GREEN-IT: ÖKONOMISCHE UND ÖKOLOGISCHE INTERDEPENDENZEN BEIM BETRIEB VON IT-SYSTEMEN

Christian Kriz
BA Thesis an der Technischen Hochschule Wildau
Betreuer: Prof. Dr. Christian Müller
Fachbereich Wirtschaft Informatik Recht
D-15745 Wildau, Hochschulring 1, Brandenburg
E-mail: christian.kriz@arcor.de

Stichwörter

Green IT, Effizienz, Stromverbrauch, Programmiersprachen, Software, Hardware.

Zusammenfassung

Unter Green-IT[1] versteht man die Verringerung von Umweltbelastungen durch IT-Systeme. Damit verbunden ist häufig die Senkung des Stromverbrauchs durch Steigerung der Energieeffizienz. In der Literatur stößt man oft auf Fallbeispiele und sog. *Best Practices*[2] für die gewerbliche Implementierung von Green-IT. Statt Fallbeispielen finden sich hier mehr technische Aspekte mit Fokus auf Rechner, Softwareentwicklung und Rechenzentren. Abschließend werden auch Smartphones, politische Aspekte sowie die Situation an Hochschulen thematisiert.

1 Vorwort

Mit dem Thema *Green IT* habe ich mich unwissentlich schon vor der Bearbeitung meiner Bachelorarbeit auseinandergesetzt. So ziemlich überall stößt man heutzutage auf das Thema Umweltschutz, nur im IT-Sektor war es mir noch nie über den Weg gelaufen, selbst nicht während meines Studiums der Wirtschaftsinformatik. Die einzigen themenrelevanten Dinge, die mir bis dato aufgefallen sind, wären die Energiemanagementeinstellungen unter Windows, wo man die Leistung seines Rechners herabsetzen kann, um Strom zu sparen, was bei Anwendungen, die kaum etwas tun, aber nahezu komplett den Rechner auslasten, auf die ich immer häufiger stoße, kaum etwas bringt, sowie die *Energy Star*- oder sonstige Aufkleber von Zertifizierungen an Bildschirmen. Daher bin ich davon ausgegangen, dass sich im Bereich der IKT[3] kaum jemand mit „Strom sparen“oder Umweltschutz beschäftigt, was in mir mehr und mehr Bedenken hervorgerufen hat, da Computer und mobile Geräte mittlerweile allgegenwärtig sind und es noch viel mehr sein werden mit dem möglichen Aufkommen des IoT[4], wenn die IKT ein anwachsendes Spektrum von Haushaltsgeräten parasitieren wird. Vielleicht bedingt durch mein vorangegangenes Studium der Physik kam ich zu der Überlegung, dass Computer und verwandte Geräte aus Energieeffizienzsicht überaus mangelhaft abschneiden müssten, da nur ein geringer Bruchteil des Stroms aus der Steckdose tatsächlich genutzt werden würde. Mein Gedankengang dabei war etwa so:

Der Strom treffe beim Rechner als erstes auf das Schaltnetzteil, das aufgrund seines Aufbaus nicht 100% der eingehenden Leistung an die Hardware weitergeben könne, da es etwa 230 Volt Wechselspannung mit etwa 50 Hertz auf Gleichspannungen von zirka 5V und 12V regeln müsse. Die „Hauptstromkonsumenten“bei der Hardware seien CPU[5] und GPU[6], die genau wie das Netzteil so einen großen Teil der zugeführten elektrischen Energie in Wärmeenergie umwandeln, dass sie eine eigene Kühlung brauchen, genau wie oft die sogenannte Northbridge und Southbridge der Hauptplatine. Über der Hardware liegt, vereinfacht gesagt, das Betriebssystem mit seinen Gerätetreibern, welches einen Teil der bereitgestellten Rechenleistung für sich in Anspruch nimmt. Auf dem Betriebssystem läuft die Software, die Programme, die, so meine Vermutung, im seltensten Fall auf Energieeffizienz oder überhaupt effizient programmiert seien. Die Ausgabe am Rechner geschieht schließlich über einen Bildschirm, wiederum mit integriertem Netzteil und Schaltungslogik und der völlig umsonst leuchtet, wenn wir

[1] Informationstechnik
[2] Beste Verfahren
[3] Informations- und Kommunikationstechnologie
[4] Internet of Things (deutsch: Internet der Dinge)
[5] Central Processing Unit (deutsch: Zentraler Prozessor)
[6] Graphics Processing Unit (deutsch: Grafikprozessor)

Nutzer gerade gar nicht hingucken. So sammeln sich am Beispiel Rechner einige Ebenen von der Steckdose bis zur Ausgabe am Bildschirm, auf denen „Energieverlust“stattfindet. Wenn alle aus der Überlegung hervorgegangenen Ebenen (Schaltnetzteil, CPU, GPU, Northbridge, Southbridge, Betriebssystem, Programm, Ausgabegerät; $\sum = 8$) mit 90%iger Energieeffizienz arbeiten würden, was für alle der genannten Teile ein unrealistisch optimistischer Wert sei, läge die Gesamteffizienz η_{Gesamt} bei

$$\eta_{Gesamt} = \eta_{\text{CPU}} * \eta_{\text{GPU}} * \eta_{Northbridge} * \eta_{Southbridge} * \eta_{Betriebssystem} * \eta_{Programm} * \eta_{Bildschirm} = 0,9^8 = 0,4305 \quad (1)$$

43%, was vielleicht für Ingenieure ein akzeptabler Wert sein mag. Macht man aus den 90% für alle Teile einen potenziell realistischeren Wert von 80%, erhält man nach Gleichung 2 nur noch eine erschreckend geringe Gesamteffizienz von knapp 17%. Die Werte sinken dabei mit einer Verringerung der Einzeleffizienzen der Teile oder der Erhöhung der Anzahl in Betracht genommener Teile exponentiell.

$$\eta_{Gesamt} = 0,8^8 = 0,1678 \quad (2)$$

Stimmten diese Werte für die Energieeffizienz einigermaßen mit der Realität überein, hätte das enorme Implikationen für den Betrieb beispielsweise in Firmen, in denen vorrangig an Rechnern gearbeitet wird, in Rechenzentren oder gar dem Internet.

