.

wegen zunehmend besserer Hardware neu entwickelte Software kontinuierlich aufgeblähter, ressourcenintensiver und energieineffizienter wird. Dies trägt wesentlich zum stetig steigenden Energiebedarf bei. Bestätigt wird dies durch Untersuchungen von Philippot et. al, in denen festgestellt wurde, dass nicht nur große Unterschiede im Energiebedarf verschiedener Webseiten existieren, sondern vor allem neuere Seiten mehr verbrauchen.[8] Daher sollte die Entwicklung ressourceneffizienter Webanwendungen stärker in den Fokus rücken. Insbesondere, da bereits kleine Änderungen im Quellcode große Auswirkungen auf die Energieeffizienz haben können.[9, 10] All diese Probleme und Entwicklungen sollten Softwareentwicklern Anlass dazu geben, sich mehr mit den Auswirkungen ihres Codes auf den Energieverbrauch eines Systems zu beschäftigen und diesen zu verstehen, da nicht zuletzt sie es sind, die einen maßgeblichen Einfluss auf den Stromverbrauch ihrer Anwendung haben.

1.2 Zielsetzung

Ziel dieser Ausarbeitung soll es sein, das Verständnis für den Stromverbrauch einer Webanwendung zu erhöhen, indem versucht wird, die oft gestellte Frage nach der optimalen Verteilung von Arbeitsaufwänden zwischen Frontend und Backend zu beantworten. Diese unterschiedlichen Ansätze werden in der vorliegenden Ausarbeitung als Lastverteilung benannt. Die Arbeit nähert sich dieser Fragestellung aus dem Blickwinkel der Energieeffizienz. Mögliche Verschlechterungen der Nutzererfahrung durch möglicherweise längere Ladezeiten werden nicht in die Ergebnisse mit einbezogen. Zusätzlich werden weitere Faktoren nachhaltiger Softwareprodukte, abseits ihres Stromverbrauchs, nicht untersucht.

Hierfür wird mithilfe der Frameworks *React* und *Microsoft ASP.NET* eine Client-Server-Struktur entwickelt, die in der Lage ist, realitätsnahe Arbeitsaufwände für Webanwendungen sowohl clientseitig als auch serverseitig auszuführen. Dieses Programm kann daraufhin in beiden Szenarien mithilfe eines System Under Tests (SUT) auf Energieeffizienz verglichen werden. Die Auswertung der erlangten Werte erfolgt als *Blackbox*. Ferner sollen die dabei erlangten Werte auch Rückschlüsse auf

[8] Vgl. Philippot, O., Anglade, A., Leboucq, T., „Characterization of the energy consumption of websites: Impact of website implementation on resource consumption“, 2nd International Conference on ICT for Sustainability, S. 171–178.

[9] Vgl. Hindle, A., „Green mining: A methodology of relating software change to power consumption“, 2012 9th IEEE Working Conference on Mining Software Repositories (MSR), IEEE, 2012, S. 78–87.

[10] Vgl. Singh, J., Naik, K., Mahinthan, V., „Impact of Developer Choices on Energy Consumption of Software on Servers“, in Procedia Computer Science, Jg. 62, 2015, S. 385–394.

praktische Anwendungsfälle zulassen und eine Empfehlung für die reale Entwicklung von Webapplikationen ermöglichen.

Durch diese Vorgehensweise sollen innerhalb dieser Ausarbeitung die folgenden Forschungsfragen beantwortet werden:

F1: *Wie groß ist der Einfluss der Lastverteilung auf die Energieeffizienz einer Webanwendung?*

F2: *Welche Faktoren sind ursächlich für die entstehenden Unterschiede?*

F3: *Wie ergibt sich eine optimale Lastverteilung für Webanwendungen in praktischen Anwendungsszenarien*

1.3 Aufbau der Arbeit

Um die Zielsetzung mitsamt der Forschungsfragen korrekt bearbeiten zu können ist die vorliegende Arbeit wie folgt aufgebaut:

Im **zweiten Kapitel** wird die theoretische Wissensbasis für Stromverbräuche innerhalb eines Computersystems beschrieben. Hierbei werden zunächst Grundannahmen bezüglich des Energiebedarfs erläutert. Weiterhin werden die bedeutsamsten Erkenntnisse anderer Untersuchungen genannt. Daraufhin werden die Komponenten, die hauptverantwortlich für den Strombedarf einer Webanwendung sind, aufgezählt und zueinander ins Verhältnis gesetzt.

Kapitel drei behandelt daran anschließendes Grundlagenwissen über Möglichkeiten den Energiebedarf von Software zu ermitteln. Dies umfasst grundlegende Charakteristiken, sowie Metriken, Ziele und die dafür möglichen Messgeräte. Ebenso wird ein beispielhafter Aufbau mitsamt Ablauf dargelegt, der exemplarisch für verschiedenste Anwendungsszenarien befolgt werden kann.

Der spezifische Aufbau und Ablauf der in dieser Arbeit durchgeführten Messungen werden in **Kapitel vier** beschrieben. Hierbei werden hauptsächlich Änderungen am in Kapitel 3 vorgestellten Vorgehen aufgezeigt, die durch die Besonderheiten von Webanwendungen und dem Ziel der Untersuchung bedingt sind. Des Weiteren erfolgt eine Beschreibung der entwickelten Anwendung, mitsamt der genutzten Frameworks

und ausgewählter Arbeitsaufwände, sowie der Berechnungsmethoden, die für einen optimalen Erkenntnisgewinn entwickelt wurden.

Anschließend werden die durch die Messung erhaltenen Daten in **Kapitel fünf** aufgezeigt. Hierbei erfolgt eine detaillierte Auflistung der Messwerte für alle durchgeführten Phasen. Zusätzlich werden die gemessenen Stromverbräuche je Phase graphisch dargestellt.

Kapitel sechs umfasst im Anschluss an die erhaltenen Daten eine detaillierte Diskussion dieser. Dies beinhaltet eine spezifische Auswertung, Gegenüberstellung und Bewertung. Zudem findet in diesem Abschnitt die Beantwortung der gestellten Forschungsfragen anhand der Messwerte statt.

Zuletzt werden in **Kapitel sieben** die gewonnenen Erkenntnisse sowohl zusammengefasst als auch kritisch betrachtet. In Ergänzung dazu wird ein Ausblick auf weitere interessante Themen in diesem Bereich gegeben. Abschließend wird ein Fazit zu dieser Ausarbeitung gezogen.

2. Grundlagen: Stromverbrauch

2.1 Energiebedarf

Aufgrund der hohen Komplexität heutzutage eingesetzter Geräte gestaltet sich eine Schätzung des Stromverbrauchs von Software schwierig. Dabei resultieren Probleme unter anderem aus der Entstehungsweise dieses Verbrauchs. So beansprucht Software zunächst nur die entsprechende Hardware. Diese wiederum verursacht den gesamten Energieverbrauch des Systems. Gemessene Verbräuche umfassen somit stets den Einfluss beider Faktoren. Durch das Zusammenspiel verschiedenster Hardware-Komponenten kann der Bedarf an Energie nicht auf spezifische Bauteile zurückgeführt werden. Auf Grund dieser Verknüpfung ist die Ursache für gemessene Verbräuche nur durch weitgehende Untersuchungen lokalisierbar. Für eine grundlegende Unterteilung des Gesamtverbrauchs eines Computersystems schlägt Acar deshalb vor, diesen zunächst in zwei Arten einzuteilen. Den statischen und den dynamischen Stromverbrauch.[11]

Der *statische Strom* beschreibt dabei den Energiebedarf, der notwendig ist, um das System und alle enthaltenen Komponenten funktionsfähig zu halten.[12] Somit wird dieser maßgeblich durch die Hardware und deren Funktionsweise bestimmt. Dabei wird der Einfluss von Software nur betrachtet, wenn es sich um zwingend notwendige Prozesse, wie beispielsweise das Betriebssystem, handelt. Stromverbräuche dieser Art fallen unabhängig von der aktuellen Auslastung des Systems an.[13]

Im Gegensatz dazu steht der *dynamische Strom*, der sich auf den Verbrauch bezieht, der während der Bearbeitung von Aufgaben zusätzlich zum statischen Verbrauch benötigt wird. Auf diesem Anteil des gesamten Energiebedarfs eines Computersystems sollte stets der Hauptfokus liegen, wenn es darum geht, den durch Software induzierten Energiebedarf zu ermitteln.[14]

Wie sich anhand dieser Einteilung bereits erkennen lässt, besitzt jedes Computersystem einen Grundverbrauch, der nicht durch Software beeinflusst wird und in späteren Berechnungen oder Messungen beachtet werden muss. Dabei sollte ebenso berücksichtigt werden, dass moderne Hardwarekomponenten, vor allem im

[11] Vgl. ACAR, H., Software development methodology in a Green IT environment, 2017.
[12] Vgl. ebd.
[13] Vgl. ebd.
[14] Vgl. ebd.

mobilen Bereich, dazu in der Lage sind, ihren Stromverbrauch dynamisch an die benötigte Rechenleistung anzupassen. Dies bedeutet, dass gewonnene Erkenntnisse abhängig von der entsprechend verwendeten Hardware und den durchgeführten Berechnungen sein können.[15] Zusätzlich existieren heutzutage, je nach Hersteller und Produktreihe, große Unterschiede in der Energieeffizienz unterschiedlicher Komponenten.[16] Somit sind Vergleiche ähnlicher Algorithmen bezüglich deren Energiebedarf zwar möglich, allerdings lassen sich nur Tendenzen ableiten. Eine exakte Benennung der Unterschiede behält lediglich für das genutzte System seine Gültigkeit und ist nicht zwangsläufig auf andere Systeme mit anderer Hardware in gleichem Maße übertragbar.

2.2 Erkenntnisse bisheriger Forschungen

Durch die steigende Bedeutung der Nachhaltigkeit von Anwendungen konnten durchgeführte Forschungen bereits eine breite Wissensbasis aufbauen. Ein weites Forschungsfeld bilden dabei Veröffentlichungen zur Energieeffizienz von Anwendungen. Insbesondere der Fokus auf Messtechniken sollte hierbei erwähnt werden. Die Erkenntnisse aus diesen Untersuchungen zeigen, dass durch die Nutzung eines SUT nahezu alle Applikationen mit minimalen Anpassungen im Versuchsaufbau untersucht werden können.[17] Durch die vielen denkbaren Anwendungsgebiete für Software mussten jedoch Einschränkungen für Vergleiche unterschiedlicher Anwendungen festgestellt werden. Durch die mannigfaltigen Einsatzgebiete und Funktionalitäten ergibt sich, dass ausschließlich Produkte mit ähnlicher oder gleicher Funktionalität miteinander verglichen werden können. Nichtsdestotrotz konnten Untersuchungen, die unter dieser Annahme verliefen, deutliche Unterschiede feststellen. Exemplarisch hierfür stehen die Erkenntnisse von Kern, Guldner und Naumann, die dies durch Messungen von verschiedenen Textverarbeitungsprogrammen belegen konnten (s. Abbildung 1). Ähnliche Resultate konnten bereits bei Messungen anderer Anwendungsfälle, wie Webbrowsern oder Content Management Systemen belegt werden.[18]

[15] Vgl. Pinto, G., Castor, F., Liu, Y.D., „Mining questions about software energy consumption“, in Devanbu, P., Kim, S., Pinzger, M. (Hg.), Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014, ACM Press, New York, New York, USA 2014, S. 22–31.
[16] Vgl. Gröger et al., 2018.
[17] Vgl. ebd.
[18] Vgl. ebd.

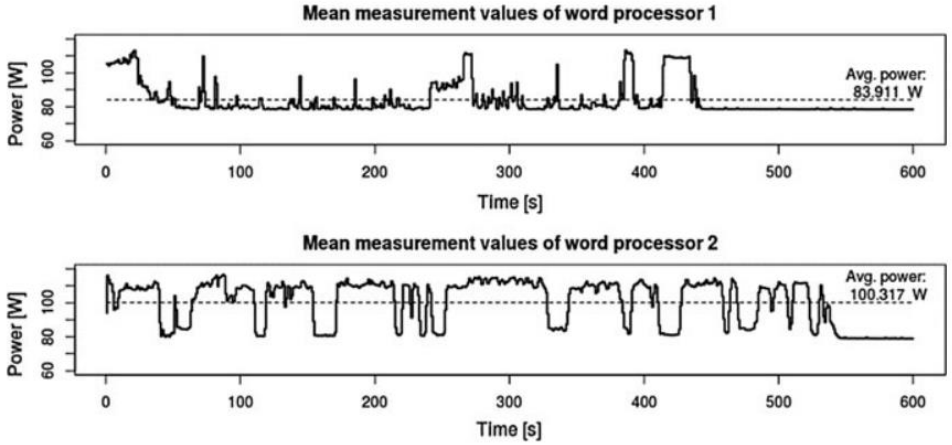

Abbildung 1: Gegenüberstellung der Stromverbräuche zweier Anwendungen zur Textverarbeitung
Quelle: Kern et al., 2019

Da Stromverbrauch durch Software erst im Zuge dieser Veröffentlichungen in den Fokus rückte, war es notwendig mehr Aufmerksamkeit für diese Problematik zu gewinnen. So konnte Pang in einer Umfrage unter Programmierern ermitteln, dass lediglich 18 Prozent der Befragten die von ihnen entwickelte Software anhand ihrer Energieeffizienz bewerten und dies zudem kein von Anwendern oder Kunden häufig gefordertes Qualitätskriterium darstellt.[19] Eine mögliche Ursache dafür liegt auch im geringen Wissen über Nachhaltigkeit von Softwareprodukten. Gründe hierfür können vielseitiger Natur sein. So ist die Nachhaltigkeit von Applikationen einerseits weiterhin unterrepräsentiert in Lehrplänen[20], andererseits existieren auch kaum feste Vorgaben, die Aspekte bei der Beurteilung von nachhaltiger Software zu beachten sind. Aus dieser Motivation heraus entwickelten Gröger et al. im Auftrag des Umweltbundesamtes Bewertungsgrundlagen „für ressourceneffiziente Software“.[21] Diese dienten im weiteren Verlauf auch als Grundlage für die Erstellung des Umweltzeichens Blauer Engel für Software. Mit diesem konnte bereits ein erstes Produkt ausgezeichnet werden.[22] In näherer Zukunft können Nutzer diese Auszeichnung in ihrer Entscheidungsfindung berücksichtigen. Auch die Entwicklung von Software erhielt im Zuge dessen klare Anhaltspunkte in allen Bereichen der Nachhaltigkeit. Dieses vermutete Verhalten der Nutzer konnte zusätzlich bereits in

[19] Vgl. Pang, C., Hindle, A., Adams, B., Hassan, A.E., „What Do Programmers Know about Software Energy Consumption?“, in IEEE Software, Jg. 33, Nr. 3, 2016, S. 83–89.
[20] Vgl. Torre, D., Procaccianti, G., Fucci, D., Lutovac, S., Scanniello, G., „On the Presence of Green and Sustainable Software Engineering in Higher Education Curricula“, 2017 IEEE/ACM 1st International Workshop on Software Engineering Curricula for Millennials (SECM), IEEE, 2017, S. 54–60.
[21] Vgl. Gröger et al., 2018.
[22] Vgl. Torre et al., 2017.

einer Veröffentlichung von Kern bestätigt werden, in welcher gezeigt wurde, dass eine Mehrheit der Nutzer ein als nachhaltig ausgezeichnetes Produkt seinem nicht zertifizierten Pendant vorziehen würden, insofern es keine Einschränkungen der Funktionalität aufweist.[23]

Im Zusammenhang damit ist es jedoch notwendig anzumerken, dass eine stromsparende Implementierung zwar einen wesentlichen Bestandteil einer nachhaltigen Applikation darstellt, eine solche allerdings nicht alleinig ausmacht. In mehreren Studien mit unterschiedlichen Ansätzen wurde aus diesem Grund versucht Qualitätsaspekte zu ermitteln. Eine dabei häufig verbreitete Annahme ist, dass ein Softwareprodukt über seinen gesamten Lebenszyklus hinweg möglichst geringe Auswirkungen auf die Umwelt haben sollte.[24] Innerhalb dieses Lebenszyklus wird die beanspruchte Hardware und deren Verbrauch natürlicher Ressourcen beachtet. Eine Übersicht über die genannten Zusammenhänge findet sich in Abbildung 2.

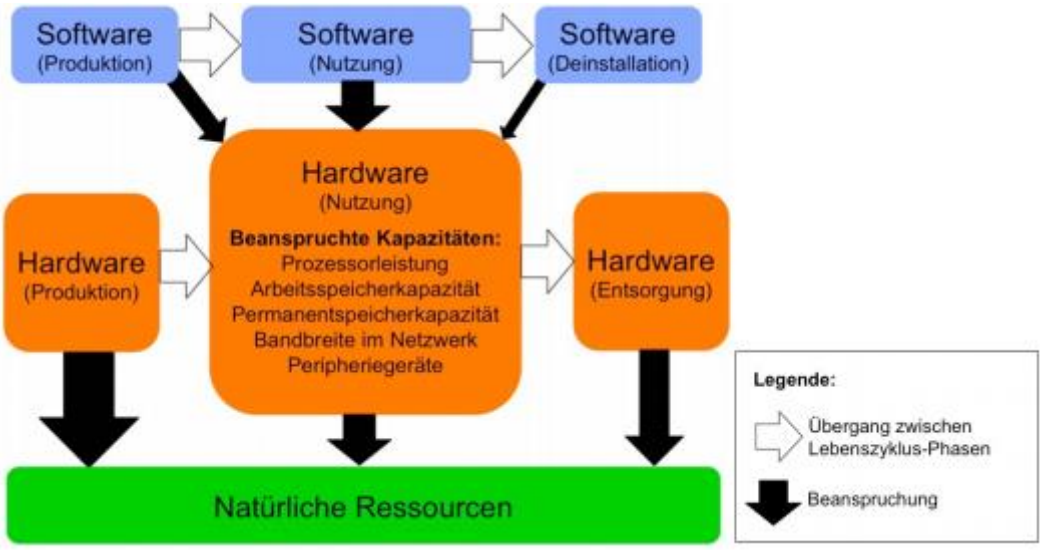

Abbildung 2: Beanspruchung natürlicher Ressourcen durch Hardware und Software

Quelle: Gröger et al., 2018

Um so viele dieser Faktoren wie möglich zu berücksichtigen und auch Produktverantwortliche zu unterstützen, war die Entwicklung von Frameworks zur nachhaltigen Softwareentwicklung ein weiterer wesentlicher Teil veröffentlichter Arbeiten. Ein dabei häufig genutzter Ansatz liegt im Greensoft-Modell, das die Entwicklung eines Softwareprodukts über den gesamten Lebenszyklus hinweg

[23] Vgl. Kern, E., „Green Computing, Green Software, and Its Characteristics: Awareness, Rating, Challenges“, in Otjacques, B., Hitzelberger, P., Naumann, S., Wohlgemuth, V. (Hg.), From Science to Society, Springer International Publishing, Cham 2018, S. 263–273.
[24] Vgl. Gröger et al., 2018.

unterstützt und somit Entwickler, Administratoren und die Nutzer der Software miteinschließt.[25] Dabei werden in vier Phasen sowohl direkte als auch indirekte Einflüsse bei der Erstellung, Aufrechterhaltung und Nutzung der Anwendung betrachtet.

Zwar lassen sich nahezu alle der genannten Erkenntnisse auch auf Client-Server Architekturen übertragen, jedoch fehlt es insbesondere in diesem Bereich an der Benennung von spezifischen Faktoren, die Einfluss auf die Nachhaltigkeit ausüben. Allen voran gilt dies bei Betrachtungen der Energieeffizienz. So bilden Stromverbräuche bisher zumeist nur in batteriebetriebenen Systemen wie Smartphones oder Laptops einen Schwerpunkt in der Entwicklung. Nichtsdestotrotz entsteht auch bei der Nutzung und Entwicklung von Webanwendungen ein ökologischer Fußabdruck, der reduziert werden muss. Insbesondere in diesem Bereich mangelt es zum aktuellen Zeitpunkt sowohl an einer Zertifizierung wie dem Blauen Engel als auch an Richtlinien oder Tools für Entwickler.

2.3 Stromverbrauch einer Webanwendung

Um den Stromverbrauch einer Client-Server-Architektur zu verstehen, ist es zunächst notwendig zu betrachten in welchen Bereichen der Energieverbrauch auftritt. So benötigen einerseits die Systeme von Client und Server selbst Strom, andererseits sollte auch die Datenübertragung über das jeweilige Netzwerk und die entsprechende Visualisierung nicht vernachlässigt werden. Da sich die für den Stromverbrauch relevante Hardware eines Servers von der eines Clients nicht etwa durch die genutzten Komponenten, sondern meist nur durch deren prozentualen Anteil am gesamten Energiebedarf unterscheidet (s. Abbildungen 3 und 4), werden diese im Folgenden lediglich eingeordnet und es findet keine separate Trennung zwischen Client und Server statt. Eingegrenzt wird dieses Kapitel durch das Nichtbeachten von Stromverbräuchen durch nicht ressourcenschonendes Verhalten im Umgang mit den Geräten und anderen extern anfallenden Verbräuchen, wie beispielsweise der Kühlung eines Servers. Ebenso werden Gründe für den statischen Stromverbrauch von genannten Komponenten, die durch deren Bauweise oder Architektur bedingt sind, nicht weiter ausgeführt. Verbesserungen in diesen Bereichen obliegen den Herstellern der Bauteile. Diese Eingrenzungen werden vorgenommen, da diese Art

[25] Vgl. Mahmoud, S.S., Ahmad, I., „A green model for sustainable software engineering“, in International Journal of Software Engineering and Its Applications, Jg. 7, Nr. 4, 2013, S. 55–74.

von Verbräuchen nicht durch die entsprechende Software beeinflusst werden kann, und daher für diese Ausarbeitung nicht weiter relevant ist.