Wie ich spätestens mit der Literaturrecherche für diese Arbeit herausfand, war ich nicht der erste, der sich mit dem Thema auseinandersetzt. Über diese Thematik, die meist als *Green IT*, manchmal *Green ICT* (ICT[7], deutsch IKT) und seltener als *Green Computing* bezeichnet wird, sind schon eine ganze Reihe Bücher und akademischer Arbeiten entstanden. Die Thematik Green IT kann man dabei sicherlich als sehr facettenreich bezeichnen, da sie sich nicht nur mit der Energieeffizienz von Netzteilen oder Hardware beschäftigt, sondern auch interdisziplinär zum Beispiel mit Einsparungen für Unternehmen aus betriebswirtschaftlicher Sicht beim Betrieb von Rechnerclustern, mit dem Nutzungsverhalten von Smartphones bis dahin, wie Programmiersprachen konzipiert sind und wie diese genutzt werden.

Ist es tatsächlich gerechtfertigt, sich über die Energieeffizienz im gesamten IT-Bereich Sorgen zu machen oder wären mögliche Einsparungen vernachlässigbar? Hat die IKT überhaupt einen beträchtlichen Anteil in der Energiewirtschaft eines Betriebs oder einer Nation? Wo spielt im IT-Sektor Energieeffizienz überhaupt eine Rolle, wo hat man sich damit auseinandergesetzt und wo wird Ineffizienz vielleicht auch einfach hingenommen? Diese und weitere Fragestellungen konnte ich im Rahmen dieser Arbeit für mich und den interessierten Leser klären.

Vom Aufbau her werde ich so verfahren, dass zuerst Zusammenhänge am PC[8] betrachtet werden, wobei von einzelnen Bauteilen zum Gesamtsystem aggregiert wird bis hin zur Nutzung des Softwarelebenszyklus. Anschließend wird der Horizont auf Unternehmen, den mobilen Sektor, mögliche Entwicklung des IoT bis hin zur politischen Situation expandiert.

2 Interdependenzen

Die beiden Optionen, wie der Stromverbrauch von Rechnersystemen strategisch gesenkt werden kann, sind:

1. Hohe Energieeffizienz des Systems.
2. Senkung der Systemnutzung.

Auf Punkt 2 soll in dieser Arbeit weniger eingegangen werden. Die Reduzierung der Systemnutzung beschäftigt sich vor allem damit, Nutzer auf Green IT aufmerksam zu machen und Mechanismen zu implementieren, wie die Zeit, die Systeme im angeschalteten Zustand verbringen, reduziert werden kann (auch beispielsweise durch Softwareautomation (vgl. [61])) [36].

Nicht ganz abwegig war mein Ansatz in der Einleitung, den Gesamtstromverbrauch eines Rechners auf seine einzelnen Komponenten verteilt zu untersuchen. Vom Prinzip her lässt sich der Gesamtprozess der IKT-Nutzung ähnlich in Teilprozesse gliedern, bis sich am Ende die Umweltauswirkungen einzelner Elemente analysieren lassen. Es sind häufig bestimmte Teilprozesse, Baugruppen- oder Elemente am Gesamtverbrauch stärker beteiligt als andere. Am Rechner hat das Schaltnetzteil zusammen mit dem Hauptprozessor einen Anteil von über 50% am Gesamtenergieeinsatz (vgl. Abb. 1). Eine Senkung oder Optimierung des Verbrauchs dieser Teile hätte also die größten Auswirkungen auf den Verbrauch des Gesamtsystems.

2.1 Stromversorgung

Bei der Spannungsversorgung von Logikschaltungen gibt es das grundsätzliche Problem, dass diese in der Regel mit Gleichspannungen von 3,3V und 5V, zugehörige aktive Kühler mit 12V arbeiten, aber die Netzspannung eine sinusförmige Wechselspannung von etwa 230V ist. Daher benötigen alle geläufigen Rechnermodelle ein Schaltnetzteil, das die Netzspannung gleich auf drei unterschiedliche, erwähnte Gleichspannungen transformiert.

[7] Information and Communications Technology (deutsch: siehe IKT)

[8] Personal Computer (ortsfester Rechner)

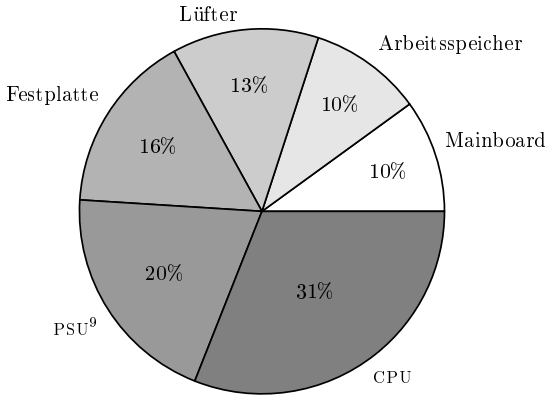


Abbildung 1: Anteile der Bauteile/-gruppen am durchschnittlichen Gesamtverbrauch eines Rechners [39]

Zwar ist der prinzipielle Aufbau eines Schaltnetzteils wie in Abb. 2 dargestellt nicht sonderlich komplex, trotzdem ist der Vorgang, einen Strom, der ständig die Polarität und die Spannung ändert, zu einer möglichst stromspitzenfreien Gleichspannung zu glätten, verlustbehaftet und marktübliche Modelle haben eine typische Energieeffizienz im Bereich 65% bis 75% [40], während am Prozessor, der mit Gleichspannungen im Bereich von 1,5V betrieben wird, tatsächlich nur noch 50% ankommen (wegen weiterer Spannungskonversionen auf der Hauptplatine) [1], obwohl es bereits geschützte Zertifizierungen für den Markt gibt, wovon die kleinste "80 Plus" (von 2005) ist, die ein Gerät erhalten kann, wenn es folgende Kriterien erfüllt:

- Mehr als 80% Leistungseffizienz bei allen Lastzuständen und
- ein Wirkleistungsfaktor von mindestens $0,9$ bei 100% Last.

Rechner, die von einem "80 Plus" zertifizierten Netzteil versorgt werden, können jährlich etwa 85kWh Strom einsparen, Server sogar 300kWh/a im Vergleich zur herkömmlichen Stromversorgung.

Die hauptsächlichen Leistungsverluste bei den herkömmlichen Netzteilen, die elektrotechnisch meist auf Eintaktflusswandler- oder Gegentaktflusswandlertopologie gebaut sind, werden durch Rauschen, Schaltverluste und Belastungen hochfrequenter Schaltvorgänge verursacht [40]. Die genannten Topologien beschreiben, wie prinzipiell die Spulen, die die eigentliche Transformationsarbeit leisten (und selbst nie 100% Wirkungsgrad erreichen können bei der Umwandlung von elektrischer zu elektromagnetischer und wieder zu elektrischer Energie zurück), zueinander geschaltet sind, was signifikanten Einfluss auf die Gesamtschaltung hat. Dort gibt es verschiedene Optimierungsansätze durch Anpassung der Bauteilkennwerte oder der Gesamtschaltung. In den letzten Jahren war eine Effizienzsteigerung durch LLC[10]-Topologie in der Forschung nur begrenzt erfolgreich. Der Wirkungsgradspielraum nach oben hin sei fast ausgeschöpft und Green IT müsse anderswo, beispielsweise softwareseitig ansetzen [60, 35].

Ein neues Problem, mit dem Schaltnetzteile erst seit einigen Jahren zu kämpfen haben, ist, dass der Rechner zwischen Ruhemodus und Volllast innerhalb von Nanosekunden hin- und herschaltet, wobei die Leistungsaufnahmedifferenz in diesem Zeitintervall inzwischen mehrere hundert Watt betragen kann, besonders dann, wenn noch eine Grafikkarte mit hoher Leistungsaufnahme im System verbaut ist. Schaltnetzteile arbeiten dann am effizientesten, wenn die Leistungsaufnahme des Rechners am größten ist [50], was nur Einsparungen bringen kann, wenn der Rechner die meiste Zeit auf Last läuft und möglichst wenig inaktiv ist. Die Effizienzverteilung nach Auslastungsgrad ist in Abb. 3 dargestellt. Insgesamt ist festzuhalten:

1. Netzteile arbeiten umso effizienter, je höher die Leistungsaufnahme ist. Längere Zeiträume der Inaktivität des Systems sind also nicht nur möglicherweise „Stromverschwendung", wenn Rechner gar nicht benutzt werden, sondern auch ungünstig für die Energieeffizienz beim Rechnerbetrieb.
2. Schaltnetzteile können aufgrund ihres elektrotechnischen Funktionsprinzips nie 100% Wirkungsgrad erreichen, gewöhnliche Modelle arbeiten im Bereich von 65% bis 75%.
3. "80 Plus" oder besser zertifizierte Netzteile können Strom einsparen. In einem Rechenbeispiel für die Schweiz ließe sich dort der jährliche Stromverbrauch durch Komplettumstellung auf „80 Plus"-Netzteile um 55GWh reduzieren [1].
4. Da Schaltnetzteile in jedem Rechner vorhanden sind, hätte eine Verbesserung ihrer Energieeffizienz eine flächendeckend positive Auswirkung auf die Energieeffizienz von Rechnern zur Folge. Kontrollgremien bzw. der Gesetzgeber könnte hier ansetzen und Vorschriften für einen Mindestwirkungsgrad einführen.

2.2 Hardware

2.2.1 Prozessoren

Aus Abb. 1 geht hervor, dass der größte Stromkonsument im Rechnersystem der Hauptprozessor mit grob $\frac{1}{3}$ des Gesamtenergiebedarfs ist. Die sehr

[10] Spule-Spule-Kondensator ('L' steht in der Elektrotechnik für Spule, 'C' für Kondensator)