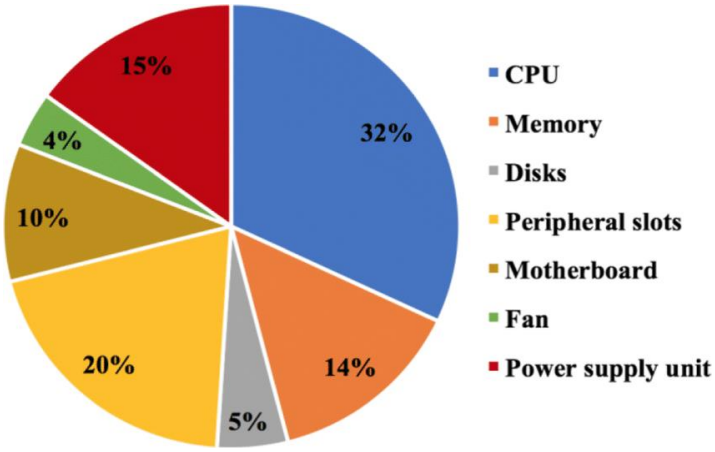

Abbildung 3: Stromverbrauch eines typischen Servers
Quelle: Vasques et al., 2017

Der gesamte Stromverbrauch eines Computersystems wird dabei maßgeblich durch die CPU, den Arbeitsspeicher und die Festplatte verursacht. Zusätzlich fällt sowohl in Servern als auch in den Rechnern von Clients statischer Energieverbrauch durch die Bauweise der Komponenten an. Dies umfasst Energie für den Chipsatz beziehungsweise das Motherboard oder Verluste bei der Stromversorgung (s. Abbildung 3 und 4). Ein nur beim Client auftretender Verbrauch liegt in der Darstellung der Webapplikation und im Betreiben des Bildschirms. Für den Client stellt dies sogar die größte Quelle für Stromverbräuche dar. Für Webanwendungen ist weiterhin der Einfluss des Netzwerks und einer eventuell betriebenen Datenbank zu betrachten. Zwar verbrauchen all diese Komponenten Energie, dennoch ist es notwendig ihren entsprechenden Einfluss zu kennen, um die Bedeutung der Komponenten ins Verhältnis setzen zu können. Zusätzlich kann dadurch besser beurteilt werden, in welchem Umfang die einzelnen Komponenten durch Software beeinflussbar sind.

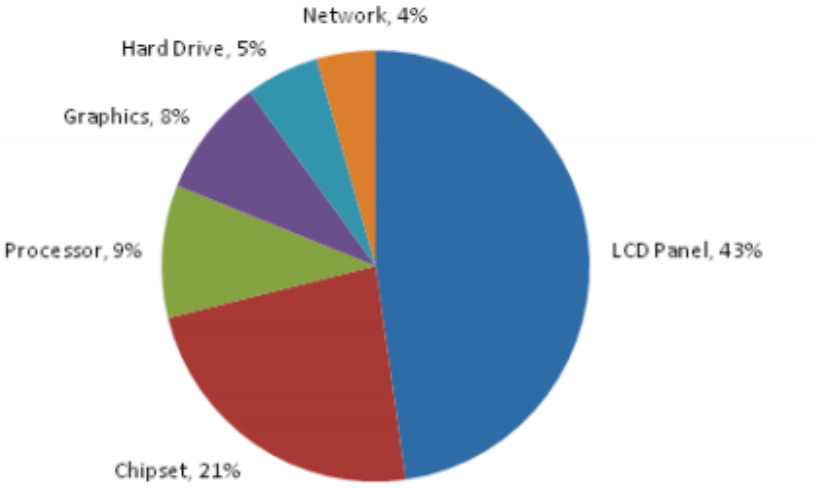

Abbildung 4: Stromverbrauch eines typischen Rechners

Quelle: Acar, 2017

2.3.1 Prozessor

Wie in Abbildung 3 erkennbar ist, wird der Großteil des Stromverbrauchs durch den Prozessor verursacht. Bei Servern konnte herausgefunden werden, dass gut ein Drittel des kompletten Energiebedarfs auf die CPU zurückgeführt werden kann. Im Gegensatz dazu ist dieser Anteil bei Rechnern oder Laptops wesentlich geringer (s. Abbildung 4). Wie groß dieser genau ist, hängt dennoch sowohl vom verwendeten System und den darin enthaltenen Bauteilen als auch der ausgeführten Software ab. In Studien konnte beobachtet werden, dass selbst nominell identische Prozessoren Unterschiede im Energiebedarf aufweisen. So konnten im Rahmen dieser Messungen Unterschiede von bis zu 29,6% festgestellt werden.[26] Wie groß diese Unterschiede sind, ist dabei jedoch deutlich abhängig von der aktuellen Auslastung der Architektur.[27] Ebenso konnte durch von Kistowski aufgezeigt werden, dass sich verschiedene Rechenleistungen in ihrem Energiebedarf signifikant unterscheiden.[28] Dies gilt, selbst wenn sie die gleiche Ressource beanspruchen.[29] Allerdings ist davon auszugehen, dass neuere Prozessoren nicht nur immer leistungsfähiger, sondern auch immer

[26] Vgl. v. Kistowski, J., Block, H., Beckett, J., Spradling, C., Lange, K.-D., Kounev, S., „Variations in CPU Power Consumption“, in Avritzer, A., Iosup, A., Zhu, X., Becker, S. (Hg.), Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering, ACM, New York, NY, USA 2016, S. 147–158.
[27] Vgl. ebd.
[28] Vgl. v. Kistowski, J., Block, H., Beckett, J., Lange, K.-D., Arnold, J.A., Kounev, S., „Analysis of the Influences on Server Power Consumption and Energy Efficiency for CPU-Intensive Workloads“, in John, L.K., Smith, C.U., Sachs, K., Lladó, C.M. (Hg.), Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering, ACM, New York, NY, USA 2015, S. 223–234.
[29] Vgl. v. Kistowski et al., 2015.

energieeffizienter werden, wodurch diese Unterschiede in neueren Bauteilen deutlich geringer sind.[30] Aus diesen Untersuchungen lässt sich schließen, dass Prozessoren einen großen Einfluss auf den dynamischen Stromverbrauch eines Systems haben. Dies wird durch Untersuchungen von Capra bestätigt. Bei jenen wurde festgestellt, dass die CPU sogar für den deutlich größten Teil des dynamischen Stromverbrauchs verantwortlich ist.[31] Aus diesen Gründen sollte bei Betrachtungen der Energieeffizienz von Software stets der größte Fokus auf dem Einfluss von Prozessoren liegen.

2.3.2 Speicher

Ein nächster großer Teil des Stromverbrauchs wird durch verschiedene Speicherarten verursacht. Die Unterscheidung findet hierbei zwischen Arbeitsspeicher und Festplatte statt. Von diesen beiden Arten entsteht der hauptsächliche Energiebedarf durch den Arbeitsspeicher (s. Abbildung 3 und 4), der am häufigsten in Form von DRAM genutzt wird. Ein bei vielen Webanwendungen zusätzlich anfallender Punkt liegt, insofern benötigt, auch in der Nutzung von Datenbanken. Obwohl der Anteil der Speichereinheiten am gesamten Energieverbrauch in manchen Systemen sogar in der gleichen Größenordnung liegen wie der Prozessor, konnten Untersuchungen zeigen, dass der größte Teil davon dem statischen Stromverbrauch zuzuordnen ist. Diese Erkenntnisse beruhen auf der Grundlage, dass DRAM konstant wieder aufgefrischt wird und Festplatten sich kontinuierlich drehen. Diese Aktionen werden unabhängig vom anfallenden Arbeitsaufwand durchgeführt. Somit wird der Stromverbrauch nicht maßgeblich durch die Software beeinflusst.[32] Auf Grund dessen empfiehlt Capra die Arbeitslast von Algorithmen vermehrt auf Speichermedien zu verschieben, wodurch ein geringerer dynamischer Verbrauch entstehen soll.[33] Für eine Reduzierung des gesamten Stromverbrauchs von Datenbanksystemen schlagen Karyakin et. al hingegen eine Fokussierung auf den statischen Verbrauch vor.[34] Es konnte dennoch herausgefunden werden, dass ein, wenn auch geringer, dynamischer Stromverbrauch

[30] Vgl. v. Kistowski et al., 2015.
[31] Capra, E., Francalanci, C., Slaughter, S.A., „Measuring Application Software Energy Efficiency“, in IT Professional, Jg. 14, Nr. 2, 2012, S. 54–61.
[32] Vgl. ebd.
[33] Vgl. ebd.
[34] Karyakin, A., Salem, K., „An analysis of memory power consumption in database systems“, Proceedings of the 13th International Workshop on Data Management on New Hardware, ACM, New York, NY, USA 2017, S. 1–9.

bei Speichereinheiten entstehen kann.[35] Untersuchungen zeigten zudem eine signifikante Zunahme des Anteils, den DRAM in immer neueren Systemen einnimmt.[36] Auf Grund dieser Entwicklungen ist ein stetig steigender Anteil der Speicher am dynamischen Stromverbrauch dennoch wahrscheinlich. Auch wenn dieser geringer als der der Prozessoren ist, sollte er bei Untersuchungen des Energiebedarfs stets miteinbezogen werden.

2.3.3 Andere stromverbrauchende Komponenten

Die restlichen Komponenten werden im Folgenden, aufgrund ihrer ähnlichen Bedeutung für die Thematik, innerhalb eines Kapitels zusammengefasst. Hierbei werden Bestandteile des jeweiligen Systems, wie die Lüftung, der Chipsatz, das Mainboard oder auch die Stromzufuhr betrachtet. Bestandteile dieser Art finden sich sowohl auf dem Client als auch den jeweiligen Servern wieder. Ihre Bedeutung für den Stromverbrauch besteht darin, dass der durch sie induzierte Bedarf nahezu ausschließlich dem statischen Anteil des Gesamtverbrauchs zuzuordnen ist. Zudem sind es diese Komponenten, die gemeinsam während des Leerlaufs eines Systems den größten Teil der Energie verbrauchen.[37] Zwar unterscheiden sich diese Komponenten in ihrem jeweiligen Anteil, dennoch kann keiner von diesen direkt durch den Nutzer oder durch Software beeinflusst werden. Hierbei gilt, dass lediglich die entsprechenden Hersteller der Teile Einfluss auf deren Energieeffizienz haben. Zudem ist es zum Teil aufgrund der Bauweise nicht möglich die Verbräuche der einzelnen Komponenten voneinander zu trennen.[38] Durch diese Fakten werden sie in dieser Ausarbeitung nicht detaillierter behandelt.

2.3.4 Stromverbrauch im Netzwerk

Den Stromverbrauch der Netzwerkübertragung zu ermitteln ist eine komplexe Aufgabe. Zwar wurden bisher viele Versuche unternommen, um zu schätzen wie viel Energie pro versendetem Gigabyte verbraucht wird, leider unterscheiden sich diese

[35] Vgl. Mahesri, A., Vardhan, V., „Power Consumption Breakdown on a Modern Laptop“, in Falsafi, B., Vijaykumar, T.N. (Hg.), Power-aware computer systems: Fourth International Workshop, PACS 2004, Portland, OR, USA, December 5, 2004 revised selected papers, Springer, Berlin, New York 2005, S. 165–180.
[36] Vgl. Paul Dempsey, „Rambus CEO calls for collaboration and an architectural focus for memory“, https://www.techdesignforums.com/blog/2013/11/04/rambus-ceo-collaboration-architecture-memory/, Stand: 13.04.2022.
[37] Vgl. Mahesri et al., 2004.
[38] Vgl. ebd.

aber nicht nur in ihrer Methodik, sondern auch in ihren Ergebnissen stark. So werden erhaltene Ergebnisse maßgeblich durch die Größe des vorhandenen Netzwerks, dem Jahr der Untersuchung und getroffenen Annahmen beeinflusst.[39] Die jeweils erhaltenen Werte unterscheiden sich so um bis zu 7 kWh/GB.[40] Eine Angleichung dieser Berechnungen von Aslan zeigte jedoch, dass selbst mit verschiedener Methodik ähnliche Ergebnisse erzielt werden können. Durch diese Angleichung konnte 2015 ein Verbrauch von 0,06 kWh je übermitteltem Gigabyte bestimmt werden, der von mehreren anderen Studien unterstützt wird. Allerdings ließ sich ebenfalls die Tendenz ableiten, dass sich der pro versendetem Gigabyte im Internet verursachte Stromverbrauch alle zwei Jahre etwa halbiert.[41] Somit kann der aktuelle Energiebedarf, der durch Datenverkehr während der Nutzung einer Webanwendung verursacht wird, nur geschätzt werden. Generell lässt sich die Aussage treffen, dass der Stromverbrauch im Netzwerk von der Größe der verschickten Daten abhängt und mit zunehmender Anzahl dieser oder mit zunehmender Anzahl von Aufrufen der Seite ebenfalls ansteigt. Möglichkeiten zur Reduzierung des Verbrauchs sind selbst trotz geringem Einfluss auf das genutzte Netzwerk existent. Für eine Verringerung dieses Bedarfs sollte das Hauptaugenmerk zunächst auf einer Reduzierung der versendeten Datenmenge liegen. Somit ist die Einführung und Nutzung effizienter Datenformate eine grundlegende Anforderung an nachhaltige Webapplikationen.[42] In Ergänzung dazu stellten Dick et. al bereits 12 Prinzipien vor, mit welchen die Größe versandter Dateien effektiv reduziert werden kann.[43] Abschließend sollte noch angemerkt werden, dass auch bei sinkendem Verbrauch pro Gigabyte, der Gesamtverbrauch linear mit der Anzahl der Nutzer ansteigt und deshalb besonders für häufig genutzte Services schnell die Ursache beträchtlicher Stromverbräuche sein kann.

[39] Vgl. Aslan, J., Mayers, K., Koomey, J.G., France, C., „Electricity Intensity of Internet Data Transmission: Untangling the Estimates“, in Journal of Industrial Ecology, Jg. 22, Nr. 4, 2018, S. 785–798.
[40] Vgl. ebd.
[41] Vgl. ebd.
[42] Vgl. Gröger et al., 2018.
[43] Vgl. Dick, M., Naumann, S., Held, A., „GREEN WEB ENGINEERING - A Set of Principles to Support the Development and Operation of “Green” Websites and their Utilization during a Website’s Life Cycle“, Proceedings of the 6th International Conference on Web Information Systems and Technology, SciTePress - Science and and Technology Publications, 2010, S. 48–55.

2.3.5 Stromverbrauch der Benutzeroberfläche

Ein letzter zu beachtender Faktor beim Stromverbrauch einer Webanwendung stellt die Nutzeroberfläche dar. Leider wird diese in einem Großteil der Untersuchungen nicht betrachtet, da meist nur Rechner ohne Monitore oder einzelne Hardware-Komponenten Beachtung finden und es sich hierbei oft um ein externes Gerät handelt.[44] Nichtsdestotrotz ist eine graphische Oberfläche zwangsläufig immer Teil einer Webanwendung und sollte somit auch in entsprechenden Betrachtungen vorkommen. So verursacht das Betreiben des Bildschirms sogar den meisten Strom innerhalb eines Rechners.[45] Allerdings muss auch hierbei wieder der größte Teil des Energiebedarfs dem statischen Stromverbrauch zugeordnet werden. Zudem wurde festgestellt, dass die exakten Stromverbräuche von Monitoren je nach Modell und Hersteller, selbst bei gleicher Größe, stark schwanken können.[46] Zudem kann weder vorausgesetzt werden, dass jeder Nutzer einen energieeffizienten Monitor besitzt, noch können Aussagen über das verwendete Modell oder die Größe des Geräts gegeben werden. Feststeht dennoch, dass ein Teil des durch den Bildschirm verursachten Stromverbrauchs auf Medien, aber auch auf die Farbauswahl des Designs zurückzuführen ist. Daher kann eine überarbeitete Farbauswahl auf Webseiten gemeinsam mit verbesserten Bildern und Videos auf Nutzeroberflächen bereits für Einsparungen im Stromverbrauch von vier bis sechs Watt pro Stunde und Nutzer führen.[47] Auch wenn diese Werte auf den ersten Blick gering wirken, gilt es zu beachten, dass diese Reduktion bei jedem Nutzer auftritt.

Zusätzlich dazu gilt bei Stromverbrauch durch Nutzeroberflächen, dass mögliche Einsparungen linear mit der Nutzungsdauer der Anwendungen skalieren. Weiterhin sollte beachtet werden, dass klare Designs die Nutzungszeit einer Anwendung und somit auch deren Stromverbrauch reduzieren kann. Durch ein Zusammenspiel dieser Faktoren können auch hierbei große Energieeinsparungen möglich gemacht werden.

[44] Vgl. Pandikumar, S., Kabilan, S.P., „Principles and Holistic Design of Green Web Portal“, in International Journal of Computer Applications, Jg. 65, Nr. 9, 2013, S. 23–29.
[45] Vgl. ebd.
[46] Vgl. ebd.
[47] Vgl. ebd.

2.4 Zusammenfassung des Kapitels

Die im vorherigen Kapitel genannten Komponenten sind die maßgeblichen Verursacher des Stromverbrauchs innerhalb einer Webanwendung. Mithilfe der dabei gesammelten Informationen können nun Messmethoden für unterschiedlichste Untersuchungen verglichen und bewertet werden. Ebenso ermöglicht diese Auflistung ein besseres Verständnis über die daraus erhaltenen Ergebnisse. Dadurch können einerseits Messungen besser geplant und zusätzlich auch Resultate besser verstanden werden. Ebenso ist es hilfreich für die Umsetzung, der daraus entstehenden Erkenntnisse, in die Praxis, da lediglich die Gewissheit über erhöhten Stromverbrauch einer Software keine direkten Ansätze zur Verbesserung bietet.

3. Grundlagen: Messung des Verbrauchs

3.1 Charakteristiken einer Messung der Energieeffizienz

Für die Durchführung von Messungen zur Ermittlung des Stromverbrauchs einer Software kann, aufgrund der mannigfaltigen Anwendungsmöglichkeiten jener, kein fester Ablaufplan angegeben werden. Um dennoch die Qualität einer solchen sicherstellen zu können, benötigt es feste Kriterien. Aus diesem Grund definierte von Kistowski vier relevante Charakteristiken. Diese wurden speziell für Messungen des Stromverbrauchs von Software entwickelt. Allerdings handelt sind dabei lediglich um leicht abgewandelte Formen von den für Untersuchungen der Performanz relevanten Faktoren.[48] Alle dieser Punkte sollten für eine solche Versuchsdurchführung beachtet und bestmöglich umgesetzt werden.

Reproduzierbarkeit

Es ist notwendig bei erneuter Durchführung des Experiments unter gleichen Bedingungen nahezu identische Ergebnisse mit nur minimalen Unterschieden zu erhalten. Um dies zu erreichen, sollten zudem die relevantesten Faktoren der Systemumgebung aufgelistet werden.[49]

Fairness

Es müssen alle relevanten Spezifikationen für den Aufbau der Versuchsumgebung angegeben werden. Dies umfasst neben Messgeräten auch Details über die verwendete Hardware. Des Weiteren sollten alle Komponenten, die Stromverbräuche verursachen ebenfalls aufgelistet werden. Dieser Schritt ist notwendig, da somit sichergestellt werden kann, dass die durchgeführten Untersuchungen unter gleichem Aufbau auch von anderen Durchführenden wiederholt werden können.[50]

Verifizierbarkeit

Die Korrektheit der erhaltenen Ergebnisse muss überprüfbar sein. Dafür müssen mögliche Messungenauigkeiten angegeben werden. Im Optimalfall werden diese entweder direkt durch den Hersteller der Messgeräte oder, falls Software-

[48] Vgl. v. Kistowski, J., Lange, K.-D., Arnold, J.A., Sharma, S., Pais, J., Block, H., „Measuring and Benchmarking Power Consumption and Energy Efficiency“, in Wolter, K., Knottenbelt, W., van Hoorn, A., Nambiar, M., Koziolek, H. (Hg.), Companion of the 2018 ACM/SPEC International Conference on Performance Engineering, ACM, New York, NY, USA 2018, S. 57–65.
[49] Vgl. ebd.
[50] Vgl. ebd.

Messmethoden angewandt werden, direkt durch die genutzten Tools angegeben. Falls diese nicht der Fall ist, müssen diese eigenständig bestimmt werden.[51]

Verwendbarkeit

Durch die zunehmende Komplexität von Messtechnik und entsprechenden Softwarelösungen entsteht der Bedarf vorgenommene Konfigurationen sowie die Nutzung des Messgerätes zu beschreiben. Ebenfalls sollten Hinweise zur Bedienung dieser Geräte gegeben werden. Sollten unterstützende Anwendungen für die Konfiguration der Messtechnik oder für das Auslesen der Messwerte genutzt wurden sein, gilt es diese ebenfalls in die Beschreibung mit aufzunehmen. Dies sorgt dafür, dass selbst Nutzer, die über wenig bis keine Erfahrung in diesem Bereich verfügen, in der Lage sind die Messungen nachzuvollziehen oder nachzustellen.[52]

In Ergänzung zu diesen Punkten gibt von Kistowski für Untersuchungen der Performanz von Software einen weiteren Punkt an, der auch bei Messungen der Energieeffizienz nicht vernachlässigt werden darf. Dabei handelt es sich potentiell sogar um die bedeutsamste Charakteristik verlässlicher Untersuchungen.[53]

Relevanz

Die auf dem System ausgeführten Berechnungen beziehungsweise die simulierte Last müssen speziell für das Ziel der Untersuchungen und der entsprechenden Software ausgewählt werden. Ebenso sollte der ausgewählte Arbeitsaufwand des Systems eine möglichst realitätsnahe Auslastung hervorrufen. Dadurch wird sichergestellt, dass die gewonnen Erkenntnisse auch in der praktischen Anwendung verwertbar sind.[54]

3.2 Metriken

Metriken bilden die Grundlage eines jeden Vergleichs. Aus diesem Grund sollte stets vor Beginn einer Untersuchung Klarheit darüber geschaffen werden, welche Bewertungskriterien angewendet werden und ob diese für das jeweilige Szenario passend sind. Für Messungen der Energieeffizienz von Software ist dies besonders relevant. So wurde Software bislang mit einem Fokus auf bestmögliche Performanz

[51] Vgl. v. Kistowski et al., 2018.
[52] Vgl. ebd.
[53] Vgl. v. Kistowski, J., Arnold, J.A., Huppler, K., Lange, K.-D., Henning, J.L., Cao, P., „How to Build a Benchmark“, in John, L.K., Smith, C.U., Sachs, K., Lladó, C.M. (Hg.), Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering, ACM, New York, NY, USA 2015, S. 333–336.
[54] Vgl. ebd.

entwickelt. Jedoch konnte herausgefunden werden, dass Optimierungen einer Software für Energieeffizienz nicht zwangsläufig mit dem Ziel einer gesteigerten Performanz vereinbar sind.[55] Dies unterstützt den Bedarf nach neuen Metriken. Diese sollten möglichst genaue Auskunft darüber geben, wie stromsparend eine Anwendung betrieben werden kann und ob, respektive wie groß, Unterschiede zu anderen Softwareprodukten sind. Ebenso muss bedacht werden, dass Softwareprodukte in vielen unterschiedlichen Bereichen eingesetzt werden können und sich daher auch in ihrer Funktionalität deutlich unterscheiden. Zusätzlich gilt, dass jegliche gemessenen Verbräuche stets abhängig vom ausführenden System sind.[56] Aus diesen Gründen lassen sich Erkenntnisse, die durch die Verwendung der im folgenden aufgezeigten Metriken gewonnen wurden, ausschließlich vergleichen wenn die entsprechenden Softwareprodukte eine ähnliche Funktionalität aufweisen und auf einem nahezu identischen System ausgeführt werden.

Eine weitere für Webapplikationen notwendige Bedingung liegt in dem Fakt, dass das Bewertungskriterium alle beteiligten Komponenten, sowohl Client als auch Server, miteinbeziehen und auch auf diese anwendbar sein sollte. Verbesserungen dürfen daher nicht nur in einzelnen Bereichen betrachtet werden. Vielmehr sollten sie für den gesamten Betrieb der Applikation, also end-to-end, angestrebt werden.[57] Somit kann keine Anwendung von getrennten Bewertungskriterien, für beispielsweise Backend und Frontend, stattfinden. Ein Beispiel für eine solche, findet sich in der oft verwendeten Metrik der Power Usage Effectiveness (PUE).[58] Obwohl diese ISO standardisiert und weit verbreitet ist, werden dabei nur Stromverbräuche der Datencenter-Infrastruktur bewertet (s. Abbildung 5). Für eine allumfassende Bewertung von Webapplikationen ist dies allerdings nicht ausreichend, da der durch Software induzierte Energiebedarf nicht betrachtet wird.[59]

[55] Vgl. Pinto et al., 2014.
[56] Siehe Kapitel 2.1.
[57] Vgl. Grosskop, K., „PUE for end users - Are you interested in more than bread toasting?“, in Bunse, C., Gottschalk, M., Naumann, S., Winter, A. (Hg.), 2nd Workshop EASED@BUIS 2013 Energy Aware Software-Engineering and Development, 2013, S. 15–16.
[58] Vgl. ebd.
[59] Vgl. ebd.

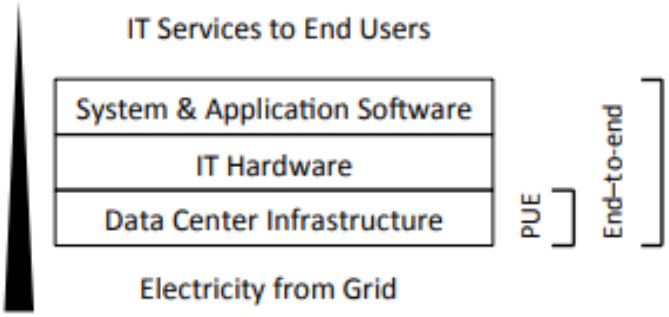

Abbildung 5: PUE Metrik im Vergleich zu end-to-end Ansätzen
Quelle: Grosskop, 2013

Im Folgenden werden deshalb zwei Metriken vorgestellt, die diese Bedingungen erfüllen und für die Planung und Bewertung von Messvorgängen für Webanwendungen relevant sind. Da beide Metriken sich inhaltlich ergänzen respektive aufeinander aufbauen stellen sie für diese Ausarbeitung verwendbare Bewertungskriterien dar. Es sei dennoch angemerkt, dass aktuell viel Aufmerksamkeit auf diesem Bereich der Forschung liegt und daher mehrere mögliche Ansätze existieren. Somit handelt es sich bei den hier vorgestellten Kriterien lediglich um einen Ausschnitt.

3.2.1 Energieeffizienz

Johann et. al empfehlen als allgemeingültige Metrik die Energieeffizienz. Diese wird dabei wie folgt definiert:

$$Energieeffizienz = \frac{erbrachte\ nützliche\ Leistung}{Energieverbrauch}$$

Formel 1: Energieeffizienz nach Johann
Quelle: Johann et al., 2012

Der dabei geforderte Energieverbrauch kann je nach Art der Software und Ziel der Untersuchung festgelegt werden. Für eine möglichst aussagekräftige Bewertung wird jedoch empfohlen, den gesamten Energiebedarf des Systems, das die Berechnungen ausführt, zu messen.[60] Im Fall einer Webapplikation würde dies sowohl den Verbrauch des Clients, sowie den des Servers und einer möglicherweise genutzten Datenbank einschließen.

[60] Vgl. Johann et al., 2012.

Die benannte erbrachte nützliche Leistung wird benötigt, da es die vielfältigen Anwendungsszenarien von Computersystemen und deren Software nicht zulassen ausschließlich anhand des Energiebedarfs die Energieeffizienz zu ermitteln. Deshalb muss vor jeder Anwendung dieser Metrik klar definiert werden, was genau erbrachte nützliche Leistung im jeweiligen Kontext bedeutet. Das dafür empfohlene Vorgehen sieht vor, zunächst Standardnutzungsszenarien für das entsprechende System festzuhalten. In diesen Szenarien werden Arbeitsaufwände definiert, die, in allen zu untersuchenden Applikationen, häufig in der Realität ausgeführte Operationen darstellen. Mithilfe dieser Szenarien ist es möglich, während der Ausführung dieser, Stromverbräuche zu ermitteln und mit den Resultaten anderer Anwendungen oder Implementierungen zu vergleichen, in welchen sich ähnliche Standardnutzungsszenarien wiederfinden. Als ein praktisches Beispiel dafür gibt Johann ein Tool an, das Graphen generiert. In diesem Zusammenhang kann erbrachte nützliche Leistung als die Anzahl der generierten Graphen beschrieben werden.[61]

Durch die Bildung des Quotienten dieser Leistung und der dafür benötigten Energie wird es möglich verschiedenste Anwendungsfälle abzubilden. So ist diese Metrik einerseits für Webapplikationen nutzbar, andererseits konnten beispielsweise auch Unterschiede im Strombedarf verschiedener Textverarbeitungsprogramme aufgezeigt werden.[62] Bei dieser Berechnungsweise gilt, dass je größer der erhaltene Wert ist, desto positiver ist dieser zu bewerten. Hierbei ist zu beachten, dass Verbesserungen der erhaltenen Resultate sowohl durch nachhaltigere Software als auch durch Nutzung effizienterer Hardware bedingt sein können. Daraus ergibt sich für die Nutzung dieser Bewertungsgrundlage für Softwareprodukte der Bedarf, Messungen unter Nutzung möglichst einheitlicher Hardware vorzunehmen, um den Einfluss dieser auf das Endergebnis bestmöglich zu reduzieren. Eine weitere Einschränkung findet sich bei der Bewertung der erhaltenen Resultate. Denn es ist nicht möglich, ein theoretisch bestmögliches Ergebnis festzulegen. Aus diesem Grund kann keine Einschätzung darüber getroffen werden, ab wann ein Softwareprodukt als energieeffizient respektive energieineffizient klassifiziert werden kann. Außerhalb von Vergleichen ist somit keine Einschätzung einzelner Produkte mithilfe dieser Metrik möglich.

[61] Vgl. Johann, T., Dick, M., Naumann, S., Kern, E., „How to measure energy-efficiency of software: Metrics and measurement results“, 2012 First International Workshop on Green and Sustainable Software (GREENS), IEEE, 2012, S. 51–54.
[62] Vgl. Kern et al., 2019.

Durch die Option sowohl die erbrachte Leistung als auch den einbezogenen Energieverbrauch flexibel zu definieren, ist es zudem notwendig zu erwähnen, dass die erhaltenen Werte nur vergleichbar sind, falls das gewählte Vorgehen innerhalb der Standardnutzungsszenarien einheitlich ist und der durchgeführte Rechenaufwand eine ähnliche Funktionalität bietet.
Durch all diese Einschränkungen zeigt sich, dass die Anwendbarkeit dieser Bewertungsgrundlage stark eingeschränkt ist. Nichtsdestotrotz bieten die dabei erhaltenen Resultate eine hohe Aussagekraft, falls alle Anforderungen erfüllt wurden und Unsicherheitsfaktoren bestmöglich ausgeschlossen werden konnten.[63]

3.2.2 Verbrauch nahe des optimalen Bereichs

Eine weitere Herausforderung der gerade dargelegten Metrik liegt darin, dass es für internetbasierte Dienste nicht ausreichend ist, die Energieeffizienz eines Serversystems ausschließlich auf Basis der pro nützlicher Arbeit benötigten Energie zu beurteilen. Dies beruht auf der Tatsache, dass Server mehrere Anfragen unterschiedlicher Nutzer parallel abarbeiten müssen. Abhängig von der dabei anfallenden Rechenlast verändert sich auch der Stromverbrauch eines Systems (s. Abbildung 6). Für eine allgemeine Aussage über die Effizienz eines solchen Systems müssen diese Berechnungsaufwände daher stets in die entsprechenden Berechnungen miteinbezogen werden. Dabei gilt zu beachten, dass der Energiebedarf nicht linear mit steigender Last ansteigt. Stattdessen ist von einem nicht vernachlässigbaren statischen Anteil auszugehen[64], der durch einen dynamischen Anteil, abhängig von der Last, ergänzt wird. Kombiniert mit der Definition der Energieeffizienz nach Johann aus verrichteter Arbeit durch Energieverbrauch kann der Schluss gezogen werden, dass die Energieeffizienz eines Systems nicht als ein fester Wert angenommen werden kann. Vielmehr ergibt sich dieser aus einer Abhängigkeit mit der jeweiligen Arbeitslast.

[63] Vgl. Gröger et al., 2018.
[64] Siehe Kapitel 2.1.

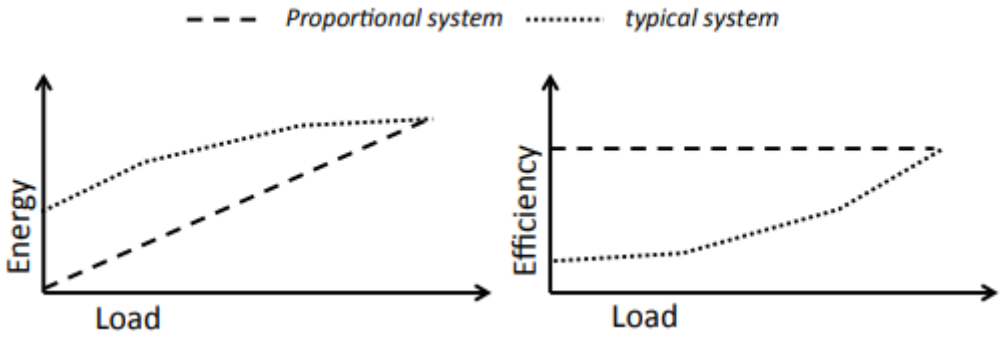

Abbildung 6: Vergleich der Energieeffizienz eines typischen und eines proportionalen Systems
Quelle: Grosskop, 2013

Durch eine Miteinbeziehung der Last verliert die im vorherigen Kapitel beschriebene Bewertungsgrundlage jedoch nicht an Bedeutung. Vielmehr handelt es sich um eine Ergänzung dieser. So wird die Auslastung, zu der die beste Effizienz auftritt, allgemein als „sweet spot“ (deutsch: optimaler Punkt) bezeichnet. Die beschriebene Korrelation verhält sich allerdings stark abhängig von der im System verbauten Hardware.[65] Untersuchungen belegten zudem, dass im Kontrast zu Abbildung 4 der Punkt der höchsten Effizienz nicht zwangsläufig mit der größtmöglichen Last einhergeht.[66] Aus diesen Gründen kann keine generelle Aussage zu der genauen optimalen Belastung von Systemen gegeben werden. Zusätzlich sollte beachtet werden, dass es sich bei der zu tragenden Arbeitslast in der Realität ebenfalls nicht um einen stabilen Faktor handelt, sondern sich diese abhängig von Uhrzeit oder Wochentag verändern kann.

Um diesen Aspekt dennoch in Bewertungen mit einbeziehen zu können, beschreibt Grosskop die Metrik „Consumption Near Sweet Spot“ (CNS, deutsch: Verbrauch Nahe des Optimalen Bereichs).[67] Diese baut auf der Definition der Energieeffizienz von benötigter Energie pro verrichteter Arbeit auf und wird folgendermaßen definiert:

$$CNS = \frac{EU_S}{EU_{avg}}$$

Formel 2: Consumption Near Sweet Spot
Quelle: Grosskop, 2013

[65] Vgl. Götz, S., Ilsche, T., Cardoso, J., Spillner, J., „Software Energy-Efficiency with Sweet Spot Frequencies“, 2014, 10 Seiten.
[66] Vgl. ebd.
[67] Grosskop, 2013.

Dabei gibt EU_S die Energieeffizienz an, die das zu bewertende System im Optimalfall erreichen kann. Es handelt sich dabei um die Energieeffizienz, von welcher sichergestellt ist, dass sie in der Praxis erreicht werden kann. Daher sollte dieser Wert durch Messungen bereits im Vorhinein bestimmt werden. Dabei wird nicht beachtet, ob es theoretisch bessere Werte geben könnte.[68]

EU_{avg} hingegen, beschreibt die durchschnittliche Energieeffizienz, die das System im realen Einsatz aufweist.[69]

Durch die Nutzung dieser Formel wird ein Wert erhalten, der immer zwischen 0 und 1 liegt, wobei 1 einer kontinuierlich optimalen Auslastung entsprechen würde. Für eine Verbesserung der erhaltenen Werte sind dabei zwei Strategien denkbar. Eine erste Zielsetzung kann sein, das System stets nahe dem Bereich einer optimalen Auslastung zu halten. Dies ist mittels Techniken, wie Leistungsoptimierung und Workload-Platzierung, in der Realität bereits umsetzbar.[70] Denkbar erscheint dies vorrangig für große Anwendungen, für deren Betrieb mehrere Serversysteme notwendig sind, von denen auch einige bei Bedarf heruntergefahren werden können. Eine zweite Maßnahme besteht darin, den Energiebedarf des Systems möglichst proportional zu gestalten (s. Abbildung 6). Dies ist in der Praxis allerdings nur schwierig umsetzbar, da dies von einem System im Ruhezustand keinen Stromverbrauch erwarten würde. Dennoch sollte dieses Ziel vorrangig durch Entwickler von Anwendungen verfolgt werden, deren Hardware für eine funktionierende Applikation dauerhaft in Betrieb sein muss.

3.3 Möglichkeiten der Messung

Aktuell existieren bereits viele Ansätze wie der Stromverbrauch eines Computersystems beziehungsweise eines Softwareprodukts gemessen werden kann. Diese wurden jedoch häufig unter verschiedenen Zielsetzungen und Annahmen entwickelt. Aus diesem Grund gibt es teils deutliche Unterschiede, sowohl in der Ausführung, als auch in der Genauigkeit der Ergebnisse. Diese Differenzen zwischen verschiedenen Ansätzen machen eine Unterteilung dieser sinnvoll. Daher werden im folgenden Kapitel verschiedene Ansätze zunächst nach deren Zielsetzung

[68] Vgl. Grosskop, 2013.
[69] Vgl. ebd.
[70] Vgl. ebd.

unterschieden. Daraufhin erfolgt eine Unterteilung der unterschiedlichen Messtechniken.

3.3.1 Ziele von Messungen

Für die Durchführung eines Experiments zum Energiebedarf von Anwendungen ist es zunächst notwendig festzulegen, welches Ziel mit der Untersuchung verfolgt wird. Einerseits ist es denkbar Rückschlüsse auf ineffizienten Quellcode schließen zu wollen, andererseits kann ein System oder ein Softwareprodukt auch als Gesamtheit untersucht werden. Abhängig davon können Messungen grundlegend in zwei Arten eingeteilt werden (s. Abbildung 7).

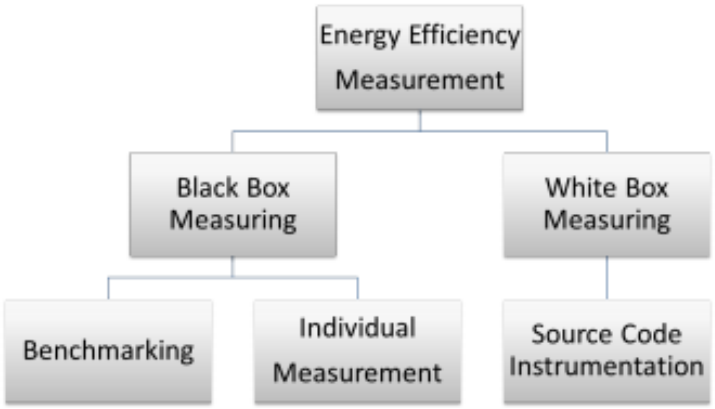

Abbildung 7: Methoden zur Messung der Energieeffizienz

Quelle: Johann et al., 2012

Blackbox Messungen

Diese betrachten den Energieverbrauch eines Computersystems als Einheit und sind die meistgenutzte Form der Untersuchung.[71] Um möglichst genaue Erkenntnisse über die Energieeffizienz einer Software während ihrer entsprechenden Nutzung gewinnen zu können, werden Messungen dieser Form häufig während der Ausführung von Standardnutzungsszenarien (engl. Benchmarking)[72] durchgeführt. Da diese Szenarien abhängig von der Art und dem Anwendungsgebiet der Software sind, können nur Softwareprodukte mit möglichst gleicher Funktionalität verglichen werden. Dennoch besteht die Möglichkeit zur Messung von Energieeffizienz in festgelegten Szenarien, die für die jeweilige Anwendung individuell entwickelt wurden.

Ergebnisse, die aus Messungen dieser Art erhalten werden, betrachten die genutzte Hardware gemeinsam mit der entsprechenden Software.[73] Dadurch ist der statische

[71] Vgl. Johann et al., 2012.
[72] Vgl. Gröger et al., 2018.
[73] Vgl. Johann et al., 2012.

Stromverbrauch nicht mehr direkt vom Dynamischen zu unterscheiden. Aus diesem Grund ist es bei der Auswertung der Resultate auch stets ratsam den Faktor der Ressourceninanspruchnahme der unterschiedlichen Hardware zu beachten. Ein weiteres Problem von Blackbox Messungen besteht darin, dass die erhaltenen Werte keine direkten Rückschlüsse auf energieineffiziente Codestellen zulassen. Zwar können verschiedene Implementierungen einer Software miteinander verglichen werden, dennoch wirkt dies für umfangreiche Codebasen aufgrund des hohen Aufwandes nicht praktikabel. Nichtsdestotrotz ist diese Messmethodik, im Vergleich zu spezifischeren Quellcodeuntersuchungen, mit deutlich weniger Aufwand durchzuführen und liefert für die meisten Anwendungsfälle ausreichend Daten. Darüber hinaus können diese Messungen ohne Wissen über den zugrunde liegenden Quellcode durchgeführt werden. Somit lassen sich auch fremde Softwareprodukte untersuchen.

Whitebox Messungen

Zielsetzungen dieser Art versuchen hingegen ineffiziente Codestellen zu identifizieren. Obwohl dies eine in der Softwareentwicklung gefragte Methodik ist, fehlt Entwicklern häufig noch die Unterstützung durch Tools.[74] Ein Grund dafür liegt in dem Fakt, dass bisher bestehende Coding-Richtlinien nur in Teilen zur Steigerung der Energieeffizienz beitragen. Untersuchungen von Fowlers Techniken zum Refactoring von bestehendem Quellcode zeigten unter anderem, dass nur manche seiner vorgeschlagenen Methoden zu einem geringeren Energiebedarf führten, wohingegen andere diesen sogar erhöhten.[75] Dies verstärkt den Bedarf an Möglichkeiten zur Überprüfung von Quellcode. Das Finden von generellen Richtlinien für die Softwareentwicklung gestaltet sich jedoch sehr schwierig. So existieren Unterschiede im Energiebedarf, zusätzlich zu den bisher aufgezeigten Bereichen, wie verwendeter Hardware, auch zwischen verschiedenen Programmiersprachen.[76]

Zum heutigen Stand existieren bereits mehrere Ansätze für Whitebox Messungen.

[74] Vgl. Manotas, I., Pollock, L., Clause, J., „SEEDS: a software engineer's energy-optimization decision support framework“, in Jalote, P., Briand, L., van der Hoek, A. (Hg.), Proceedings of the 36th International Conference on Software Engineering, ACM, New York, NY, USA 2014, S. 503–514.

[75] Vgl. Park, J.-J., Hong, J.-E., Lee, S.-H., „Investigation for Software Power Consumption of Code Refactoring Techniques“, SEKE 2014, S. 717–722.

[76] Vgl. Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J.P., Saraiva, J., „Energy efficiency across programming languages: how do energy, time, and memory relate?“, in Combemale, B., Mernik, M., Rumpe, B. (Hg.), Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering, ACM, New York, NY, USA 2017, S. 256–267.

Allerdings weisen diese häufig stark unterschiedliche Vorgehensweisen auf. So reichen die Vorschläge vom manuellen Einbauen und Nutzen von Performanz Zählern zur eigenständigen Überprüfung des Codes [77], über Entscheidungsframeworks[78], bis hin zu statistischen Methoden zum Auffinden ineffizienter Codefragmente.[79] Dennoch ist es für vielfältige Entwicklungsprozesse möglich, mit Hilfe dieser oder ähnlicher Methoden Whitebox Messungen durchzuführen und ineffiziente Codestellen zu finden. Durch das Ziel des Aufdeckens ineffizienter Codefragmente kann diese Art der Untersuchung nur durchgeführt werden, falls der dem Programm zugrunde liegende Code einsehbar ist. Fremde Programme sind hierbei somit konsequent ausgeschlossen. Im Gegensatz zu Blackbox Messungen ist es bei dieser Zielsetzung nicht möglich verschiedene Softwareprodukte untereinander zu vergleichen. Stattdessen wird sich ausschließlich auf die zu entwickelnde Anwendung fokussiert, um diese so stromsparend wie möglich zu gestalten.