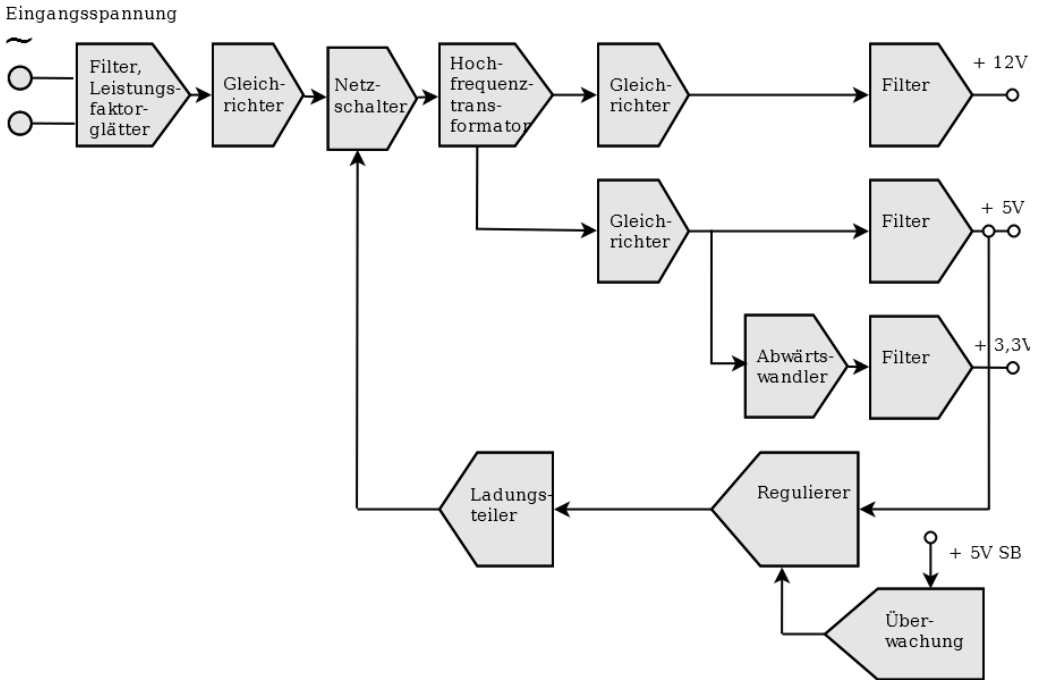


Abbildung 2: Vereinfachte Arbeitsweise eines Schaltnetzteils [1]

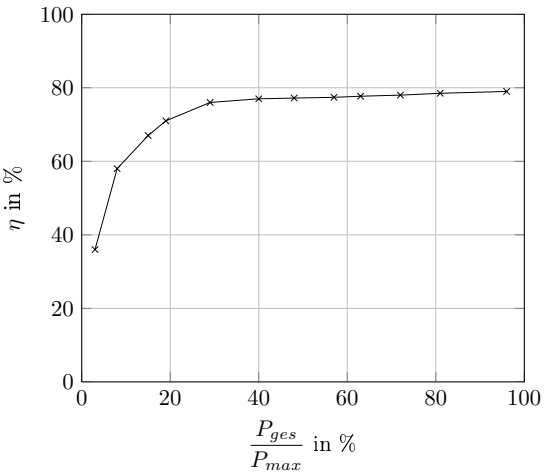


Abbildung 3: Typische Effizienzverteilung (Effizienz η) eines Schaltnetzteils nach Auslastungsgrad (Gesamtleistung (aller Ausgänge) P(ges) / Maximalleistung P(max)) [1]

ähnlich arbeitenden GPUs spielen bei Rechnern außerhalb des Unternehmensbereichs von Servern und Workstations eine ähnlich große Rolle. Eng zusammen hängt mit dem Verbrauch beider Prozessortypen die Kühlung, die etwa $\frac{1}{8}$ ausmacht, denn die meiste Hitze wird von ihnen abgeleitet. Für die Kühlung aller anderen Bauteile reichen in der Regel lüfterlose Passivkühlkörper, die keinen Strom benötigen, aus. Während die Rechenleistung pro Watt Leistungsaufnahme über die letzten Hardwaregenerationen konstant geblieben ist, stieg die Rechenleistung selbst stetig an und der Preis pro Einheit Rechenleistung nahm kontinuierlich ab. Eine Ursache dafür ist, dass Endverbraucher aus dem Bereich der Privathaushalte weniger Interesse an Energieeffizienz haben, die im Haushalts-PC-Bereich wenig umworben wird, als an einem performanten System mit gutem Preis-Leistungsverhältnis, wovon ein Symptom der sogenannte „Megahertz - Krieg“ist: Megahertz wird als Einheit der Taktfrequenz (eigentlich Hz[11]; $1Hz = \frac{1}{1s}$; $1MHz = 1000000Hz$) bei Prozessoren genutzt und lässt sich leichter vergleichen und vermarkten als aussagekräftige, aber komplizierte Benchmarks [10].

Als Alternative zu Benchmarks lässt sich die Rechenleistung von CPUs oder GPUs auch mittels Flops[12] vergleichen. Je nach Anwendungsbereich macht hier das eine oder andere mehr Sinn. Um Endnutzern ein realistischeres Bild vom Leistungsvolumen der Prozessoren zu malen, wäre es wünschenswert, wenn eine der aussagekräftigen Kennzahlen sich im Marketing der Produkte durchsetzen könnte. Benchmarks sind häufig nur in Computerzeitschriften zu finden.

CPUs haben heutzutage meist nicht mehr eine feste Taktfrequenz, mit der sie arbeiten. Häufig wird die Taktfrequenz vom Betriebssystem oder einer Treibersoftware, welche von den großen Chipherstellern angeboten wird, dynamisch beim Betrieb so geregelt,

[11] Hertz, physikalische Einheit der Frequenz

[12] Floating-Point-Operations per Second (deutsch: Gleitkommaoperationen pro Sekunde)

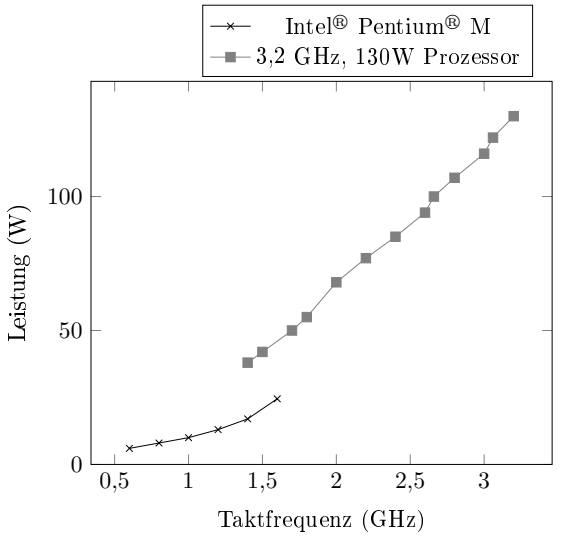


Abbildung 4: Leistungsaufnahme in Abhängigkeit von der Taktfrequenz exemplarischer CPUs [36]

dass sie im Lastbetrieb ihr Maximum erreicht und im Leerlauf eine geringe Frequenz ähnlich der Leerlaufdrehzahl eines Automotors hat. Der Stromeinsatz von CPUs steht, wie in Abb. 4 zu erkennen ist, nicht in linearer Abhängigkeit zur Taktfrequenz, sondern steigt stärker an, umso höher die Frequenz: Beispiel-CPU 2 läuft bei 1,5GHz mit 42W; bei 3,0GHz mit 116W, was eine scheinbare Leistungssteigerung von 100% bei einem Mehrverbrauch von $\frac{116W}{42W} - 1 \approx 176\%$ bedeutet. „Scheinbar"ist die Leistungssteigerung deshalb, da ein Prozessor mit doppelter Taktfrequenz nicht doppelt so leistungsfähig ist wie vorab besprochen. Der Faktor liegt bei $1,0 < x < 2,0$ im unteren Drittel, wobei er von System zu System stark variiert. Aus wirtschaftlicher Sicht hat man also mit wachsenden Grenzkosten bei der Prozessorleistung zu tun. Von Werk aus sind Prozessoren nie mit der energetisch wirtschaftlichsten Taktfrequenz konfiguriert, sondern mit solcher, die mit dem mitgelieferten Kühlkörper noch möglichst sicher zu managen ist. Dabei ist die Wärmeentwicklung nicht proportional zur Taktfrequenz, sondern ungefähr proportional zur Leistungsaufnahme [36].

Der Einsatz dynamischer Prozessorfrequenz ist also wichtiges Mittel für deren Energieoptimierung und eine Übertaktung, wie sie häufig von Computerspielenthusiasten vorgenommen wird, ist aus Green IT-Sicht nicht zu empfehlen. Das Gegenstück zum Übertakten jedoch - das Undervolting, auch DVFS[13] - hat aus Energieeffizienzsicht Vorteile: Da der Rechenleistungszuwachs von Prozessoren mit einem überproportionalen Energieaufwand verbunden ist (Abb. 4), kann mit absichtlichem Undervolting überproportional (zur Rechenleistung) viel Strom gespart werden [28].

Prozessoren selbst sind nicht eigenständige Systeme, auch wenn deren Bestandteile nicht modular sind wie die eines typischen Rechners. Moderne CPU bestehen hauptsächlich aus einer Kontrolleinheit, FPU[14], ALU[15] und Registern, die nach Nähe zur Kontrolleinheit *Cache* L[16]1 (am nähesten und typischerweise am kleinsten) bis L3 genannt werden, wobei eine CPU heutzutage in aller Regel aus mehreren parallelisierten CPU-Logiken (Kernen) besteht. Etwa $\frac{1}{4}$ des Energiekonsums eines typischen Prozessors fällt im Lastbetrieb auf die Register, etwa $\frac{1}{4}$ auf den ROB[17], der sich um die Reihenfolge der Anweisungsausführung kümmert und wiederum zirka $\frac{1}{4}$ auf die *Instruction Queue*, die Anweisungswarteschlange [20].

Für den Endverbraucher spielen diese Messwerte sicher keine Rolle; für Chipdesigner, besonders für Prozessoren im mobilen Bereich, wo Energienutzung wegen begrenzter Akkukapazität auch ohne Blick auf Green IT eine starke Rolle spielt, dafür sehr.

2.2.2 Arbeitsspeicher

Mit der steigenden Leistungsfähigkeit der Prozessoren und ihrer Parallelisierung nehmen auch die Anforderungen an Größe und Bandbreite der Arbeitsspeicher zu. Die heute geläufigsten Arbeitsspeicher sind DRAM[18]-DIMM[20]s. Prinzipiell auffälligster Stromfresser beim Betrieb von DRAM ist der forcierte Standby-Modus. Arbeitsspeicher kann während des Rechnerbetriebs nie ganz abgeschaltet werden, es muss sich zu jeder Zeit ein Reststrom in den Speicherkondensatoren befinden, da es sich beim Arbeitsspeicher um flüchtigen Speicher handelt, der in kürzester Zeit seine Daten verliert, sobald er nicht mehr mit Strom versorgt wird. Im Standby-Modus verbraucht der Speicher etwa $\frac{1}{3}$ soviel Strom wie während eines Lese- oder Schreibvorgangs [36].

Die Adressierung, bei der vor jedem Lese- und Schreibvorgang erst die Adresse angegeben und verarbeitet werden muss, auf die zugegriffen werden soll, ist energetisch ein hinzunehmendes Übel und lässt sich der Architektur wegen nicht vermeiden. Eine Studie hat jedoch gezeigt, dass es möglich ist, durch Aufteilung der RAM-Chips (ein DRAM DIMM enthält mehrere Speicherchips), genauer der Speicherbänke, in kleinere Einheiten die Energieeffizienz um 44% zu steigern bei einer Leistungseinbuße von 7,4% [68].