3.3.2 Messgeräte

Ein weiterer unerlässlicher Punkt für Messungen ist die Auswahl der geeignetsten Messtechnik. Durch die Relevanz der Energieeffizienz von Software in vielen Bereichen der IT, existieren viele Ansätze für die Messung des Stromverbrauchs von Systemen. Unterschiede finden sich dabei vorrangig in der Genauigkeit, den einbezogenen Komponenten und der Komplexität von Einbau und Nutzung, die von der entsprechenden Technik verlangt wird. Acar unterscheidet hierbei in drei Gruppen (s. Abbildung 8). Wobei jede dieser Gruppen für die im vorherigen Abschnitt genannten Ziele von Messungen genutzt werden kann.

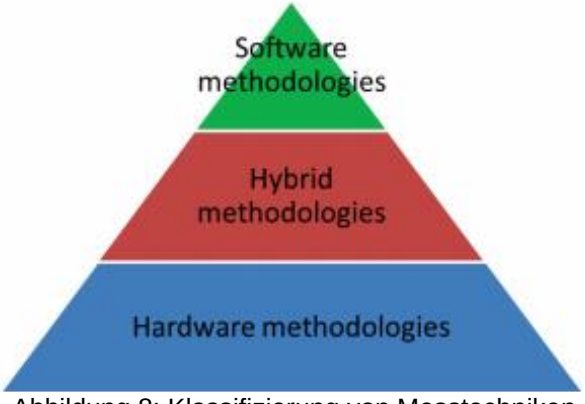

Abbildung 8: Klassifizierung von Messtechniken
Quelle: Acar, 2017

[77] Vgl. Johann et al., 2012.
[78] Vgl. Manotas et al., 2014.
[79] Vgl. Pereira, R., Carção, T., Couto, M., Cunha, J., Fernandes, J.P., Saraiva, J., „SPELLing out energy leaks: Aiding developers locate energy inefficient code“, in Journal of Systems and Software, Jg. 161, 2020, S. 110463.

Die wohl meistgenutzte Methode bildet hierbei die Hardware. Dabei wird ein externes Strommessgerät genutzt, um den Stromverbrauch eines Systems zu erfassen. Messungen auf diese Weise haben den Vorteil, dass die erhaltenen Resultate genauer sind als bei anderen Methoden.[80] Dies beruht auf der Tatsache, dass Messtechnik, in Form von Strommessgeräten, aufgrund der Einbauart, direkt an der Stromversorgung des Systems oder fest im Stromnetz installiert, alle anfallenden Verbräuche zuverlässig aufnehmen kann. Weiterhin besteht die Möglichkeit, über den Einbau spezialisierter Hardware in das Computersystem, Stromverbräuche einzelner spezifischer Komponenten zu ermitteln.[81] Da das Messgerät in keinem direkten Zusammenhang mit der verwendeten Hardware steht, erfolgt der Einbau für sämtliche Computersysteme gleich. Weiterhin kann davon ausgegangen werden, dass Messungen stets innerhalb einer angegebenen Fehlertoleranz liegen, die vom Hersteller angegeben wird. Weiterhin sind viele Produktgruppen von Software damit untersuchbar.[82] Nichtsdestotrotz muss beachtet werden, dass die erhaltenen Werte sowohl statischen als auch dynamischen Verbrauch einschließen. Infolgedessen können Probleme bei Messversuchen von eingebetteten Systemen oder Anwendungen in virtuellen Umgebungen entstehen.[83] Diese Probleme äußern sich zumeist in ungenauen Messwerten durch Hintergrundprozesse im System, die kein Teil der Untersuchungen sein sollten.

Um den Einfluss dieser Faktoren auf das Endergebnis zu verringern, wird häufig ein System Under Test verwendet.[84] Eine weitere Problematik dieses Ansatzes liegt in der Tatsache, dass für spezialisierte Messgeräte Kosten, sowohl für den Erwerb, als auch für die Installation der Hardware anfallen. Abhängig vom verwendeten Gerät kann es zudem vorkommen, dass dieses fest eingebaut werden muss, wodurch Messungen nur an einem Standort durchgeführt werden können und Fachwissen für die Installation benötigt wird.

Eine zweite Möglichkeit der Messtechnik bietet Software. Die Herangehensweise besteht hierbei darin, die im zweiten Kapitel aufgezeigten Komponenten für das

[80] Vgl. Acar, 2017.
[81] Vgl. ebd.
[82] Dick, M., Kern, E., Drangmeister, J., Naumann, S., Johann, T., „Measurement and Rating of Software Induced Energy Consumption of Desktop PCs and Servers“, in Hoffman, H., Kotheimer, O., Feilke, S. (Hg.), Innovations in sharing environmental observations and information, Shaker, Aachen 2011, S. 290–299.
[83] Vgl. ebd.
[84] Siehe Kapitel 3.4.

jeweilige System zu analysieren und eine Berechnungsmethode aufzustellen, mit welcher der Stromverbrauch in Abhängigkeit der Auslastung ermittelt werden kann. Aktuell existieren bereits viele Ansätze wie dies praktisch umgesetzt werden kann.[85] Jedoch wurden diese häufig unter verschiedenen Annahmen entwickelt.[86] So muss bei der Analyse dieser Techniken stets überprüft werden, die Komponenten mit welchem Einfluss in die Berechnungen mit einbezogen werden. Da viele dieser Ansätze davon ausgehen, dass der Prozessor der Hauptverursacher des Stromverbrauchs ist, wird dieser oft als einzige Komponente betrachtet.[87] Nichtsdestotrotz konnte aufgezeigt werden, dass der Prozessor zwar den Hauptverursacher des dynamischen Stromverbrauchs darstellt, dieser aber nur für etwa ein Drittel des gesamten Energiebedarfs eines Systems verantwortlich ist und somit andere Komponenten ebenfalls signifikant zu diesem beitragen.[88] Übereinstimmend dazu konnte durch Studien belegt werden, dass Berechnungsmethoden, die nur den Einfluss einzelner Komponenten betrachten, deutlich ungenauer sind als andere, die mehrere Komponenten heranziehen.[89] Ebenso stehen innerhalb dieser Gruppe beispielsweise Tools wie das Intel Power Gadget[90], das lediglich Stromverbräuche der CPU betrachtet und diese unabhängig von laufenden Prozessen ausgibt, im direkten Vergleich zu Ansätzen wie Joulemeter[91], womit Stromverbräuche virtueller Umgebungen bestimmt werden können. Dadurch wird deutlich, dass durch Techniken dieser Gruppe zwar eine Mehrzahl der Anwendungsgebiete von Software abgedeckt werden können, diese allerdings nicht zwangsläufig verlässliche Ergebnisse liefern und je Anwendungsfall ausgewählt und evaluiert werden müssen. Außerdem verbraucht der Betrieb solcher Software selbst Strom, wodurch Messwerte verfälscht werden können.

Ein Vorteil dieser Herangehensweise liegt darin, dass keine extra Kosten für die Anschaffung neuer Technik anfallen. Ebenso ist die Installation häufig simpler und es wird kein Fachwissen benötigt. Ebenfalls kann diese Installation meist unabhängig vom genutzten System stattfinden. Nichtsdestotrotz ist ein Vergleich verschiedener Untersuchungen, die Softwaremesstechniken verwenden, nur möglich, wenn

[85] Vgl. Acar, 2017.
[86] Vgl. ebd.
[87] Vgl. ebd.
[88] Vgl. Kapitel 2.3.
[89] Vgl. Acar, 2017.
[90] Vgl. McKay, T., Konsor, P.C., Intel® Power Gadget, Intel, 2014.
[91] Vgl. Kansal, A., Zhao, F., Liu, J., Kothari, N., Bhattacharya, A.A., „Virtual Machine Power Metering and Provisioning“, in Association for Computing Machinery (Hg.), SoCC '10: Proceedings of the 1st ACM symposium on Cloud computing, ACM Press, New York, NY 2010, S. 39–50.

entweder die gleiche Softwarelösung genutzt wurde oder sichergestellt werden kann, dass die genutzten Berechnungsmethoden sich ähneln und sich die Fehlertoleranzen innerhalb eines akzeptablen Bereichs befinden.

Eine letzte mögliche Herangehensweise liegt in einer Zusammenführung der beiden gerade vorgestellten Varianten. Diese hybriden Methoden versuchen die Unkompliziertheit der softwarebasierten Ansätze mit der Messgenauigkeit von Hardware zu vereinen.[92] Jedoch ist davon auszugehen, dass dadurch auch jegliche negativen Aspekte der entsprechenden Herangehensweisen übernommen werden. Einerseits wird so spezialisierte Hardware benötigt, die Kosten für Anschaffung und Einbau verursacht. Andererseits verlangt die Auswertung der dabei erhaltenen Resultate eine Software, deren Energiebedarf Messungenauigkeiten erhöht und die zusätzlich auf genutzte Hardware abgestimmt werden muss. Aufgrund dessen wird ein solcher Ansatz nur für spezielle Anwendungsfälle als sinnvoll angesehen, wobei es dennoch notwendig ist, diese Methode von Experten entwickeln und evaluieren zu lassen. Somit ist ein Einsatz dieser für die breite Masse der Softwareprodukte und Computersysteme nicht praktikabel.[93]

3.4 System Under Test als Messaufbau

Der Versuchsaufbau ist einer der entscheidendsten Faktoren bei Messungen zur Energieeffizienz. Der am häufigsten verwendete Ansatz ist dabei der SUT. So wird dieser Ansatz unter anderem in Untersuchungen des Umweltbundesamtes empfohlen[94] und ist für die Beantragung des Blauen Engels für Softwareprodukte verpflichtend durchzuführen. Gröger definiert ein SUT als „Hardwaresystem, dessen verbrauchte Energie und verwendete Hardwarekapazitäten gemessen werden“.[95] Dabei werden auch statische Stromverbräuche, wie beispielsweise durch das Betriebssystem, betrachtet. Messungen dieser Art erfolgen häufig während eines Standardnutzungsszenarios. Ein typischer Versuchsaufbau umfasst dabei neben des SUT ein Leistungsmessgerät, ein System zur Datenerfassung und Auswertung (DEA), sowie einen Lasttreiber (s. Abbildung 9). Außerdem wird durch die DEA ein Energieeffizienzbericht generiert, der alle während des Versuchs generierten Daten enthält. Dieser Versuchsaufbau ist allerdings nicht als zwingende Anordnung zu

[92] Vgl. Acar, 2017.
[93] Vgl. ebd.
[94] Vgl. Gröger et al., 2018.
[95] Ebd.

betrachten. Vielmehr kann sie je nach zu untersuchendem System oder vorhandener Technik noch angepasst werden, ohne an Funktionalität zu verlieren.

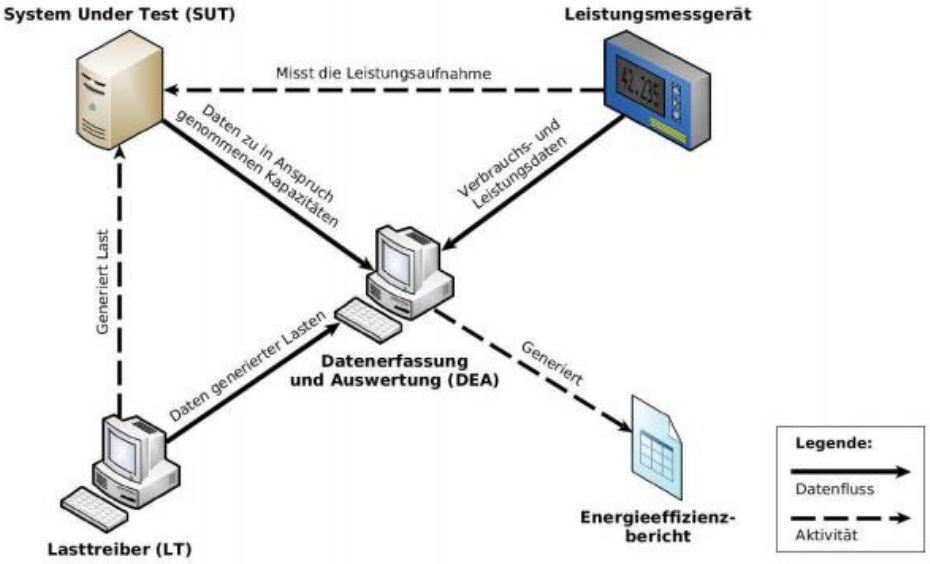

Abbildung 9: Beispielhafter Versuchsaufbau mit einem SUT

Quelle: Gröger et al., 2018

Der SUT enthält dabei alle zu untersuchenden Kapazitäten des Systems. Dazu zählt die zu testende Anwendung, ebenso wie der dafür benötigte Compiler, die Laufzeitumgebung, das Betriebssystem und all jene Hardware, die für den Betrieb dieser Software notwendig ist.[96] Falls erforderlich kann dies auch Standardsoftware zum Erfassen von Ressourcennutzung, wie beispielsweise der prozentualen Auslastung von Prozessor oder Arbeitsspeicher, enthalten. Jegliche anderen Anwendungen und Prozesse, die nicht zwingend für den Betrieb der untersuchten Software notwendig sind, sollten beendet werden. Die mithilfe dieser Tools gesammelten Daten werden kontinuierlich an das System für die Datenerfassung und Auswertung weitergeleitet. Hauptaufgabe des SUT bleibt es jedoch die untersuchte Software zu betreiben.

Um diesen Messaufbau in möglichst vielen Anwendungsbereiche verwenden zu können, werden keine genaueren Angaben zum Aufbau des SUT gemacht. Für eine Webanwendung wäre es somit auch möglich, dass dieser nicht nur aus einem System besteht, sondern aus separaten SUTs für Server und Client. Daher müssen, im Sinne der Reproduzierbarkeit und Fairness der erhaltenen Daten, Informationen über die dabei genutzte Hardware und ausgeführte Software angegeben werden.

[96] Vgl. Gröger et al., 2018.

Die durch diesen SUT verbrauchte Energie wird über das Leistungsmessgerät erfasst. Die Realisierung kann hierbei über alle in Kapitel 4 vorgestellten Methoden erfolgen. Unabhängig davon, ist es für den Erhalt relevanter Daten unabdingbar, dass die ausgewählte Herangehensweise einige Bedingungen erfüllt. So sollten die erhaltenen Werte in regelmäßigen Abständen an die DEA gesendet werden, um Veränderung der Verbräuche zeitnah feststellen zu können. Aus diesem Grund ist es notwendig, dass dieses Messgerät über eine Schnittstelle zum Auslesen der Daten verfügt. Um dabei nach Möglichkeit alle innerhalb dieses Intervalls auftretenden Schwankungen des Energieverbrauchs miteinzubeziehen, sollten die Mittelwerte über den Abständen gebildet und versendet werden.[97] Zudem kann der Stromverbrauch des SUT sich je nach Anwendung schnell ändern, weshalb hierbei gilt, je kleiner das Intervall, umso aussagekräftiger sind die erhaltenen Daten. Für eine genaue Messung sollte dieses Intervall jedoch nicht größer als eine Sekunde werden, da dies andernfalls Ergebnisse verfälschen könnte.[98] Auch die Art des SUT beziehungsweise der Applikation sollte bei der Auswahl und der Einstellung dieses Messgerätes Beachtung finden. So kann für das Beispiel der Webanwendung mit mehreren SUTs, je nach Gerät und Konfiguration, sowohl deren aggregierte Leistungsaufnahme als auch getrennte Verbräuche gemessen werden. Um die Verifizierbarkeit der gemessenen Werte zu gewährleisten, muss stets die Messunsicherheit der verwendeten Methode angegeben werden. Um generell sicherzustellen, dass erhaltene Werte miteinander vergleichbar sind, sollte diese allerdings nicht über einem Prozent liegen.[99] Falls eine Messmethode gewählt wurde, die anderweitige Software auf dem SUT benötigt, muss hierbei auch der Einfluss dieser auf den gemessenen Gesamtverbrauch betrachtet werden.

Der Lasttreiber (LT) ist in einem solchen Versuchsaufbau für die Generierung der Arbeitslast zuständig. Dabei besteht die Notwendigkeit praxisnahe Arbeitsabläufe auszuwählen, um den Erhalt realitätsnaher Resultate sicherstellen zu können. Dementsprechend ist die generierte Last abhängig von der untersuchten Anwendung. Dies geschieht normalerweise durch Nutzung von Automatisierungssoftware oder speziellen Benchmark-Programmen.[100] Eine Umsetzung dieser Form wird benötigt, um die Anforderung der Reproduzierbarkeit einzuhalten. Einerseits wird so sichergestellt,

[97] Vgl. v. Kistowski et al., 2018.
[98] Vgl. ebd.
[99] Vgl. ebd.
[100] Vgl. Gröger et al., 2018.

dass stets die gleiche Last generiert wird und andererseits werden damit menschliche Unsicherheitsfaktoren, wie unterschiedlicher Abstand zwischen Klicks innerhalb einer Anwendung, ausgeschlossen. In einem Szenario einer Webanwendung kann der Lasttreiber zusätzlich Aktionen eines oder mehrerer gleichzeitig agierender Clients simulieren, um für die Praxis relevante Bedingungen zu schaffen. Dies kann ebenso das Simulieren verschiedener Nutzergruppen mit unterschiedlichen Abläufen innerhalb des Programms umfassen.[101] Sollte es aufgrund der Architektur einer Anwendung nicht möglich sein diese von einem anderen System aus zu belasten oder werden reale Klicks in der Nutzeroberfläche benötigt, kann dieser Lasttreiber auch auf dem SUT installiert und betrieben werden. Dabei gilt es zu beachten, dass die Nutzung eines solchen wiederum selbst zum Stromverbrauch beiträgt und gegebenenfalls Messwerte verfälschen kann. Weiterhin muss bedacht werden, dass Programme dieser Art sich nicht nur untereinander in ihrer Energieeffizienz unterscheiden, sondern diese sogar von der Komplexität der Nutzeroberfläche abhängen kann. So kann bereits eine komplexere Nutzeroberfläche schon zu einem erhöhten Energieverbrauch der Automatisierungssoftware führen.[102]

Alle dabei gesammelten Daten werden an die Datenerfassung und Auswertung (DEA) gesendet. Diese Werte umfassen alle Informationen über den Stromverbrauch, mögliche Daten über die Ressourcenauslastung des SUT, sowie die vom LT generierte Last.[103] Durch Auswertung dieser kann die DEA schlussendlich einen Bericht über die Energieeffizienz des Systems erstellen. Der Aufbau eines solchen Berichts unterliegt jedoch keinen festen Vorgaben, sondern ist vielmehr vom Aufbau des SUT und der untersuchten Software abhängig. So kann dieser einen direkten Vergleich zweier Softwareprodukte enthalten oder auch nur graphische Darstellungen der gesammelten Daten eines Systems. Lediglich die, für einen späteren Vergleich zu dieser Untersuchung, zwangsläufig benötigten Informationen sollten enthalten sein. Dies umfasst Spezifikationen des SUT, der untersuchten Anwendung und der generierten Last. Ebenso sollten alle für die Durchführung der Messung genutzten Anwendungen aufgezeigt werden. Details über das System der DEA sind dabei weder für den Versuchsaufbau noch für den Effizienzbericht relevant.

[101] Vgl. Gröger et al., 2018.
[102] Vgl. Dick et al., 2011.
[103] Vgl. ebd.

3.5 Ablauf der Messungen

Für eine möglichst gute Qualität der aus diesem Versuchsaufbau erhaltenen Werte sollten bei der Durchführung v. Kistowskis Charakteristiken Beachtung finden. Insbesondere aus den Punkten der Reproduzierbarkeit und Verwendbarkeit leitet sich die Notwendigkeit ab, den Ablauf von Messungen zu strukturieren. Eine solche Struktur wurde bereits in Untersuchungen des Umweltbundesamtes entwickelt und sieht eine grundlegende Einteilung in drei Phasen vor.[104]

In der ersten Phase wird der Stromverbrauch des Systems im Ruhezustand gemessen. Dieser Schritt wird benötigt, um herauszufinden wie groß der statische Stromverbrauch des genutzten SUTs ist. Mithilfe dieser Grundauslastung wird es später möglich den dynamischen Stromverbrauch aus den Messwerten zu ermitteln, indem die Grundauslastung vom Energiebedarf unter Last subtrahiert wird. Hierbei wird das SUT im eingeschalteten Zustand ohne jegliche Art von Anwendungen betrachtet. Auch die untersuchte Software ist hierbei zwar installiert, jedoch nicht gestartet. Eine Ausnahme bilden dabei sowohl das Betriebssystem als auch möglicherweise genutzte Automatisierungssoftware. Es ist empfehlenswert die dafür notwendigen Installationen mittels einer Standardkonfiguration für das entsprechende Szenario vorzunehmen. Dabei handelt es sich im Optimalfall um ein, an zentraler Stelle gespeichertes, Festplattenimage aller nötiger Software sowie dem Betriebssystem und notwendiger Hardware. [105] Dadurch kann sichergestellt werden, dass bei jeder dieser Messungen der Grundauslastung eine identische Ausgangslage vorliegt.
In dieser Phase der Messung wird versucht einen konstanten Stromverbrauch zu ermitteln. Dafür sollte der Verbrauch so lange gemessen werden, bis sich ein nahezu konstanter Wert eingestellt hat. Zusätzlich sollte dieser Vorgang mehrmals wiederholt werden, um die Messwerte zu validieren. Gröger et. al. empfehlen hierfür, zehn Messungen von jeweils etwa zehn Minuten durchzuführen und den Durchschnittswert dieser als Grundauslastung anzunehmen.[106]

Im darauffolgenden Schritt wird der Einfluss der Software untersucht. Dabei wird zunächst eine Leerlaufmessung durchgeführt.[107] Ziel dieser Phase ist es, herauszufinden wie viel Strom durch das System, inklusive der entsprechenden

[104] Vgl. Gröger et al., 2018.
[105] Vgl. ebd.
[106] Vgl. Dick et al., 2011.
[107] Vgl. ebd.