[13] Dynamic Voltage Frequency Scaling (deutsch: dynamische Spannung-/Frequenzdimensionierung)

[14] Floating Point Unit (deutsch: Gleitkommaeinheit)

[15] Arithmetic Logic Unit (deutsch: Arithmetisch-logische Einheit)

[16] Level (im Kontext CPU-Cache)

[17] Re-order Buffer (deutsch: Sortierungspuffer)

[18] Dynamic Random Access Memory (deutsch: dynamischer RAM[19])

[20] Dual Inline Memory Module (deutsch: Doppelreihiges Speichermodul)

Eine andere Forschungsarbeit zeigt, dass sich durch eine Architekturänderung, die sich in bestehende Hardware integrieren lässt, wiederum etwa 37% höhere Energieeffizienz erreichbar ist. Die Änderung besteht darin, bei hochgradig parallelisierten Speichertransaktionen automatisch solche zu erkennen und zu unterdrücken, die mit Sicherheit zu einem Transaktionsabbruch - banal ausgedrückt *ins Leere* - führen.

2.2.3 Sonstiges

Die wichtigsten übrigen Verbraucher des Rechners sind neben Prozessor und Arbeitsspeicher Festplatte (gemeint sind HDD[21]; SSD[22] sind viel energieeffizienter und damit weniger bedenklich aus Green IT-Sicht) auch das Peripheriegerät Bildschirm.

	HDD	SSD
A: Leistung bei Last	8W	<1W
B: Leistung bei Leerlauf	5W	$\sim$ 0W
$\frac{B}{A}$	62,5%	$\sim$ 0%

Tabelle 1: Vergleich stromverbrauchsrelevanter Kennzahlen beispielhafter HDDs und Halbleiterspeicher [36]

Festplatten haben im Gegensatz zu SSDs bewegliche Teile und brauchen im Leerlauf 59% bis 71% soviel Strom wie im Zugriffsmodus, was sich ungünstig auf ihre Energieeffizienz auswirkt. Halbleiterspeicher (von denen SSDs eine Teilmenge bilden) benötigen im Vergleich dazu im „Leerlauf“meist sogar fast keine Energie. Bei Lese- oder Schreibvorgängen liegen sie bei unter 1W. Der Stromverbrauch ist bei SSDs stärker modellabhängig als bei Festplatten[36]. Auch bei den Speichermedien ist es trivial, es sei aber trotzdem nochmal erwähnt, dass der Stromverbrauch sich nicht nur mit effizienterer Hardware reduzieren lässt, sondern auch mit geringerer Nutzung, also in diesem Fall mit einer geringeren zu speichernden Menge an Daten.

Bildschirme sind streng genommen nicht Hardware, sondern zählen eigentlich mit Drucker, Scanner, Lautsprecher etc. zu den Peripheriegeräten. Sie haben mit Ausmusterung der CRT[23]-Technologie stark an Energieeffizienz gewonnen. Während die typische Leistungsaufnahme eines CRT-Monitors bei bis zu 150W lag, liegt sie bei modernen Flachbildschirmen mit gleichgroßer Bildschirmdiagonale mit typischerweise unter 50W bei einem Drittel des Werts der CRT-Monitore. Der Stromverbrauch ist

[21] Hard Disk Drive (deutsch: Festplatte)
[22] Solid State Drive (deutsch: Halbleiterlaufwerk)
[23] Cathode Ray Tube (deutsch: Kathodenstrahlröhre, Braun'sche Röhre)

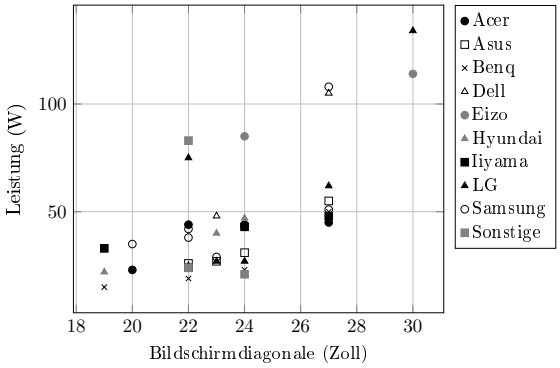


Abbildung 5: 40 Monitore nach Hersteller: Leistung in Abhängigkeit zur Bildschirmdiagonalen [3]

von Modell zu Modell unterschiedlich, steigt aber mit der Größe an (vgl. Abb. 5) [34, 3]. Manche Hersteller scheinen auf Energieeffizienz Wert zu legen: Zwei Modelle eines Fabrikanten haben die niedrigste Leistung und ein Modell derselben Firma die zweitniedrigste, verglichen mit Modellen anderer Hersteller bei gleichbleibender Bildschirmdiagonale. Da hier andere Kenngrößen fehlen bzw. unterhalb der Modelle nicht konstant sind, wie maximale Auflösung, Luminanz und Bildwiederholrate, ist die Studie nur bedingt aussagekräftig.

Würde man die Liste der stromhungrigsten Peripheriegeräte weiterführen, kämen an nächster Stelle nach den Monitoren wohl die Drucker, bei denen besonders die Laserdrucker stromintensiv arbeiten. Tintenstrahldrucker haben eine typische Leistung von zirka 12W, Laserdrucker etwa die achtfache. Hinter den Druckern kämen Lautsprecher und Router, die sich im selben Bereich von der Leistung her bewegen, mit unter 10W [34].

2.3 In der Softwareentwicklung

Was auf den ersten Blick unintuitiv erscheinen mag, für mich war es jedenfalls so, ist, dass Green IT auch überall in der Softwareentwicklung zu finden ist. Um das mit einem simplen Beispiel zu veranschaulichen, werde ich das vorangegangene Thema „Monitore“nochmal aufgreifen:

Der Stromverbrauch von Bildschirmen ist nicht nur abhängig von Bildschirmdiagonale und Modell, sondern auch von der Gesamtluminanz des angezeigten Bilds. Der Stromverbrauch lässt sich also reduzieren mit Verringerung der Helligkeitseinstellung am Bildschirm. Darüber hinaus gibt es auch die Idee, beispielsweise Internetseiten so zu gestalten, dass die Gesamtluminanz des Monitors mit der angezeigten Seite möglichst gering ist. Der Stromverbrauch kann bei unterschiedlichen Bildschirmhelligkeiten um bis zu einem Faktor von ungefähr 8 variieren [36].

Der Bildschirm ist nicht das einzige Gerät, dessen

Stromkonsum sich von Software beeinflussen lässt. Hier bestehen weitere Interdependenzen:

Um die Rechenbelastung des Systems bei Betrieb einer Software so gering bzw. so energieeffizient wie möglich zu halten, lassen sich schon bei der Programmierung eben dieser Optimierungen bei der Wahl der Programmiersprache sowie bei der Verwendung effizienter Algorithmen vornehmen, wie in den folgenden Abschnitten gezeigt wird. Ein weiterer Aspekt ist es, auf Softwareentwicklungsprozessoptimierungsebene den Entwicklungsaufwand zu minimieren. Eine Möglichkeit dafür, in die ich während meines Praktikums und der anschließenden Tätigkeit am Fraunhofer-Institut für Produktionsanlagen und Konstruktionstechnik im Geschäftsfeld Modellbasiertes Entwickeln einen Einblick hatte, nämlich die Wiederverwendung (engl. *reuse*), wird im Kapitel Software Reuse behandelt.

2.3.1 Wahl der Programmiersprache

Evident ist, dass Software, die häufig bzw. lange Zeit auf einer großen Anzahl von Rechnern betrieben wird, eine kritischere Rolle für die Weltenergiewirtschaft spielt als ein Programm, das als Hausarbeit von einem Studenten angefertigt wurde. Wenn es tatsächlich so ist, dass Programmiersprachen Einfluss auf die Energieeffizienz ihrer Software haben, mag man daraus die Frage extrapolieren: Sind die Programme, die eine wichtige Rolle im täglichen Betrieb von IT-Systemen haben, denn in einer effizienten Programmiersprache geschrieben? Studien zeigen unwidersprüchlich, dass Programme, die die gleiche Aufgaben verrichten, nicht die gleiche Effizienz bei der Erledigung vorweisen, wenn sie in unterschiedlichen Programmiersprachen geschrieben sind. Um die Studien miteinander vergleichbar zu machen, wurde das arithmetische Mittel relevanter Messwerte einer Studie als 1 definiert und die Relation dieser Messwerte jeweiliger Programmiersprachen zum Durchschnitt als Vergleich benutzt. In Abb. 6 ist die relative Abweichung der Messergebnisse zu den arithmetischen Mitteln jeweiliger Studien dargestellt. Da in den Studien meist eine Zeit oder ein Verbrauch (beispielsweise *belegter Arbeitsspeicher*) gemessen wurde, zeigen kleinere Werte immer eine höhere relative Effizienz an.

Die Ergebnisse der Studien unterscheiden sich leicht, was wohl an unterschiedlichen verwendeten Versuchsalgorithmen liegt: In der Arbeit von M. Prager wurde mit einer Matrizenmultiplikation gearbeitet, Aruba/Fernández-Villavente verwendeten eine stochastische makroökonomische Wachstumsfunktion nach dem Solow-Modell und die Mitarbeiter der Universität Lille programmierten eine rekursive Lösung für das Spiel *Türme von Hanoi*. Insgesamt lässt sich aber klar eine Rangordnung erkennen, die von C und der davon abgeleiteten, objektorientierten Version C++ angeführt wird. Bei Laufzeit interpretierte Skriptsprachen wie Python und Perl schneiden am schlechtesten (ineffizientesten bei der Problemlösung bezogen auf Rechenzeit/CPU-Ausnutzung) ab. Auch der Arbeitsspeichereinsatz ist bei ihnen sowie auch bei Java auffällig hoch [48, 5, 53].

Warum werden dann weniger effiziente Programmiersprachen wie Java oder Python überhaupt benutzt und warum wurden sie überhaupt entwickelt? Ein wichtiges Maß an Effizienz einer Programmiersprache wurde bislang verschwiegen und auch nur in einer der Studien gemessen, und zwar die Effizienz, mit der Programmierer in einer Sprache ein Problem lösen können. Es zeigt sich, dass in den unterschiedlichen Programmiersprachen die Programmierer

a) unterschiedlich schnell programmieren (gemessen in unkommentierten Quelltextzeilen pro Stunde),

b) Programme vom Quelltextumfang her unterschiedlich umfangreich verfassen und

c) für dasselbe Programm unterschiedlich viel Entwicklungszeit benötigen.

Das Ergebnis war, dass Java, C++ und C im Schnitt etwa doppelt soviel Arbeitsaufwand für Programmierer beim Lösen einer Softwareentwicklungsaufgabe verursachen als stärker abstrahierte Sprachen wie beispielsweise Python oder Perl [54]. Zwischen Programmen verschiedener Programmiersprachen sind zusätzlich Qualitätsunterschiede zu erkennen. So ist C++ im Vergleich mit C einfacher zu warten und weniger fehleranfällig [13].