Anwendung, benötigt wird (E_L) und wie stark der Energieverbrauch des Systems, allein durch den Betrieb dieser im Leerlauf, ansteigt. Einerseits können erhaltene Werte in der letzten Phase genutzt werden, um den puren dynamischen Stromverbrauch der Anwendung unter Last zu ermitteln. Andererseits können, durch Subtraktion der Grundauslastung (E_G), Informationen über die Energieeffizienz der Applikation im Leerlauf (E_{AL}) gewonnen werden.

$$E_{AL} = E_L - E_G$$

Formel 3: Stromverbrauch der Anwendung im Leerlauf
Quelle: In Anlehnung an Gröger et al., 2018

Erkenntnisse dieser Art sind vor allem relevant, da Anwendungen häufig nicht beendet werden, nachdem sie genutzt wurden. Vielmehr werden diese oftmals weiterhin im Hintergrund ausgeführt, wodurch der Energiebedarf eines Rechners ansteigt, selbst wenn der Nutzer nicht direkt mit der Applikation interagiert. Für den Fall einer Webapplikation muss zudem betrachtet werden, dass Server dauerhaft betrieben werden müssen und deshalb, gerade in Zeiten mit niedriger Auslastung, große Einsparungen bei diesen möglich sind.

Der Ablauf von Phase zwei besteht darin, dass die Applikation lediglich gestartet wird, aber noch keine simulierten Interaktionen von Nutzern stattfinden. Weiterhin ist es möglich, durch diese Phase den Einfluss anderer Applikationen auf den Stromverbrauch zu ermitteln, die für eine Nutzung der Software zwingend erforderlich sind.[108] Ein Anwendungsfall hierfür wäre zum Beispiel ein Browser für die Nutzung eines Content-Management-Systems in Form einer Webanwendung. Auch in dieser Phase sollte erneut ein konstanter Stromverbrauch erhalten werden, weshalb mit weiteren Interaktionen erst nach dem Erfassen eines solchen begonnen werden kann. Ähnlich zur ersten Phase sollte der dabei erhaltene Wert durch mehrfache Messungen verifiziert werden. [109]

Als letzte Phase erfolgt die tatsächliche Messung des Energiebedarfs während der Nutzung des Systems. Bei einem Aufbau, wie im vorherigen Kapitel gezeigt, sollte dabei ein Standardnutzungsszenario durch einen Lasttreiber ausgeführt werden. Die

[108] Vgl. Gröger et al., 2018.
[109] Vgl. ebd.

exakte Durchführung orientiert sich hierbei wiederum an der ersten Phase.[110] Die dabei durch das Leistungsmessgerät erhaltenen Daten enthalten den gesamten Stromverbrauch, während einer typischen Nutzung des Systems. Aus diesem Grund werden die dabei erhaltenen Daten auch als Brutto-Auslastung (E_B) bezeichnet. Für eine weiterführende Auswertung dieser, kann nun die Grundauslastung subtrahiert werden, wodurch der gesamte durch die Anwendung verursachte Energiebedarf (E_A) ermittelt werden kann.

$$E_A = E_B - E_G$$

Formel 4: dynamischer Stromverbrauch durch gesamte Anwendung
Quelle: In Anlehnung an Gröger et al., 2018

Ein ähnlicher Ansatz kann durch Subtraktion der Leerlaufauslastung von der Brutto-Auslastung ermittelt werden. In diesem Fall liefert das Ergebnis dieser Berechnung Auskunft über Unterschiede im Verbrauch der Software, je nachdem ob sie aktiv genutzt wird oder sich im Leerlauf befindet (E_{AN}).

$$E_{AN} = E_B - E_L$$

Formel 5: Mehrverbrauch der Anwendung unter Last
Quelle: In Anlehnung an Gröger et al., 2018

Durch eine solche Messdurchführung kann sichergestellt werden, dass unabhängig vom überprüften System, möglichst exakte Daten erhalten werden können. Diese sind zudem sowohl reproduzierbar als auch verständlich, selbst wenn kein breites Fachwissen über Messgeräte oder Energieeffizienz existiert. Ein weiterer Vorteil dieser Vorgehensweise liegt darin, dass dieses unabhängig von der gewählten Messmethodik und von der Struktur des SUTs durchgeführt werden kann. Für den Fall eines getrennten Aufbaus von Client und Server im SUT müsste so lediglich in jeder Phase der gewünschte Wert der jeweiligen Komponente separat bestimmt werden. Am dargelegten Ablauf würde sich selbst dabei allerdings nichts verändern.[111]

[110] Vgl. Gröger et al., 2018.
[111] Vgl. ebd.

4. Methodik

In diesem Kapitel wird die genaue Vorgehensweise erläutert, die genutzt wurde um die Forschungsfragen zu beantworten. Hierbei wird zunächst die Implementierung der entwickelten Webanwendung beschrieben. Mit Hilfe dieser kann die Energieeffizienz einer Webanwendung mit unterschiedlichen Lastverteilungen bestimmt werden. Dabei wird neben den ausgewählten Frameworks auch auf die ausgewählte Last eingegangen. Zusätzlich werden sämtliche Spezifikationen des genutzten Versuchsaufbaus angegeben. Ebenso wird beschrieben, wie dieser Aufbau zur Vermeidung von Messungenauigkeiten vorbereitet wurde und welche sonstigen Aufwände zur Qualitätssicherung der Ergebnisse unternommen wurden. Weiterhin werden Informationen zum Ablauf der Messungen gegeben und es wird darauf eingegangen, wie aus gemessenen Werte die Gesamtverbräuche berechnet und verglichen werden können.

4.1 Allgemein

Der allgemeine Aufbau des in dieser Arbeit durchgeführten Experiments orientiert sich an den im vorherigen Kapitel dargelegten Grundlagen zur Messung von Stromverbräuchen von Software. Für die Beantwortung der Forschungsfrage, einem Vergleich der Energieeffizienz bei unterschiedlicher Lastverteilung, genügt es eine Blackbox Messung durchzuführen, da keine genaue Ursache im Quellcode festgelegt werden muss. Der genutzte Messaufbau folgt dabei dem beschriebenem SUT, wobei kein externer Lasttreiber benötigt wird, da die Arbeitsaufwände durch die Implementierung selbst erzeugt werden. Als Messgerät wurde, aufgrund der Möglichkeit genauere Resultate zu erhalten, ein externes Hardwaremessgerät verwendet. Um eine Beantwortung der Forschungsfrage zu ermöglichen, mussten kleinere Änderungen des Messablaufs und der Berechnungen vorgenommen werden.

4.2 Arbeitslast

Die Auswahl der Arbeitslast, die auf dem untersuchten System ausgeführt wird, ist für die Bewertung der späteren Resultate von großer Bedeutung.[112] Hierbei gilt, dass ausgeführte Berechnungen auch im realen Einsatz des jeweiligen Softwareprodukts häufig Anwendung finden sollten. Um Unterschiede des Energiebedarfs bei unterschiedlicher Lastverteilung von Webanwendungen zu untersuchen, gilt also, dass

[112] Siehe Kapitel 3.2.2.

ausgewählte Arbeitsaufwände typischerweise sowohl auf dem Client als auch auf dem Server ausgeführt werden können. Ebenso sollten Vorgänge anhand der von ihnen beanspruchten Hardwarekomponenten ausgewählt werden.[113] Um größtmögliche Unterschiede feststellen zu können, sollte die größte Quelle des dynamischen Energiebedarfs, der Prozessor, am stärksten belastet werden. Für eine Webanwendung ist es zudem ratsam eine möglichst realitätsnahe Zahl gleichzeitig agierender Clients zu simulieren. All diese Anforderungen verfolgen das Ziel, erhaltene Messwerte relevant und validierbar zu gestalten. Ein Beispiel für solche Aufgaben findet sich in der Verarbeitung von Listen. Diese können sowohl clientseitig als auch serverseitig sortiert, gefiltert oder durchsucht werden. Ein weiterer Vorteil liegt dabei darin, dass diese Aufwände mit einer beliebigen Anzahl an Elementen innerhalb der Liste durchführbar sind und für verschiedenste Datentypen angewendet werden können. Weitere Flexibilität bieten diese zudem im Hinblick auf die exakte Implementierungsweise, die wiederum unterschiedliche Hardwarekapazitäten auslasten können. Da der Umfang dieser Arbeit lediglich den Einfluss der Lastverteilung umfasst und nicht davon ausgegangen werden kann, dass dabei gefundene Tendenzen sich durch die Auswahl verschiedener Algorithmen verändern, umfasst die Implementierung lediglich Sortierverfahren. Als genaues Verfahren wurde Quicksort gewählt, weil es sich hierbei um einen der meist genutzten Sortieralgorithmen handelt und somit ein praxisnahes Beispiel darstellt.

Für die Auswahl der Daten innerhalb der Listen, muss beachtet werden, dass diese reproduzierbar sein müssen. Besonders für Such- und Sortieralgorithmen sollte deshalb ein besonderes Augenmerk darauf liegen, dass erhaltene Resultate nicht durch zufällige Einflüsse, wie eine bereits zufällig korrekt sortierte Liste, verfälscht werden. Da teilweise große Unterschiede der Komplexität zwischen dem besten und dem schlechtesten möglichen Fall existieren, könnte dies sowohl die Korrektheit der erhaltenen Werte als auch deren Reproduzierbarkeit gefährden.

Um diese Zufallsfaktoren bestmöglich auszuschließen, werden Messungen beider Konfigurationen stets mit der gleichen Ausgangslage ausgeführt. Zusätzlich wird dieses Vorgehen für beide Berechnungswege mit identischen Listen ausgeführt und, wie in Kapitel 3.5 beschrieben, für zehn unterschiedliche Listen wiederholt. Durch dieses Vorgehen ist es möglich, die Mittelwerte der entsprechenden Messreihen

[113] Siehe Kapitel 2.3.

verschiedener Konfigurationen zu vergleichen. Dieser Vergleich kann aufgrund der eindeutigen Nutzungsweise stets pro sortierter Liste stattfinden. Nach der in Kapitel 3.2.1 aufgezeigten Definition der Energieeffizienz bildet somit jede sortierte Liste die verrichtete nützliche Arbeit. Um Unterschiede im statischen Verbrauch durch getrennte Anwendungen für die jeweiligen Konfigurationen zu vermeiden, wurde lediglich eine Anwendung entwickelt, die die Möglichkeit bietet, die gewünschte Konfiguration der Lastverteilung über die Nutzeroberfläche einzustellen.
Eine erste Konfiguration besteht darin, dass der Client die unverarbeitete Liste erhält und sämtliche Berechnungen selbst, also im Frontend, ausführt. Dabei entsteht kein zusätzlicher Datenverkehr im Netzwerk. Zudem verschiebt sich der Arbeitsaufwand, sodass der Server weniger ausgelastet wird, wohingegen der Client mehr Aufwände zu tragen hat. Hierbei wird vereinfachend davon ausgegangen, dass der Strombedarf, für die im Frontend ablaufenden Berechnungen, für jeden Nutzer identisch ist. Innerhalb dieses Szenarios gilt es zudem zu beachten, dass für die Durchführung der Sortierung, selbst bei Auswahl einer Liste von realistischer Größe, eine sehr geringe Zeit vergeht. Deshalb ist es auf Grundlage des halbsekündigen Ausleseintervalls des Messgerätes[114] möglich, dass die erwarteten Spitzen im Energiebedarf nicht vom Messgerät erfasst werden. Aus diesem Grund ist es notwendig diese Arbeitsaufwände wiederholt durchzuführen. Dabei ist es wichtig, dass diese Berechnungen nicht direkt hintereinander durchgeführt werden, da sonst unrealistische Auslastungen der Hardware entstehen. Somit wird die Ausführung der wiederholten Berechnungen durch Wartezeiten im JavaScript Code begrenzt. Aus dieser Implementierungsweise entsteht eine konstante elektrische Leistung, mit Hilfe derer die exakte je Liste verbrauchte Energiemenge berechnet werden kann.

Eine zweite Implementierung beinhaltet, dass der Server sämtliche Berechnungen selbst ausführt. Dies bedeutet für den Client, dass keine Aufbereitung der erhaltenen Listen auf seiner Seite notwendig ist und somit kein dynamischer Stromverbrauch auf seiner Seite anfällt. Jedoch wird für jede Sortierung oder Filterung dieser Daten eine Anfrage an den entsprechenden Server notwendig. Dadurch entsteht erhöhter Stromverbrauch durch die Netzwerkübertragung. Zudem werden Kapazitäten des Servers, je nach Anzahl der Nutzer, deutlich stärker ausgelastet. Die Simulation dieser Clients wird hierbei direkt durch das Frontend übernommen, das eine individuell

[114] Siehe Kapitel 4.3.1.

festlegbare Anzahl simultaner Anfragen absendet. Durch diesen Schritt wird untersucht, wie stark sich die Energieeffizienz mit steigender Last verändert. Der hierbei durch den Datenverkehr entstehende Stromverbrauch kann nicht gemessen werden und wird deshalb, entsprechend Kapitel 2.2.4, geschätzt.[115] Die Größe der versendeten Daten wird dabei für einen Nutzer und somit auch eine sortierte Liste gemessen. Im Gegensatz zum ersten Szenario wird die Ausführung der Anfragen nicht begrenzt, weshalb eine hohe Auslastung der Hardware erwartet wird. Hierbei wird erneut eine konstante elektrische Leistung erwartet, da die gleiche Last durch mehrere Anfragen über einen längeren Zeitraum aufrechterhalten wird.

Durch diese beiden Implementierungen wird im Versuchsaufbau kein externer Lasttreiber benötigt, da Arbeitsaufwände von der Anwendung selbst generiert werden. Die Auswahl und der Start der jeweiligen Szenarien erfolgen durch Klicks in der Oberfläche. Dadurch kann sichergestellt werden, dass der Ablauf dem Beispiel einer realen Webapplikation folgt, in der Last durch Aktionen der Nutzer generiert wird. Auf den Einsatz eines Automatisierungstools kann jedoch verzichtet werden. Dies beruht auf der Tatsache, dass Interaktionen zur Einstellung der Konfiguration nicht Teil der Messungen sind und Arbeitsaufwände nach Start des Szenarios automatisiert ausgeführt werden. Dadurch kann ebenfalls ausgeschlossen werden, dass weitere Software für Automatisierung den Energiebedarf des SUT beeinflusst. Weiterhin wird davon ausgegangen, dass die Nutzeroberfläche nicht zum dynamischen Verbrauch des Systems beiträgt, da diese nicht Teil der Untersuchungen ist und sich in den unterschiedlichen Konfigurationen nicht unterscheidet. Die Auswahl der Arbeitsaufwände wird so gewählt, dass die erwartete konstante elektrische Leistung über mehrere Sekunden aufrechterhalten wird, um sicherzustellen, dass Messwerte nicht durch systembedingte Schwankungen beeinflusst werden.

4.3 Spezifikationen

Um während der Untersuchung von Kistowskis Charakteristiken zu befolgen, werden im folgenden Kapitel Spezifikationen des SUT, der verwendeten Frameworks sowie des Messgerätes aufgeführt. Da die gewählten Spezifikationen stets möglichst realistische Szenarien darstellen sollten, wurden alle in den folgenden Ausführungen

[115] Siehe 4.3.3.

genannten Punkte für die Messung eines Content Management Systems in Form einer Webapplikation ausgewählt.

4.3.1 Hardware

Die im Folgenden aufgezeigten Spezifikationen umfassen die gesamte verwendete Hardware. Dazu zählen detaillierte Informationen sowohl des SUT als auch des genutzten Messgerätes. Da die Auswahl des Messgerätes maßgeblichen Einfluss auf das ausgewählte SUT hat, wird dieses im Folgenden zuerst ausgeführt, bevor Details des genutzten Computersystems genannt werden.

Um den Erhalt möglichst exakter Messwerte sicherzustellen, wird zur Messung ein externes Strommessgerät verwendet. Bei diesem handelt es sich um ein GUDE Expert Power Control 1202-1. Welches ausgewählt wurde, da es eine geringe Fehlertoleranz (s. Anhang A.1) bietet. Ebenso enthält es eine Schnittstelle für den Export der gemessenen Werte. Das Auslesen der Daten erfolgt mittels eines Python Tools. Informationen über die aktuelle Stromstärke I, die anliegende Spannung U und auch den aktuellen Leistungsfaktor λ werden damit mittels SNMP zweimal pro Sekunde abgerufen. Anhand dieser Werte wird daraufhin nach Formel 6 die elektrische Leistung ermittelt.

$$P = U * I * \lambda$$

Formel 6: elektrische Leistung bei Wechselstrom
Quelle: In Anlehnung an Aunkofer, 2009

Anhand der elektrischen Leistung und Formel 7 kann nun durch Multiplikation mit der Zeit t, die je Berechnung benötigt wird, die elektrische Arbeit ermittelt werden. Diese gleicht der während dieses Zeitraums verbrauchten Energie.

$$W = P * t$$

Formel 7: elektrische Arbeit
Quelle: LEIFIphysik

Zudem ist keine feste Installation des Gerätes notwendig, wodurch es anschließend auch für weitere Untersuchungen verwendet werden kann. Eine Einschränkung dieses Modells besteht lediglich darin, dass nur Werte eines Gerätes gemessen werden können. Sollten mehrere Geräte gleichzeitig angeschlossen werden, wird entsprechend deren aggregierte Leistungsaufnahme bestimmt. Somit ist keine direkte

Unterscheidung des Verbrauchs von Frontend und Backend möglich. Dieser Fakt kann jedoch durch Anpassungen im Ablauf der Messungen kompensiert werden.[116]

Die Auswahl des verwendeten Computersystems und dessen exakte Hardwarespezifikationen kann bei Messungen zum Stromverbrauch von Software nahezu frei getroffen werden. Es sollte lediglich beachtet werden, dass die Höhe des statischen Stromverbrauchs in direktem Zusammenhang zum ausgewählten System steht. Um auch kleinste Änderungen im dynamischen Energiebedarf sichtbar zu machen, bietet es sich an, ein möglichst kleines System zu nutzen, das aber dennoch repräsentativ für eine praxisnahe Anwendung der Software stehen kann.

Durch die Einschränkungen, die durch die Auswahl des Messgerätes bedingt sind, ergibt sich zudem, dass Verbräuche durch gleichzeitig agierende Clients, sowie der entsprechende Server nicht voneinander getrennt werden können. Aus diesem Grund können sich sowohl Frontend als auch Backend gemeinsam auf einem System befinden. Dadurch reduziert sich der gemessene statische Stromverbrauch massiv, da nur ein Gerät betrieben werden muss. Für die Untersuchungen wird ein handelsüblicher Laptop genutzt. Die Nutzung eines solchen hat im Vergleich zu Desktop-Computern den Vorteil, dass Notebooks in der Regel für mobiles Arbeiten ausgelegt sind und dementsprechend energieeffizienter betrieben werden können. Dadurch enthalten Resultate weniger statischen Verbrauch, wodurch selbst geringe Unterschiede im dynamischen Verbrauch erkennbar werden. Spezifisch handelt es sich um ein Lenovo ThinkPad X240 mit 4 GB Arbeitsspeicher und einem Intel Core i5-4300 Prozessor mit einer Taktfrequenz von 1,9 GHz. Geräte wie dieses werden häufig in Büroumgebungen eingesetzt, weshalb dieses Modell die Bedingung eines praxisnahen Systems erfüllt. Um den exakten aktuellen Verbrauch des Gerätes bestimmen zu können muss sichergestellt werden, dass jegliche benötigte Energie auch durch das Messgerät erfasst wird. Aus diesem Grund wurden beide Akkus des Notebooks entfernt, wodurch es ausschließlich im Netzbetrieb läuft. Da der statische Stromverbrauch eines Rechners stets Schwankungen unterliegen wird, wurden Maßnahmen getroffen, um dieses Hintergrundrauschen zu reduzieren. Als Betriebssystem wurde Debian 11 gewählt, da hierbei der statische Stromverbrauch geringer ist als bei Windows Betriebssystemen, aber Unterschiede bei Verbräuchen dennoch in den gleichen Relationen zu finden sind. Zusätzlich wurde darauf geachtet,

[116] Siehe Kapitel 4.3.3.

keine weiteren Geräte, die Strom verbrauchen, anzuschließen. Dies umfasst etwa die Nutzung einer externen Tastatur. Um eine gleiche Ausgangslage für jede Messung sicherzustellen wurde darauf geachtet, dass der eingebaute Monitor stets lediglich auf der geringsten Helligkeitsstufe betrieben wurde. Somit ist es ausgeschlossen, dass verschiedene Helligkeitseinstellungen des Displays die Resultate der Untersuchungen verfälschen. Gleichzeitig wurde somit die größte Quelle des statischen Stromverbrauchs bereits minimiert.

4.3.2 Frameworks

Um die Relevanz der Ergebnisse für die Praxis zu gewährleisten, muss bedacht werden, dass in Softwareentwicklungsprozessen oftmals Frameworks zum Einsatz kommen. Zwar bieten diese während des Entwicklungsprozesses viele Vorteile, jedoch ist nicht gewährleistet, dass diese auch energieeffizient betrieben werden können. Um die gewonnenen Erkenntnisse über Veränderungen im dynamischen Energiebedarf in Relation zu praxisnahen Verbräuchen setzen zu können, bietet es sich an für die Erstellung der Applikation mehrere Frameworks zu nutzen, die ebenfalls in aktuellen Entwicklungsprozessen häufig Anwendung finden.
Hierbei wird je ein Framework für die Entwicklung der Nutzeroberfläche, sowie eines für die Backendentwicklung genutzt. Deren Nutzung zudem mit JavaScript beziehungsweise C# in, für den jeweiligen Bereich, repräsentativen Programmiersprachen abläuft.

Für das Frontend bietet sich hierbei React an. Dabei handelt es sich um eine JavaScript Bibliothek, die einen hierarchischen Aufbau von Komponenten der Nutzeroberfläche ermöglicht. Diese Komponenten können mittels JavaScript verschiedenste Berechnungen ausführen. Die daraus entstehende Single-Page-Webanwendung bietet ein gutes Beispiel für den Aufbau einer modernen Webapplikation.
Ein typisches Framework für serverseitige Entwicklung findet sich in ASP.NET. Dabei handelt es sich um eine von Microsoft entwickelte Plattform, die speziell für die Entwicklung von Webanwendungen ausgelegt ist. Eine besonders häufig genutzte Funktionalität bietet dabei die Erstellung von APIs. Dieser Teil der Anwendung ist somit für alle serverseitigen Berechnungen, sowie die Beantwortung der generierten Last verantwortlich.

4.4 Messablauf und Berechnungen

Die durchgeführten Messungen folgen in den Grundzügen dem in Kapitel 3.5 beschriebenem Vorgehen. Um die Forschungsfragen mithilfe der implementierten Anwendung beantworten zu können, müssen einige Änderungen der dabei ausgeführten Schritte vorgenommen werden. Ebenso müssen Anpassungen der Berechnungsmethoden stattfinden. Eine Übersicht über alle für die Berechnungen genutzten Werte, deren Abkürzung und Bedeutung findet sich im Anhang A1.

Die Bestimmung des Grundverbrauchs eines Systems bildet auch hierbei den ersten Schritt. Da die Applikation stets auf dem gleichen System läuft, wird dieser einmalig vor Messbeginn ermittelt. Die Bestimmung dieses Werts kann somit zunächst identisch zu dem in Abschnitt 3.5 empfohlenem Vorgehen durchgeführt werden. Da für die Nutzung der Oberfläche einer Webanwendung stets ein Webbrowser benötigt wird, muss ergänzend zum bereits beschriebenem Grundverbrauch noch der Einfluss des Browsers beachtet werden. Für alle Phasen und Messungen innerhalb dieser Ausarbeitung wurde dabei Google Chrome (Version 98.0.4758) genutzt.

Für die zweite Phase der Messungen, dem Verbrauch der Anwendung im Leerlauf, müssen aufgrund der Implementierung einige Anpassungen vorgenommen werden. So ist die alleinige Bestimmung des Leerlaufverbrauchs der gesamten Anwendung, E_{AL}, nicht ausreichend. Vielmehr sollte dieser Wert in den Leerlaufverbrauch des Frontends E_{FL} sowie den des Backends E_{BL},unterteilt werden. Beide Verbräuche werden nach Formel 3 berechnet und enthalten somit keine Verbräuche der Grundauslastung.

Eine weitere Unterteilung ist für die Bestimmung des Stromverbrauchs unter Last notwendig. Diese ist hierbei anhand des gewählten Szenarios zu treffen. Für den Fall clientseitiger Berechnungen muss unabhängig von der gewünschten Nutzeranzahl lediglich eine Messung des Frontends unter der gewünschten Belastung vorgenommen werden. Währenddessen sind keine backendseitigen Prozesse auszuführen, da diese unabhängig vom Nutzer zu bewerten sind. Somit wird nur der durchschnittliche dynamische Verbrauch des Frontends während der Nutzung E_{FN} benötigt. Dieser kann durch Nutzung der Formel 4 erneut ohne den Einfluss der Grundauslastung ermittelt werden. Abschließen kann die dabei bestimmte elektrische Leistung, nach Formel 7, mit der für eine Liste benötigten Zeit t_L, multipliziert werden.

Hieraus ergibt sich die elektrische Arbeit, beziehungsweise der Energieverbrauch, je Liste im ersten Szenario. Mehrverbräuche durch Datenübertragung im Netzwerk fallen währenddessen lediglich durch die Übermittlung des JavaScript Codes an und sind daher vernachlässigbar klein.

$$E_{S1} = t_L * E_{FN}$$

Formel 8: Energieverbrauch in Szenario 1
Quelle: Eigene Formel

Für serverseitige Berechnungen wird eine abweichende Berechnungsmethode benötigt. Zwar wird hierbei erneut die durchschnittlich während des Szenarios benötigte elektrische Leistung berechnet und mit der Dauer pro Liste multipliziert, allerdings beinhaltet dies noch nicht alle anfallenden Verbräuche. So kann durch Anwendung von Formel 8 ausschließlich die elektrische Arbeit des Backends B_N berechnet werden. Jedoch entsteht bei diesem Szenario ein nicht vernachlässigbar kleiner Mehrverbrauch durch zusätzliche Datenübertragung im Internet, E_I. Dieser wird, ähnlich zur Frontendlast, einmalig für die jeweilige Listengröße bestimmt und mit der Anzahl gewünschter Nutzer multipliziert. Da dieser Wert nur geschätzt werden kann, wird im folgenden Kapitel der von Aslan ermittelte Wert verwendet. In Ergänzung dazu wird davon ausgegangen, dass seine 2015 festgestellte Beobachtung einer 2-jährlichen Halbierung des Energiebedarfs weiterhin der realen Entwicklung entspricht. Aus diesem Grund erfolgt eine Schätzung dieses Wertes in den folgenden Untersuchungen mit 0,0075 kWh/GB. Dies entspricht 0,027 Ws/kB.
Somit kann der Stromverbrauch des zweiten Szenarios E_{S2}, mithilfe folgender Formel berechnet werden.

$$E_{S2} = E_{BN} + E_I$$

Formel 9: gesamter Energieverbrauch in Szenario 2
Quelle: Eigene Formel

Durch dieses Vorgehen können mittels der Formeln 8 und 9 die Stromverbräuche beider implementierten Konfigurationen direkt verglichen werden.

5. Ergebnisse

Im Folgenden werden die Resultate der Untersuchung berechnet und aufgezeigt. Der Aufbau des Kapitels folgt dabei zum besseren Verständnis dem bereits beschriebenem Versuchsablauf und den dazugehörigen Phasen.[117]

In der ersten Phase der Untersuchung wurde der Grundbedarf der elektrischen Leistung des Computersystems ermittelt. Ein repräsentativer Ausschnitt aus dieser Messreihe wird in Diagramm 1 dargestellt.

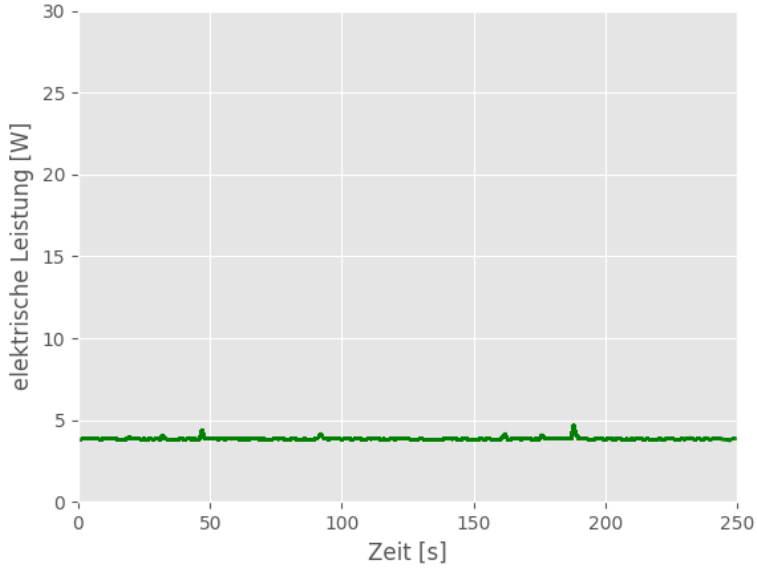

Diagramm 1: Grundbedarf des Systems
Quelle: Eigene Darstellung

Dabei ist erkennbar, dass der statische Energieverbrauch des Systems zwar kein kontinuierlich gleichbleibender Wert ist, allerdings nur geringen Schwankungen unterliegt. Dieses Muster zeichnete sich in allen, in der ersten Phase durchgeführten, Messreihen ab. Somit stellt das hierbei erhaltene Resultat den Grundbedarf des Systems dar und bildet damit die Grundlage für die folgenden Messungen. Der Durchschnitt der Messwerte in dieser Phase beträgt 3,85 W, was einen erwartet niedrigen Wert darstellt. Bemerkenswert ist hierbei, dass die gemessenen Schwankungen auftreten, obwohl das System keinerlei Interaktionen verarbeiten oder Berechnungen durchführen muss. Dies lässt vermuten, dass spätere Resultate ebenfalls durch Ausschläge in der elektrischen Leistung beeinflusst werden können, die nicht vorhersehbar sind oder durch Interaktionen hervorgerufen wurden.

[117] Siehe Kapitel 4.

Ein nächster Schritt bestand in der Bestimmung des Grundbedarfs, wobei zusätzlich ein Webbrowser im Ruhezustand aktiv war. Diagramm 2 stellt dazu einen Teil der dabei erhaltenen Daten dar.

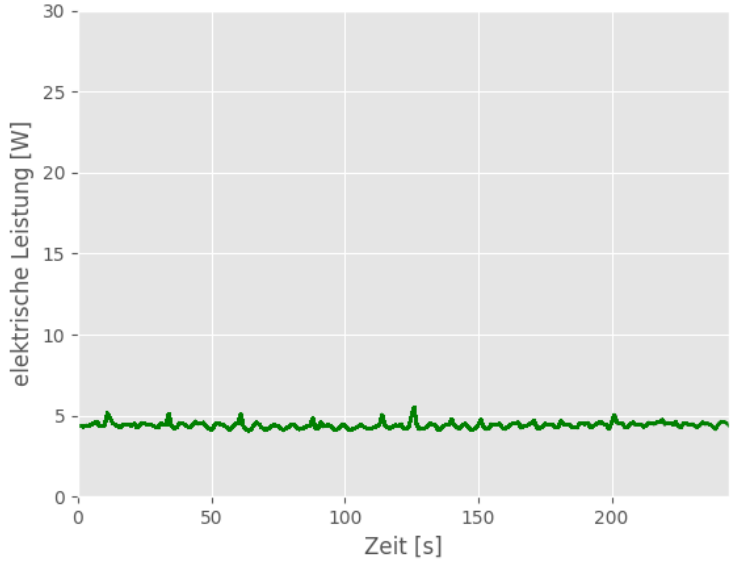

Diagramm 2: Ausschnitt aus Messung des Grundbedarfs mit Webbrowser

Quelle: Eigene Darstellung

Der Mittelwert der aus dieser Messreihe erhaltenen Werte beträgt 4,3 W. Somit zeigt sich, dass ungenutzte Anwendungen nur eine minimale Erhöhung des Grundbedarfs an benötigter elektrischer Leistung, von im Durchschnitt 0,45 W, verursachen. Auffällig ist hierbei, dass nun deutlich größere Ausschläge zu verzeichnen sind. Dies lässt den Schluss zu, dass durch die Nutzung eines Webbrowsers und mit steigender Belastung des Systems Energieverbräuche kontinuierlich inkonstanter werden.

Die versendete Datenmenge für den Aufruf des User Interface wurde zusätzlich ermittelt und beträgt 1,3 kB. Unter Berücksichtigung der getroffenen Annahmen entspricht dies einer elektrischen Leistung von 0,28 Ws je Aufruf der Webapplikation. Zwar stellt dies keinen vernachlässigbar kleinen Wert dar, jedoch zeigte sich, dass diese Datenmenge durch einen minimal geringeren Umfang des JavaScript-Codes ebenfalls nur minimal verringert wird. Somit ist es naheliegend davon auszugehen, dass für den Aufruf der Webanwendung stets ein konstanter Verbrauch unabhängig vom ausgewählten Szenario entsteht, weshalb dieser im weiteren Verlauf der Arbeit vernachlässigt werden kann.

Als folgender Schritt erfolgt das Starten der Anwendung. Hierbei wird zunächst die Leerlaufauslastung bestimmt. Diese wird separat für das React Frontend und für das ASP.NET Backend betrachtet, um die Grundlage für die späteren, bereits in Kapitel 4.4 beschriebenen, Berechnungen zu schaffen. Diagramm 3 und 4 zeigen somit Ausschnitte der Messungen zur Leerlaufauslastung des Frontends, respektive des Backends.

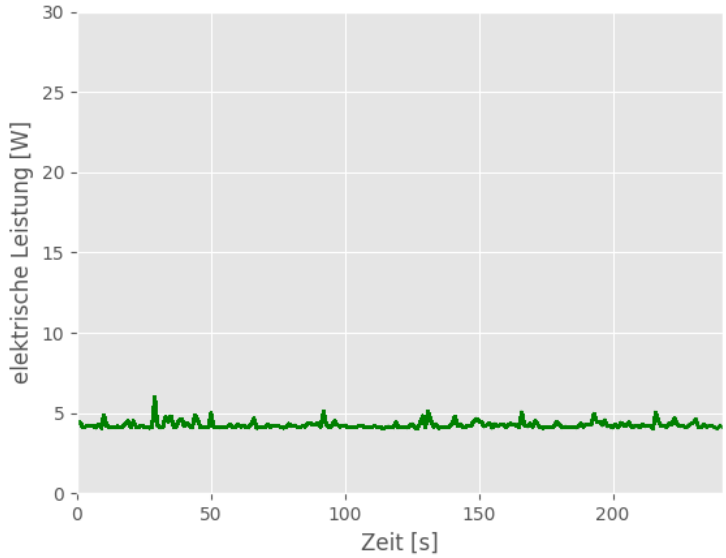

Diagramm 3: Ausschnitt des Leerlaufverbrauch des Frontends

Quelle: Eigene Darstellung

Der Leerlaufverbrauch des Frontends folgt den bisherigen Entwicklungen nicht. Es kann erkannt werden, dass ein auf React basierendes Frontend im Leerlauf den durchschnittlichen Bedarf an elektrischer Leistung nicht messbar beeinflusst. Somit kann davon ausgegangen werden, dass Stromverbräuche, während des Betriebs einer solchen Anwendung, lediglich durch die Nutzung eines Webbrowsers beeinflusst werden. Durch Anwendung von Formel 3 ergibt sich somit, dass das Frontend mitsamt Webbrowser im Ruhezustand ebenfalls einen Mehrverbrauch von 0,45 W im Vergleich zur Grundauslastung des Laptops verursacht. Jedoch kann festgestellt werden, dass die auftretenden Schwankungen sich erneut vergrößern. Diese sichtbaren Ausschläge der elektrischen Leistung treten zwar in einem geringen Rahmen auf, allerdings deutlich häufiger und in unregelmäßigeren Abständen.

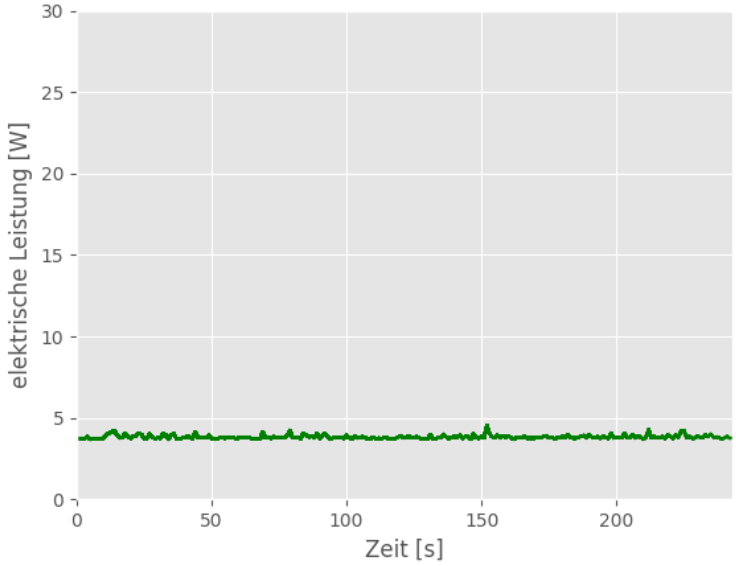

Diagramm 4: Ausschnitt des Leerlaufverbrauchs des Backends
Quelle: Eigene Darstellung

Ähnliches lässt sich bei der Untersuchung des Leerlaufverbrauchs der ASP.NET Anwendung feststellen. Da für diese Messreihe kein Webbrowser benötigt wird und dessen Einfluss somit wegfällt, lassen sich keine messbaren Veränderungen zum Grundbedarf des Systems feststellen. Im Folgenden wird deshalb davon ausgegangen, dass das Backend dieser Applikation keine Auswirkungen auf den statischen Verbrauch des Gesamtsystems hat und somit in späteren Berechnungen nicht weiter beachtet werden muss.
Mit Hilfe dieser Leerlaufverbräuche wird es in der folgenden Phase möglich, die verschiedenen Ansätze anhand der dadurch entstehenden Mehrverbräuche zu vergleichen, selbst wenn diese über einen unterschiedlichen Zeitraum untersucht werden.

Die letzte Phase der Untersuchung umfasst nun die Resultate beider im vorherigen Kapitel beschriebenen Szenarien. Hierfür werden in den Diagrammen 5 und 6 zunächst die exakt gemessenen Werte aufgezeigt. Daraufhin kann die Energieeffizienz beider Vorgehensweisen nach den bekannten Formeln berechnet und gegenübergestellt werden. Als ausreichender Arbeitsaufwand für beide Szenarien wurde eine Liste mit 10 000 Zufallszahlen festgelegt.
Für die Durchführung des ersten Szenarios konnte festgestellt werden, dass die Bearbeitungszeit für eine Liste im Durchschnitt 0,011 Sekunden beträgt. Um möglichst aussagekräftige Daten sammeln zu können wurden diese Berechnungen jeweils 1000-mal wiederholt. Diagramm 4 stellt einen Ausschnitt einer solchen Messreihe dar.

In diesem sind fünf Wiederholungen dieses Aufwandes sichtbar. Zwischen den einzelnen Aufrufen wurde beachtet, dass das System sich erneut dem bereits ermittelten Leerlaufverbrauch mit Browser annähert.

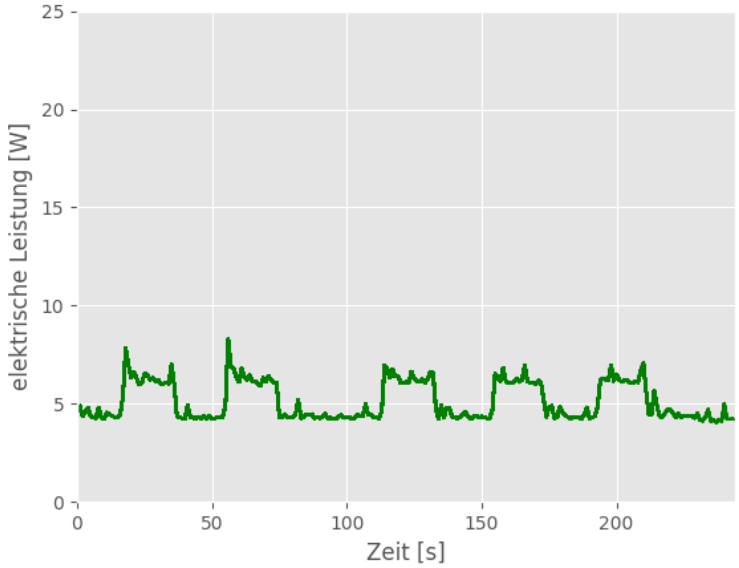

Diagramm 5: Ausschnitt aus Szenario 1

Quelle: Eigene Darstellung

Bei diesem Ausschnitt ist erkennbar, dass durch die Implementierungsweise der Anwendung ein eindeutiger Anstieg im Energiebedarf über einen längeren Zeitraum erzielt werden konnte. Jedoch werden diese Verbräuche weiterhin durch größere Schwankungen, besonders zu Beginn der Berechnungen, beeinflusst. Trotz dieser Ausschläge ist allerdings die Tendenz eines nahezu gleichbleibenden Energiebedarfs deutlich erkennbar. Dies lässt sich auch durch mehrere Versuchsdurchläufe, auch mit unterschiedlichen Listen gleicher Länge, bestätigen. Aus all diesen Messungen konnte eine durchschnittlich benötigte elektrische Leistung von 6,22 W ermittelt werden. Für die Sortierung nur einer Liste wird somit eine Brutto-Auslastung in Form von elektrischer Arbeit von 0,068 Ws benötigt. Zu beachten gilt hierbei, dass dies noch den statischen Stromverbrauch des SUT miteinbezieht. Diese Leerlaufauslastung wurde bereits in der vorherigen Phase bestimmt und beträgt für einen Zeitraum von 0,011 s 0,047 Ws. Da das Backend in diesem Szenario keinen messbaren Mehrverbrauch im Ruhezustand verursacht ergibt sich nun abzüglich des Energiebedarfs im Leerlauf nach Formel 6 ein Mehrverbrauch von 0,021 Ws. Dies bildet somit die benötigte elektrische Leistung für die Sortierung einer Liste im Frontend. Für die Definition der Energieeffizienz nach Johann (s. Formel 1) und einer erbrachten nützlichen Leistung

in Form einer sortierten Liste von 10 000 Elementen gleicht die Energieeffizienz in Szenario 1 somit diesem Resultat.

Die im zweiten Szenario durchgeführten Berechnungen unterscheiden sich in wesentlichen Punkten von den gerade Beschriebenen. So wurden je Anfrage an das ASP.NET Backend 1500 Durchläufe der Berechnungen simuliert. Dies gilt zur Simulation mehrerer Clients, ohne separate Anfragen durch das Frontend zu benötigen, da dies gemessene Stromverbräuche beeinflussen könnte. Ein Ausschnitt der Resultate aus einer solchen Messung wird in Diagramm 6 dargestellt. Dabei wurden zehn simultane Anfragen abgeschickt und somit insgesamt 15 000 Listen sortiert. Dieser Ausschnitt zeigt, wie im vorhergehenden Szenario, fünf Wiederholungen des Aufwandes. Um eine Vergleichbarkeit der Resultate zwischen diesem und dem ersten Szenario zu gewährleisten, wurden identische Listen verwendet.

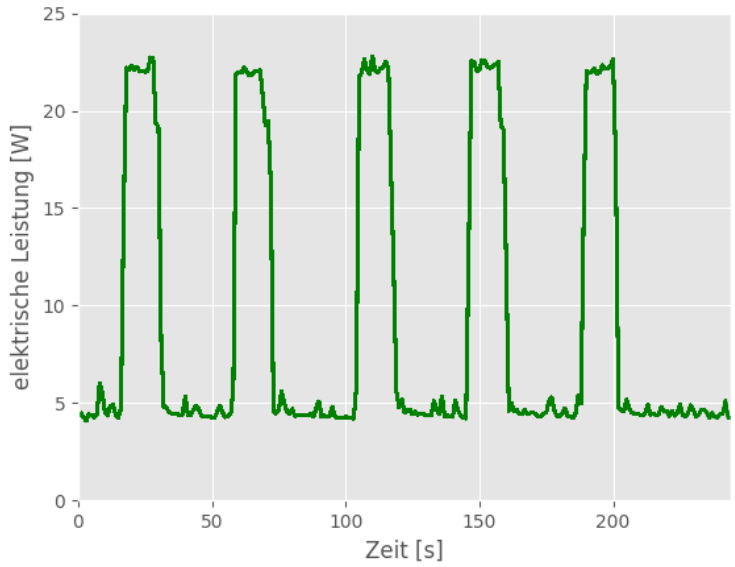

Diagramm 6: Ausschnitt aus Szenario 2
Quelle: Eigene Darstellung

Erwartungsgemäß konnte in jedem Durchlauf ein starker Anstieg des Energiebedarfs festgestellt werden. Es ist aufgrund der hohen Last naheliegend, dass diese Werte bereits dem maximalen Stromverbrauch des Systems entsprechen. Die durchschnittliche erbrachte elektrische Leistung lag während dieser Anwendungsfälle bei 21,28 W und somit deutlich über den im ersten Szenario bestimmten Verbräuchen. Ebenso wurde ermittelt, dass je Durchlauf durchschnittlich 14,2 s benötigt wurden. Für eine Liste entspricht dies einem Zeitraum von lediglich 0,00095 s und somit deutlich weniger Zeit als in der vorherigen Konfiguration. Somit wurde für die Sortierung einer

Liste in Szenario 2 insgesamt eine Brutto-Auslastung von 0,02 Ws verbraucht. Die Abzüge des Verbrauchs im Leerlauf erfolgt nach den gleichen Maßstäben wie im ersten Szenario. Daraus folgt, dass für den Zeitraum von 0,00095 s ein Leerlaufverbrauch von 0,004 Ws entsteht. Somit konnte unter dieser Lastverteilung ein Mehrverbrauch von 0,016 Ws. Zu beachten gilt hierbei, dass dies lediglich den Energiebedarf für die Berechnungen umfasst. Verbräuche durch die Datenübertragung werden deshalb im Folgenden noch geschätzt.
Die Größe einer vom Server an den Client übermittelten Liste beträgt für den aktuellen Anwendungsfall stets 47,9 kB. Nach den getroffenen Annahmen fällst somit je Anfrage an den Server elektrische Arbeit in Höhe von 1,29 Ws an. Durch Addition dieses Wertes mit dem berechneten Mehrverbrauch pro sortierter Liste ergibt sich ein Gesamtverbrauch von 1,306 Ws für Szenario 2. Da dieser Wert je Liste von 10 000 Elementen gilt, bildet er zugleich die Energieeffizienz des zweiten Szenarios.

Eine Gegenüberstellung der gesammelten Daten beider Szenarien wird in Tabelle 1 dargestellt. Dabei beziehen sich alle aufgezeigten Resultate auf die errechneten Werte für jeweils eine sortierte Liste.

Messgrößen	**Szenario 1**	**Szenario 2**
Leerlaufauslastung [W]	4,9	4,9
Durchschnittliche elektrische Leistung unter Last [W]	6,22	21,28
Zeit [s]	0,011	0,001
Brutto-Auslastung [Ws]	0,068	0,021
Mehrverbrauch (ohne Netzwerk) [Ws]	0,021	0,016
Versendete Datenmenge [kB]	0	47,9
Verbrauch im Netzwerk [Ws]	0	1,29
Gesamter Mehrverbrauch [Ws]	0,021	1,306

Tabelle 1: Gegenüberstellung der Resultate für jeweils eine Liste

Quelle: Eigene Tabelle

Bei dem Vergleich der Energieeffizienz beider Herangehensweisen wird deutlich, dass der Energieverbrauch pro verarbeiteter Liste im zweiten Szenario den des ersten Szenarios deutlich übersteigt. Es konnte festgestellt werden, dass für den gegebenen Anwendungsfall der gesamte Mehrverbrauch für die zweite Konfiguration um über 62-mal höher ausfällt als der seines Pendants. Bemerkenswert ist hierbei, dass diese

Unterschiede gravierend durch Verbräuche in der Datenübertragung beeinflusst werden. Die gerade dargelegten Unterschiede resultieren nahezu ausschließlich aus diesem Faktor. So konnte ausgewertet werden, dass die Berechnungen ohne den Einfluss des Netzwerks auf einem stark ausgelasteten System sogar eine bessere Energieeffizienz aufweisen. Pro sortierter Liste konnten hierbei Einsparungen von rund 31 Prozent festgestellt werden. Dies zeigt, dass die in Kapitel 3.2.2 und Abbildung 17 aufgezeigte Veränderung der Energieeffizienz eines Systems je nach Last in der praktischen Anwendung großen Einfluss auf Stromverbräuche eines Systems hat.

6. Diskussion

Nachdem die Resultate der Messungen aufgezeigt wurden, werden diese im folgenden Kapitel auf die zu Beginn definierten Forschungsfragen übertragen. Für ein besseres Verständnis erfolgt diese detaillierte Betrachtung separat für die einzelnen Untersuchungsschwerpunkte. Zudem werden Handlungsempfehlungen für eine energiesparende Lastverteilung innerhalb einer Webapplikation gegeben. Ebenso erfolgt eine kritische Betrachtung der Untersuchung.

Die erste und somit auch grundlegende Forschungsfrage dieser Arbeit lautet „*Wie stark beeinflusst die Lastverteilung die Energieeffizienz einer Webanwendung?*“. Eine Beantwortung dieser Frage kann direkt aus den gesammelten Daten erfolgen. So konnte eindeutig aufgezeigt werden, dass verschiedene Implementierungen, die sich lediglich in ihrer Lastverteilung, nicht aber in den durchgeführten Berechnungen unterscheiden, deutliche Unterschiede in der Energieeffizienz, also der verbrauchten Energiemenge für die gleiche Arbeit, aufweisen. Es wurden Erhöhungen im Verbrauch des gesamten Systems über das 62-fache hinaus festgestellt. Des Weiteren konnten Einsparungen von rund 30% alleinig für die Effizienz der Berechnungen nachgewiesen werden. Dabei zeigte sich, dass diese Einsparungen nicht auf ein spezifisches Szenario begrenzt sind, sondern Bereiche für mögliche Einsparungen in beiden Szenarien vorhanden sind. Besonders deutlich wird dies im direkten Vergleich beider durchgeführter Szenarien. Zwar konnten in Szenario 2 Einsparungen während der Berechnungen erzielt werden, diese wurden dennoch durch den hierbei anfallenden Datenverkehr massiv überkompensiert. Aus diesen Gründen kann eindeutig die Schlussfolgerung getroffen werden, dass die Lastverteilung einen starken Einfluss auf die Energieeffizienz einer Webapplikation ausübt. Dabei gilt zu beachten, dass Unterschiede in der Lastverteilung jeweils Einsparungen in unterschiedlichen Bereichen hervorrufen.

Der nächste Schritt der Untersuchung beschäftigt sich mit der Frage „*Welche Faktoren sind ursächlich für die entstehenden Unterschiede?*“. Durch die Untersuchung konnten zwei Hauptfaktoren festgestellt werden, durch welche die Energieeffizienz von Webanwendungen hauptsächlich beeinflusst wird.

Netzwerkübertragungen bilden den ersten bestimmenden und auch bedeutsamsten Faktor. Es konnte aufgezeigt werden, dass selbst unter der Annahme einer 2-jährlichen Halbierung der dabei entstehenden Verbräuche, es zu einer deutlichen Überkompensation von Einsparungen kommen kann, falls zusätzliche große Datenmengen versendet werden müssen. Dabei gilt zu beachten, dass die errechneten Werte hierbei lediglich auf Schätzungen basieren. Der in den Berechnungen angenommene Verbrauch betrug daher 0,027 Ws/kB. Dies beruht auf der Tatsache, dass die exakten Verbräuche im Netzwerk im realen Einsatz durch diverse Faktoren beeinflusst werden können und somit nicht messbar sind. Nichtsdestotrotz bieten die errechneten Werte einen Anhaltspunkt und stellen Energieverbräuche durch Netzwerkübertragungen als deutlich bestimmenden Faktor für die Energieeffizienz einer Webanwendung dar. Erwähnenswert ist hierbei, dass nur Datenübertragungen berücksichtigt werden sollten, die aufgrund der Lastverteilung anfallen.

Die **Energieverbräuche während der Berechnungen** bieten eine zweite Anlaufstelle für Einsparungen. So konnte mithilfe der gemessenen Werte belegt werden, dass Unterschiede von 30% im Energiebedarf für die Bewältigung der Arbeitsaufwände abhängig vom Berechnungsort entstehen. Hierbei muss allerdings eine weitere Unterteilung stattfinden, da mehrere Punkte existieren, die Einfluss auf diesen Faktor ausüben.

Die **Auslastung des Systems** welches die Berechnungen ausführt hat, nach Grosskops Untersuchungen[118], einen maßgeblichen Einfluss auf die Stromverbräuche, die für Arbeitsaufwände anfallen. Durch die Implementierungsweise der Anwendung konnte dieser Faktor in zufriedenstellendem Maße miteinbezogen werden. So konnte in Szenario 2 durch simultane Nutzeranfragen eine deutlich höhere Auslastung als in Szenario 1 erzielt werden. Dies äußerte sich zunächst in einem merklich erhöhten durchschnittlichen Bedarf an elektrischer Leistung. Dies ist bei einem direkten Vergleich der Diagramme 5 und 6 deutlich erkennbar. Nichtsdestotrotz resultiert aus diesem Anstieg eine deutliche Reduzierung der Verarbeitungszeit, wodurch sich die verrichtete elektrische Arbeit je Arbeitsaufwand reduziert.

In Ergänzung dazu gilt es zudem die **Komplexität der Last**, die bewältigt werden muss, miteinzubeziehen. So ist davon auszugehen, dass die Größe der möglichen

[118] Vgl. Grosskop, 2013.

Verringerungen im Energiebedarf in direktem Zusammenhang zu der Zeit steht, die für den jeweiligen Arbeitsaufwand benötigt wurde. Es ist daher naheliegend, dass je komplexer eine auszuführende Berechnung ist, desto größer sind die zu erwartenden Einsparungen pro verrichteter Arbeit. Mithilfe der entwickelten Anwendung konnten gleiche Berechnungen, wie die im vorherigen Kapitel genutzten, mit einer verkürzten Liste durchgeführt werden. In Zuge dieser Durchführungen war dieser Zusammenhang ebenfalls erkennbar.

Des Weiteren darf der Einfluss der jeweiligen *Programmiersprache* nicht vernachlässigt werden. So konnten Untersuchungen verschiedener Programmiersprachen bereits feststellen, dass teils große Unterschiede zwischen diesen existieren.[119] Aus diesem Grund wurden mit JavaScript im Frontend und C# im Backend übliche Programmiersprachen für den jeweiligen Bereich verwendet, weshalb davon auszugehen ist, dass die ermittelten Werte auch diesen Faktor bereits beinhalten. Nichtsdestotrotz sind auch andere Kombinationen von Programmiersprachen oder Frameworks denkbar. Vor diesem Hintergrund sollten diese stets in Abhängigkeit von den gestellten Anforderungen der Software ausgewählt und überdacht werden.

In welchem Maße alle gerade aufgezählten Faktoren Einfluss auf die Energieeffizienz von Berechnungen ausüben und wie groß deren Auswirkung auf die in dieser Arbeit bestimmten Werte ist, wurde in dieser Ausarbeitung nicht untersucht. Aus diesem Grund kann keine weitere Gewichtung dieser Faktoren erfolgen. Naheliegend ist, dass alle genannten Punkte zu den ermittelten Unterschieden beigetragen haben.

Die letzte Forschungsfrage „*Wie ergibt sich eine optimale Lastverteilung für Webanwendungen in praktischen Anwendungsszenarien?*“ schließt weitestgehend an die ersten beiden an, da hierfür alle relevanten Faktoren miteinbezogen werden müssen. Zusätzlich bestehen für die Beantwortung dieser Fragestellung einige Beschränkungen die im Folgenden benannt werden. So kann nicht jeder Arbeitsaufwand frei austauschbar beim Client oder auf dem Server ausgeführt werden. Dies kann unter anderem durch Sicherheitsbedenken, Datenschutz oder auch geheimen Quellcode begründet sein. Ebenso können besonders rechenintensive Vorgänge oftmals nicht praktikabel clientseitig ausgelagert werden. Eine weitere Beschränkung entsteht durch die Auswahl der Technologie. So beschränken

[119] Vgl. Pereira et al., 2017.

beispielsweise gewählte Frameworks teilweise die Komplexität von Logik, die auf Seiten des Clients ausführbar ist. All diese Gründe werden im Folgenden nicht betrachtet. Stattdessen wird lediglich von Berechnungsaufwänden gesprochen, die sowohl clientseitig als auch serverseitig ausgeführt werden können.

Für den aufgezeigten Fall der Sortierung von Listen kann anhand der erhaltenen Messwerte gefolgert werden, dass Arbeitsaufwände dieser Art clientseitig bewältigt werden sollten, wenn das Ziel einer energiesparenden Webapplikation verfolgt wird. Dies beruht auf der Tatsache, dass die simulierten Arbeitsaufwände gering und schnell zu bewältigen sind, wobei eine größere Datenmenge versendet werden muss. Jedoch handelt es sich hierbei nur um einen von vielen denkbaren Anwendungsfällen und somit lediglich um einen kleinen Ausschnitt aus der Praxis. Nichtsdestotrotz können aus den erlangten Erkenntnissen über die bestimmenden Faktoren und deren jeweiligen Einfluss Rückschlüsse auf praktische Anwendungsfälle gegeben werden. Durch die vielfältigen denkbaren Anwendungsszenarien ist dies nur in Form von Tendenzen möglich.

Wie durch die vorhergehende Forschungsfrage gezeigt werden konnte bildet der Energiebedarf für Netzwerkübertragungen den bestimmenden Faktor. Die hierbei entstehenden Mehrverbräuche sind nur geschätzt, allerdings konnte aufgezeigt werden, dass diese möglichen Einsparungen durch effizientere Berechnungen deutlich überkompensieren. Hieraus kann die Empfehlung abgeleitet werden, dass Berechnungen, für welche große Datenmengen transportiert werden müssen, im Optimalfall stets an der Stelle ausgeführt werden sollten, an der die zu verarbeitenden Daten bereits vorhanden sind. Es gilt zu beachten, dass dabei ausschließlich zusätzlich anfallender Datenverbrauch, also jener, der durch Berechnungen auf Seiten des Clients im Gegensatz zu serverseitigen Ausführungen vermieden werden würden, betrachtet wird. Im Einklang dazu stellte sich Szenario 1, im Vergleich zum zweiten Szenario, als stromsparender heraus.

Für Anwendungsfälle in denen solcher Datenverkehr nicht oder lediglich in geringer Menge anfällt oder Datenübertragung nicht vermeidbar ist, müssen weitere Abwägungen stattfinden. Hierbei ist durch Einbeziehung aller genannten Faktoren davon auszugehen, dass Berechnungen im Backend weniger Energie benötigen als im Frontend. Leider können durch die im weiteren Verlauf dieses Kapitels beschriebenen Einschränkungen keine allgemeingültigen Werte für diese Effizienzsteigerung gegeben werden. Aus diesem Grund kann keine Aussage darüber

getroffen werden ab welcher Komplexität Energieeinsparungen für Berechnungen die Mehrverbräuche im Netzwerk ausgleichen. Dies eröffnet den Bedarf für spezielle Untersuchungen für den jeweiligen Anwendungsfall. Nichtsdestotrotz kann eine allgemeine Tendenz erkannt werden. So konnten durch die durchgeführten Messungen zwar Einsparungen von über 30% gezeigt werden, jedoch stellt dies in absoluten Zahlen lediglich eine eingesparte elektrische Leistung von 0,005 Ws je Liste dar. Diese Reduzierung stellt sich in Relation zu beispielsweise 4,3 Ws, alleinig für den Betrieb des Laptops über eine Sekunde, als minimal heraus. Weiterhin muss beachtet werden, dass diese Einsparungen je Arbeitsaufwand und somit auch je Nutzer der Anwendung anfallen. Des Weiteren können sich, falls alle bisher aufgezeigten Faktoren größtmöglichen Einfluss ausüben, die erzielten Einsparungen in diesem Bereich noch erhöhen. Insbesondere gilt dies für komplexere Berechnungen, da die durchgeführten Sortierungen von Listen mit vergleichsweise geringem Rechenaufwand durchführbar sind.

Zusammenfassend lässt sich festhalten, dass Berechnungen, für welche eine große Datenmenge übermittelt werden muss, ausschließlich besonders energieeffizient ausgeführt werden können, wenn diese Datenübertragung so minimal wie praktisch möglich ausfällt. Wohingegen Lasten mit lediglich geringer Datenmenge aber dennoch hoher Komplexität und hohem Zeitbedarf stets auf dem Server ausgeführt werden sollten. Eine weitere in der Praxis durchführbare Möglichkeit besteht in einer Aufteilung der Last zwischen Client und Server. So könnte in bestimmten Szenarien eine Vorverarbeitung und somit eine Reduzierung der Datenmenge auf Seiten des Clients geschehen. Auf deren Grundlage kann daraufhin die Ausführung komplexerer Berechnungen durch den Server geschehen. Durch dieses Vorgehen würde eine Reduzierung der übermittelten Daten gemeinsam mit einer effizienteren Berechnung erreicht werden. Anhand dieser Erkenntnisse sollte ebenso festgehalten werden, dass eine energiesparende Webanwendung Verbräuche am effektivsten durch eine Minimierung der Datenübertragung reduzieren kann. Da dies nicht nur anhand der Lastverteilung möglich ist, sollte dies ein Kriterium in allen Bereichen der Softwareentwicklung darstellen. Ansatzpunkte hierfür finden sich beispielsweise in

Untersuchungen von Kern, in welchen spezifische Maßnahmen vorgestellt werden.[120] Ähnliche Ansatzpunkte existieren ebenfalls bereits in anderen Veröffentlichungen von Dick.[121, 122]

Da Messungen der Energieeffizienz von Software anfällig für Störungen sind, gibt es Limitationen der durchgeführten Messungen, die aufgeführt werden müssen. Wie bereits erwähnt sind Stromverbräuche stets abhängig vom verwendeten System. Dabei sollte erwähnt werden, dass die Nutzung eines Laptops für backendseitige Berechnungen keine reale Hardware darstellt, da in Serversystemen zumeist spezialisierte Hardware erwartet wird. Aus diesem Grund kann davon ausgegangen werden, dass die Effizienz der Berechnungen in realen Umgebungen Abweichungen aufweist. Weiterhin wurde der Energiebedarf für Netzwerkübertragungen lediglich geschätzt und unterliegt mehreren nicht vorhersehbaren Faktoren. Somit stellt der dargestellte Wert ausschließlich eine Annäherung dar. Des Weiteren sind Einschränkungen durch verwendete Software denkbar. Durch die Nutzung von Debian anstelle eines Windows Betriebssystems könnten Abweichungen entstehen. Zusätzlich dazu besteht die Möglichkeit, dass die genutzten Frameworks Einfluss auf gemessene Unterschiede genommen haben. Allerdings ist dieser Punkt durch den nachgewiesenen vernachlässigbaren Einfluss auf die Grundauslastung des Systems als unwahrscheinlich zu bewerten. Weiterhin sind Mehrverbräuche im Netzwerk durch die Übermittlung von JavaScript Code, der für die Durchführung der Berechnungen benötigt wird, in dieser Untersuchung vernachlässigt wurden. Nichtsdestotrotz besteht die Möglichkeit, dass, insbesondere für große Webanwendungen, auch größere Mengen an Code versendet werden müssen, damit Berechnungen clientseitig ausgeführt werden können. Eine letzte mögliche Fehlerquelle liegt in der Nutzung des Messgerätes. Zwar wurde die aktuelle elektrische Leistung halbsekündlich ermittelt, jedoch können durch dieses Intervall kurzzeitig auftretende Spitzen im Energiebedarf nicht erfasst werden. Die wohl größte Limitation dieser Ausarbeitung besteht darin, dass der Stromverbrauch als einziges Nachhaltigkeitskriterium betrachtet wird.

[120] Vgl. Kern, E., „Nutzerzentriertes Green Web Engineering“, in Horbach, M., Gesellschaft für Informatik e. V., G.f.I.e.V. (Hg.), GI Edition Proceedings 220 - Informatik 2013 - Informatik angepasst an Mensch, Organisation und Umwelt: 16.-20. September 2013 Koblenz, Germany, Köllen, Bonn 2013, S. 966–975.
[121] Vgl. Dick et al., 2010.
[122] Vgl. Dick, M., Kern, E., Johann, T., Naumann, S., Gülden, C., „Green Web Engineering - Measurements and Findings“, in Arndt, H.-K., Knetsch, G., Pillmann, W. (Hg.), Man - environment - Bauhaus. Light up the ideas of environmental Informatics: Proceedings of the 26th International Conference on Informatics - Informatics for Environmental Protection, Sustainable Development and Risk Management, Umweltbundesamt Dessau, Germany, Shaker, Aachen 2012, S. 599–606.

Allerdings umfasst die nachhaltige Softwareentwicklung deutlich mehr Faktoren. Dies wird durch Frameworks zu eben dieser, wie beispielsweise dem Greensoft-Modell oder den Bewertungskriterien des Blauen Engels für Softwareprodukte, verdeutlicht. Zwar spielt die Energieeffizienz hierbei eine große Rolle, sollte in der Praxis aber nicht alleiniger Ansatzpunkt für Verbesserungen sein.

7. Fazit und Ausblick

7.1 Fazit

Die Zielsetzung dieser Arbeit war es, den Einfluss der Lastverteilung innerhalb einer Webanwendung auf die Energieeffizienz dieser zu untersuchen. Dabei lagen Schwerpunkte sowohl auf der Größe dieser Unterschiede als auch auf der Ermittlung von Faktoren, die ursächlich für jene Veränderungen sind. Weiterhin wurde erwartet, dass die gewonnenen Erkenntnisse Rückschlüsse auf praktische Maßnahmen zulassen, durch welche die Entwicklung nachhaltigerer Webanwendungen ermöglicht wird. Für eine Untersuchung dieser Aspekte wurde eine Beispielanwendung entwickelt, die in der Lage ist, gleiche Arbeitsaufwände durch zwei Szenarien sowohl clientseitig als auch serverseitig auszuführen. Unter Nutzung eines SUT konnten somit entstehende Stromverbräuche gemessen und verglichen werden. Anhand dieses Vergleichs konnte festgestellt werden, dass die Verteilung von Arbeitsaufwänden einen deutlichen Einfluss auf die Energieeffizienz ausübt. Dabei stellte sich heraus, dass der Datenverkehr im Netzwerk für das gegebene Beispiel die Hauptursache für entstehende Stromverbräuche ist. Weiterhin konnte belegt werden, dass Berechnungen selbst stets stromsparender verliefen, wenn sie durch den Server verarbeitet wurden. Allerdings wurden mehrere Faktoren ermittelt, die die Größe der Einsparungen an dieser Stelle beeinflussen. Wie groß der Einfluss dieser einzelnen Gesichtspunkte jeweils ist, konnte bisher nicht bestimmt werden. Jedoch zeigten sich diese Einsparungen für das durchgeführte Beispiel nur in sehr geringem Ausmaß, weshalb diese deutlich durch den Einfluss der Netzwerkübertragungen überkompensiert wurden.

Auch wenn es sich bei der untersuchten Anwendung lediglich um einen Einzelfall der in der Praxis möglichen Szenarien handelt, konnten anhand der Resultate dennoch Tendenzen für allgemeine Empfehlungen abgeleitet werden. So zeigte sich, dass für eine energiesparende Webanwendung vorrangig und soweit praktikabel zusätzliche Datenübertragungen zwischen Client und Server vermieden werden sollten. Ebenfalls sind Einsparungen alleinig durch effizientere Berechnungen in deutlichem Maße möglich. Jedoch werden diese deutlich durch mehrere Faktoren bestimmt, deren Einfluss bislang nicht geklärt werden konnte. Daher bleibt für eine Umsetzung der in dieser Ausarbeitung vorgestellten Erkenntnisse eine Evaluierung des speziellen Anwendungsfalls unter Berücksichtigung aller Einflussfaktoren unabdingbar.

7.2 Ausblick

Diese Arbeit umfasst lediglich einen eingeschränkten Bereich des gesamten Themenkomplexes nachhaltiger Software. Aus diesem Grund ergeben sich aus den gewonnenen Erkenntnissen mögliche Themen, die als Grundlage für zukünftige Untersuchungen dienen können. Zunächst bietet sich mit direktem Bezug zu den durchgeführten Messungen eine Evaluation der Resultate auf praxisnahen Systemen innerhalb einer realen Anwendung an. Dabei können die vorgestellten Tendenzen sowohl in ihren Kernaussagen, als auch in den Proportionen in denen Einsparungen möglich sind, in einem realen Szenario überprüft werden. Im Zuge dessen kann zudem erforscht werden, wie groß der Einfluss der in der zweiten Forschungsfrage ermittelten Faktoren ist. Hieraus kann daraufhin eine Gewichtung dieser vorgenommen werden, mit welchen die aufgestellten Tendenzen erweitert werden können. Des Weiteren bietet sich die Möglichkeit genauer zu untersuchen, ob Faktoren oder Indizien bestimmt werden können, ab welchem Zeitpunkt Einsparungen durch Berechnungen einen Mehrverbrauch durch Netzwerkübertragungen überwiegen. Ein weiterer Ansatzpunkt liegt in der Beschränkung, dass die Energieeffizienz losgelöst von anderen Anforderungen an Webanwendungen betrachtet wurde. Besonders hervorzuheben ist hierbei die Nutzerzufriedenheit. So ist denkbar, dass durch stärker ausgelastete Server oder eine veränderte Lastverteilung, die mit häufigeren oder längeren Ladezeiten einhergeht, zudem die User Experience Änderungen erfährt. Aus diesem Grund könnte untersucht werden, wie Nutzer auf eine Verschlechterung der User Experience reagieren, wenn dies mit weniger Stromverbrauch einhergeht und inwiefern Nutzer diese Einschränkungen akzeptieren, wenn ihnen die positiven Effekte dieses Vorgehens offengelegt werden. Ein letzter Themenbereich könnte eine Untersuchung des aktuellen Wissensstandes zu nachhaltiger Softwareentwicklung sein. Grundlage hierfür ist, dass aktuelle Literatur bereits ein breites Wissen bereitstellt, jedoch oftmals wenig praktische Maßnahmen in diesem Bereich unternommen werden. Entwicklungskonzepte zur Erstellung grüner und nachhaltiger Software können im Zuge dessen ebenfalls miteinbezogen werden.

Zusammenfassend bietet diese Arbeit einen guten Einstiegspunkt für die Entwicklung energieeffizienter Webanwendungen. Nichtsdestotrotz handelt es sich um einen großen Themenbereich mit vielen denkbaren Anwendungsfällen mit unterschiedlichen Einflussfaktoren. Aus diesem Grund empfiehlt es sich weitere Forschungen in diesem

Bereich durchzuführen, um zukünftig stromsparendere Anwendungen implementieren beziehungsweise nutzen zu können und somit den steigenden Bedarf an nachhaltigen Systemen gerecht werden zu können.

Literaturverzeichnis

2012 9th IEEE Working Conference on Mining Software Repositories (MSR), IEEE, 2012.

2012 First International Workshop on Green and Sustainable Software (GREENS), IEEE, 2012.

2017 IEEE/ACM 1st International Workshop on Software Engineering Curricula for Millennials (SECM), IEEE, 2017.

2nd International Conference on ICT for Sustainability.

ACAR, H., *Software development methodology in a Green IT environment,* 2017.

ACIS 2008 proceedings, 2008.

Arndt, H.-K., Knetsch, G. , Pillmann, W. (Hg.), *Man - environment - Bauhaus. Light up the ideas of environmental Informatics: Proceedings of the 26th International Conference on Informatics - Informatics for Environmental Protection, Sustainable Development and Risk Management, Umweltbundesamt Dessau, Germany,* Shaker, Aachen 2012.

Aslan, J., Mayers, K., Koomey, J.G., France, C., „Electricity Intensity of Internet Data Transmission: Untangling the Estimates“, in *Journal of Industrial Ecology,* Jg. 22, Nr. 4, 2018, S. 785–798.

Association for Computing Machinery (Hg.), *SoCC '10: Proceedings of the 1st ACM symposium on Cloud computing,* ACM Press, New York, NY 2010.

Aunkofer, B., „Elektrische Leistung bei sinusförmigen Wechselstrom“, https://www.der-wirtschaftsingenieur.de/index.php/elektrische-leistung-bei-sinusformigen-wechselstrom/, Stand: 29.04.2022.

Avritzer, A., Iosup, A., Zhu, X. , Becker, S. (Hg.), *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering,* ACM, New York, NY, USA 2016.

Bundesnetzagentur, „Tätigkeitsbericht Telekommunikation 2016/2017“, 2017.

Bunse, C., Gottschalk, M., Naumann, S. , Winter, A. (Hg.), *2nd Workshop EASED@BUIS 2013 Energy Aware Software-Engineering and Development,* 2013.

Capra, E., Francalanci, C., Slaughter, S.A., „Measuring Application Software Energy Efficiency“, in *IT Professional,* Jg. 14, Nr. 2, 2012, S. 54–61.

Combemale, B., Mernik, M. , Rumpe, B. (Hg.), *Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering,* ACM, New York, NY, USA 2017.

Devanbu, P., Kim, S. , Pinzger, M. (Hg.), *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014,* ACM Press, New York, New York, USA 2014.

Dick, M., Kern, E., Drangmeister, J., Naumann, S., Johann, T., „Measurement and Rating of Software Induced Energy Consumption of Desktop PCs and Servers“, in Hoffman, H., Kotheimer, O., Feilke, S. (Hg.), *Innovations in sharing environmental observations and information,* Shaker, Aachen 2011, S. 290–299.

Dick, M., Kern, E., Johann, T., Naumann, S., Gülden, C., „Green Web Engineering - Measurements and Findings“, in Arndt, H.-K., Knetsch, G., Pillmann, W. (Hg.), *Man - environment - Bauhaus. Light up the ideas of environmental Informatics: Proceedings of the 26th International Conference on Informatics - Informatics for Environmental Protection, Sustainable Development and Risk Management, Umweltbundesamt Dessau, Germany,* Shaker, Aachen 2012, S. 599–606.

Dick, M., Naumann, S., Held, A., „GREEN WEB ENGINEERING - A Set of Principles to Support the Development and Operation of “Green” Websites and their Utilization during a Website’s Life Cycle“, *Proceedings of the 6th International Conference on Web Information Systems and Technology,* SciTePress - Science and and Technology Publications, 2010, S. 48–55.

Falsafi, B. , Vijaykumar, T.N. (Hg.), *Power-aware computer systems: Fourth International Workshop, PACS 2004, Portland, OR, USA, December 5, 2004 revised selected papers,* Springer, Berlin, New York 2005, x, 180.

Götz, S., Ilsche, T., Cardoso, J., Spillner, J., „Software Energy-Efficiency with Sweet Spot Frequencies“, 2014, 10 Seiten.

Gröger, J., Köhler, A., Naumann, S., Filler, A., Guldner, A., Kern, E., Hilty, L., Maksimov, Y., „Entwicklung und Anwendung von Bewertungsgrundlagen für ressourceneffiziente Software unter Berücksichtigung bestehender Methodik - Abschlussbericht“, 2018, 156 Seiten.

Grosskop, K., „PUE for end users - Are you interested in more than bread toasting?“, in Bunse, C., Gottschalk, M., Naumann, S., Winter, A. (Hg.), *2nd Workshop EASED@BUIS 2013 Energy Aware Software-Engineering and Development*, 2013, S. 15–16.

Hilty, L.M., Arnfalk, P., Erdmann, L., Goodman, J., Lehmann, M., Wäger, P.A., „The relevance of information and communication technologies for environmental sustainability – A prospective simulation study“, Jg. 21, Nr. 11, 2006, S. 1618–1629.

Hilty, L., Lohmann, W., Behrendt, S., Evers-Wölk, M., Fichter, K., Hintemann, R., „Grüne Software: Ermittlung und Erschließung von Umweltschutzpotenzialen der Informations- und Kommunikationstechnik (Green IT)“, UBA TEXTE, 2015, 68 Seiten.

Hindle, A., „Green mining: A methodology of relating software change to power consumption“, *2012 9th IEEE Working Conference on Mining Software Repositories (MSR),* IEEE, 2012, S. 78–87.

Hoffman, H., Kotheimer, O. , Feilke, S. (Hg.), *Innovations in sharing environmental observations and information,* Shaker, Aachen 2011, 10 Seiten.

Horbach, M. , Gesellschaft für Informatik e. V., G.f.I.e.V. (Hg.), *GI Edition Proceedings 220 - Informatik 2013 - Informatik angepasst an Mensch, Organisation und Umwelt: 16.-20. September 2013 Koblenz, Germany,* Köllen, Bonn 2013.

Jalote, P., Briand, L. , van der Hoek, A. (Hg.), *Proceedings of the 36th International Conference on Software Engineering,* ACM, New York, NY, USA 2014.

Johann, T., Dick, M., Naumann, S., Kern, E., „How to measure energy-efficiency of software: Metrics and measurement results“, *2012 First International Workshop on Green and Sustainable Software (GREENS),* IEEE, 2012, S. 51–54.

John, L.K., Smith, C.U., Sachs, K. , Lladó, C.M. (Hg.), *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering,* ACM, New York, NY, USA 2015.

John, L.K., Smith, C.U., Sachs, K. , Lladó, C.M. (Hg.), *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering,* ACM, New York, NY, USA 2015.

Kansal, A., Zhao, F., Liu, J., Kothari, N., Bhattacharya, A.A., „Virtual Machine Power Metering and Provisioning“, in Association for Computing Machinery (Hg.), *SoCC '10:*

Proceedings of the 1st ACM symposium on Cloud computing, ACM Press, New York, NY 2010, S. 39–50.

Karyakin, A., Salem, K., „An analysis of memory power consumption in database systems“, *Proceedings of the 13th International Workshop on Data Management on New Hardware,* ACM, New York, NY, USA 2017, S. 1–9.

Kern, E., „Nutzerzentriertes Green Web Engineering“, in Horbach, M., Gesellschaft für Informatik e. V., G.f.I.e.V. (Hg.), *GI Edition Proceedings 220 - Informatik 2013 - Informatik angepasst an Mensch, Organisation und Umwelt: 16.-20. September 2013 Koblenz, Germany,* Köllen, Bonn 2013, S. 966–975.

Kern, E., „Green Computing, Green Software, and Its Characteristics: Awareness, Rating, Challenges“, in Otjacques, B., Hitzelberger, P., Naumann, S., Wohlgemuth, V. (Hg.), *From Science to Society,* Springer International Publishing, Cham 2018, S. 263–273.

Kern, E., Guldner, A., Naumann, S., „Including Software Aspects in Green IT: How to Create Awareness for Green Software Issues“, in Kharchenko, V. (Hg.), *Green IT engineering: Social, business and industrial applications,* Springer, Cham, Switzerland 2019, S. 3–20.

Kharchenko, V. (Hg.), *Green IT engineering: Social, business and industrial applications,* Springer, Cham, Switzerland 2019, 604 Seiten.

LEIFIphysik, „Elektrische Arbeit und Leistung“, https://www.leifiphysik.de/elektrizitaetslehre/elektrische-arbeit-und-leistung/grundwissen/elektrische-arbeit-und-leistung, Stand: 29.04.2022.

Mahesri, A., Vardhan, V., „Power Consumption Breakdown on a Modern Laptop“, in Falsafi, B., Vijaykumar, T.N. (Hg.), *Power-aware computer systems: Fourth International Workshop, PACS 2004, Portland, OR, USA, December 5, 2004 revised selected papers,* Springer, Berlin, New York 2005, S. 165–180.

Mahmoud, S.S., Ahmad, I., „A green model for sustainable software engineering“, in *International Journal of Software Engineering and Its Applications,* Jg. 7, Nr. 4, 2013, S. 55–74.

Manotas, I., Pollock, L., Clause, J., „SEEDS: a software engineer's energy-optimization decision support framework“, in Jalote, P., Briand, L., van der Hoek, A.

(Hg.), *Proceedings of the 36th International Conference on Software Engineering,* ACM, New York, NY, USA 2014, S. 503–514.

McKay, T., Konsor, P.C., *Intel® Power Gadget,* Intel, 2014.

Molla, A., „GITAM: A Model for the Adoption of Green IT“, *ACIS 2008 proceedings*, 2008, S. 64.

Morley, J., Widdicks, K., Hazas, M., „Digitalisation, energy and data demand: The impact of Internet traffic on overall and peak electricity consumption“, in *Energy Research & Social Science,* Jg. 38, 2018, S. 128–137.

Otjacques, B., Hitzelberger, P., Naumann, S. , Wohlgemuth, V. (Hg.), *From Science to Society,* Springer International Publishing, Cham 2018, 322 Seiten.

Pandikumar, S., Kabilan, S.P., „Principles and Holistic Design of Green Web Portal“, in *International Journal of Computer Applications,* Jg. 65, Nr. 9, 2013, S. 23–29.

Pang, C., Hindle, A., Adams, B., Hassan, A.E., „What Do Programmers Know about Software Energy Consumption?“, in *IEEE Software,* Jg. 33, Nr. 3, 2016, S. 83–89.

Park, J.-J., Hong, J.-E., Lee, S.-H., „Investigation for Software Power Consumption of Code Refactoring Techniques“, *SEKE 2014,* S. 717–722.

Paul Dempsey, „Rambus CEO calls for collaboration and an architectural focus for memory“, https://www.techdesignforums.com/blog/2013/11/04/rambus-ceo-collaboration-architecture-memory/, Stand: 13.04.2022.

Pereira, R., Carção, T., Couto, M., Cunha, J., Fernandes, J.P., Saraiva, J., „SPELLing out energy leaks: Aiding developers locate energy inefficient code“, in *Journal of Systems and Software,* Jg. 161, 2020, S. 110463.

Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J.P., Saraiva, J., „Energy efficiency across programming languages: how do energy, time, and memory relate?“, in Combemale, B., Mernik, M., Rumpe, B. (Hg.), *Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering,* ACM, New York, NY, USA 2017, S. 256–267.

Philippot, O., Anglade, A., Leboucq, T., „Characterization of the energy consumption of websites: Impact of website implementation on resource consumption“, *2nd International Conference on ICT for Sustainability,* S. 171–178.

Pinto, G., Castor, F., Liu, Y.D., „Mining questions about software energy consumption“, in Devanbu, P., Kim, S., Pinzger, M. (Hg.), *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014,* ACM Press, New York, New York, USA 2014, S. 22–31.

Proceedings of the 13th International Workshop on Data Management on New Hardware, ACM, New York, NY, USA 2017.

Proceedings of the 6th International Conference on Web Information Systems and Technology, SciTePress - Science and and Technology Publications, 2010.

SEKE 2014.

Singh, J., Naik, K., Mahinthan, V., „Impact of Developer Choices on Energy Consumption of Software on Servers“, in *Procedia Computer Science,* Jg. 62, 2015, S. 385–394.

Stobbe, L., Proske, M., Zedel, H., Hintemann, R., Clausen, J., Beucker, S., „Entwicklung des IKT-bedingten Strombedarfs in Deutschland“, Berlin 2015.

Torre, D., Procaccianti, G., Fucci, D., Lutovac, S., Scanniello, G., „On the Presence of Green and Sustainable Software Engineering in Higher Education Curricula“, *2017 IEEE/ACM 1st International Workshop on Software Engineering Curricula for Millennials (SECM),* IEEE, 2017, S. 54–60.

v. Kistowski, J., Arnold, J.A., Huppler, K., Lange, K.-D., Henning, J.L., Cao, P., „How to Build a Benchmark“, in John, L.K., Smith, C.U., Sachs, K., Lladó, C.M. (Hg.), *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering,* ACM, New York, NY, USA 2015, S. 333–336.

v. Kistowski, J., Block, H., Beckett, J., Lange, K.-D., Arnold, J.A., Kounev, S., „Analysis of the Influences on Server Power Consumption and Energy Efficiency for CPU-Intensive Workloads“, in John, L.K., Smith, C.U., Sachs, K., Lladó, C.M. (Hg.), *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering,* ACM, New York, NY, USA 2015, S. 223–234.

v. Kistowski, J., Block, H., Beckett, J., Spradling, C., Lange, K.-D., Kounev, S., „Variations in CPU Power Consumption“, in Avritzer, A., Iosup, A., Zhu, X., Becker, S. (Hg.), *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering,* ACM, New York, NY, USA 2016, S. 147–158.

v. Kistowski, J., Lange, K.-D., Arnold, J.A., Sharma, S., Pais, J., Block, H., „Measuring and Benchmarking Power Consumption and Energy Efficiency“, in Wolter, K., Knottenbelt, W., van Hoorn, A., Nambiar, M., Koziolek, H. (Hg.), *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering,* ACM, New York, NY, USA 2018, S. 57–65.

Vasques, T.L., Moura, P., Almeida, A. de, „A review on energy efficiency and demand response with focus on small and medium data centers“, in *Energy Efficiency,* Jg. 12, Nr. 5, 2019, S. 1399–1428.

Wolter, K., Knottenbelt, W., van Hoorn, A., Nambiar, M. , Koziolek, H. (Hg.), *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering,* ACM, New York, NY, USA 2018.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als die angegebenen verwendet habe.

Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Magdeburg, 02.05.2022

_____
Florian Kleinert

Anhang

A1: elektrische Messgrößen des Gude Expert Power Control 1202-1

Elektrische Messgrößen				
Messwert	**Bereich**	**Einheit**	**Auflösung**	**Ungenauigkeit (typisch)**
Spannung (voltage)	90-265	V	0,01	< 1%
Strom (current)	0 - 16	A	0,001	< 1,5%
Frequenz (frequency)	45-65	Hz	0,01	< 0,03%
Phasenwinkel (phase)	-180 - +180	°	0,1	< 1%
Wirkleistung (active power)	0 - 4000	W	1	< 1,5%
Blindleistung (reactive power)	0 - 4000	Var	1	< 1,5%
Scheinleistung (apparent power)	0 - 4000	VA	1	< 1,5%
Powerfaktor (PF)	0 - 1	-	0,01	< 3%
Energiezähler				
Wirkenergie (total)	9.999.999,999	kWh	0,001	< 1,5%
Wirkenergie (temp)	9.999.999,999	kWh	0,001	< 1,5%

[123]

A2: allgemeine Messgrößen

Bezeichnung	**Berechnung**	**Bedeutung**
E_G	Messung	Grundverbrauch, Statischer Verbrauch des Systems
E_L	Messung	Leerlaufverbrauch, Gesamtverbrauch mit Anwendung im Leerlauf
E_{AL}	$E_L - E_G$	Verbrauch der gesamten Anwendung im Leerlauf abzüglich des Grundverbrauchs
E_B	Messung	Brutto-Verbrauch, Gesamtverbrauch von System mit Anwendung unter Last. Enthält auch statische Verbräuche
E_A	$E_B - E_G$	Durch die gesamte Anwendung verursachter dynamischer Stromverbrauch

[123] https://www.gude.info/fileadmin/user_upload/products/pdu/1202-Serie/anleitung-epc1202-serie.pdf.

E_{AN}	$E_B - E_L$	Erhöhung des dynamischen Verbrauchs bei Nutzung im Vergleich zum Leerlauf

Tabelle 2: Allgemeine Messgrößen einer üblichen Messung der Energieeffizienz von Software

Quelle: Eigene Tabelle

A3: spezifische Messgrößen

Bezeichnung	**Berechnung**	**Bedeutung**
E_{BL}	$E_L – E_G$	Verbrauch des Backends im Leerlauf. Messung ohne Frontend
E_{FL}	$E_L – E_G$	Verbrauch des Frontends im Leerlauf. Messung ohne Backend
E_{FN}	$E_B – E_G$	Verbrauch des Frontends unter Last. Messung der Brutto-Auslastung erfolgt ohne Backend
E_{BN}	$E_M – E_{FL} – E_G$	Verbrauch des Backends unter Last. Resultat stets abhängig von n. E_M steht hierbei für den gemessenen Wert, da dieser dem Brutto-Verbrauch von Frontend und Backend entspricht.
n	Festlegung	Anzahl simulierter Nutzer
E_{AN}	$E_B - E_L$	Erhöhung des dynamischen Verbrauchs bei Nutzung im Vergleich zum Leerlauf
E_I	Schätzung	Verbrauch im Netzwerk, geschätzt anhand von versandter Datenmenge.
E_{S1}	s. Formel 6	Netto-Verbrauch der Anwendung im ersten Szenario
E_{S2}	s. Formel 8	Netto-Verbrauch der Anwendung im zweiten Szenario

Tabelle 3: Spezifische Größen für die Berechnung der Energieeffizienz der implementierten Anwendung

Quelle: Eigene Tabelle