Man kann also sagen, die Energieeffizienz und die Entwicklungseffizienz verhalten sich antiproportional zueinander. Das liegt ganz einfach daran, dass Sprachen wie C und C++ weniger abstrahiert sind zum Maschinenkode, der Assemblersprache, als Skriptsprachen wie Python und Perl. Java hingegen läuft auf einer virtuellen Maschine, wodurch es sich Sicherheit bei der Laufzeit gegen Effizienz erkauft. Aus Green IT-Sicht ist der derzeitige Trend so weit wie möglich zu abstrahieren, den Entwicklungsaufwand (und damit die Kosten) so gering wie möglich zu halten, kritisch zu sehen. Wenn es um die Entwicklung von Software geht, bei der abzusehen ist, dass sie frequent und auf einer großen Anzahl von Rechnern ausgeführt werden wird, sollte man möglichst abwägen, etwas Mehraufwand in die Optimierung der Energieeffizienz der Software zu investieren.

2.3.2 Verwendung und Effizienz von Algorithmen

Eine weitere Interdependenz auf die Energieeffizienz beim Betrieb von Computern, die eng mit der Wahl der Programmiersprache zusammenhängt, ist, *wie* die Software programmiert wird. Wie das qualitativ möglich ist, wird in diesem Kapitel geklärt.

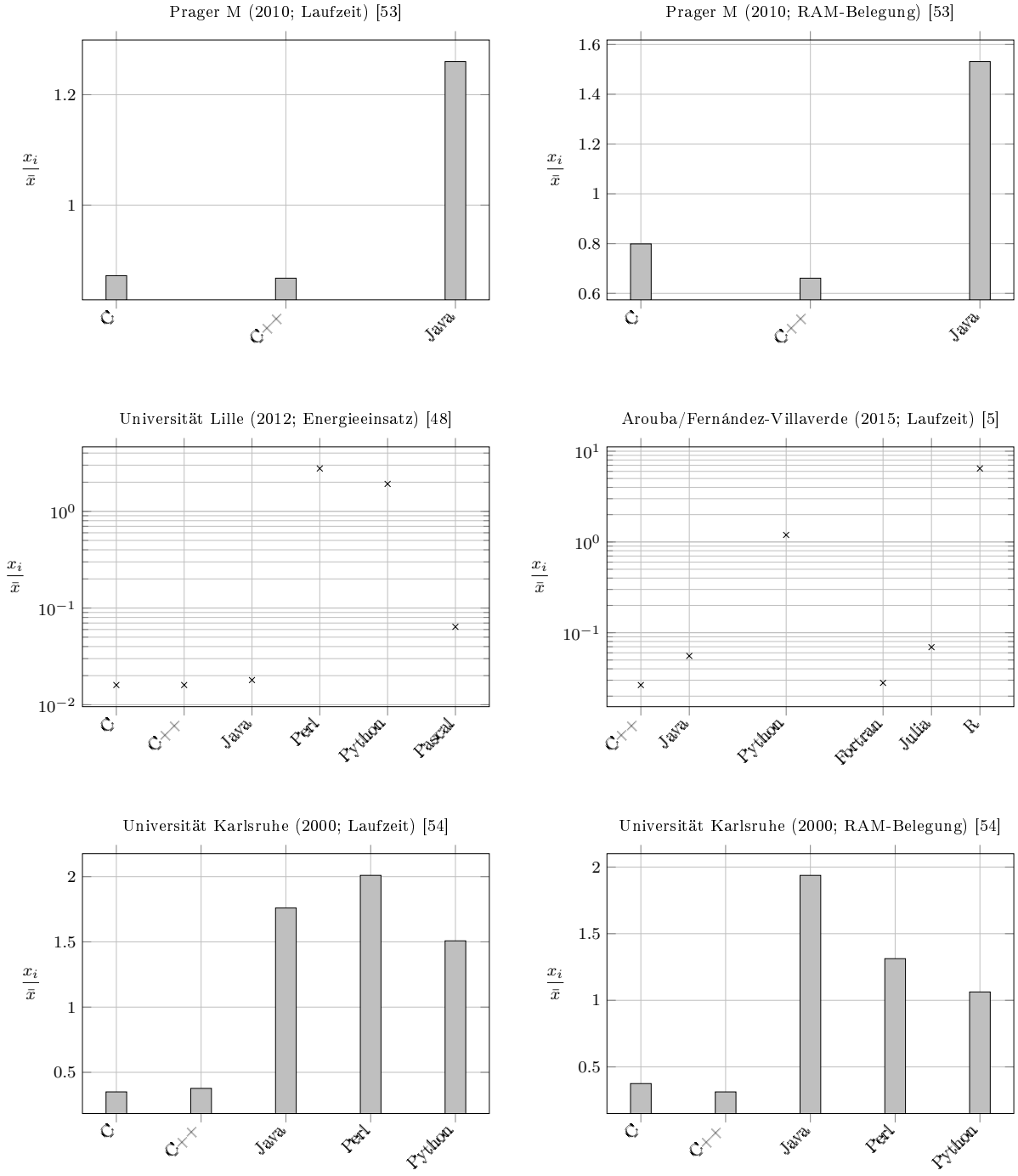


Abbildung 6: Relativer Vergleich der Effizienz von Programmiersprachen verschiedener Studien (Zusammenfassung)

Der Softwareentwickler hat also bei der Erstellung einer Applikation Einfluss darauf, wie hoch der Leistungsbedarf der Computer beim Betrieb dieser ist. In diesem Zusammenhang kommen mehrere unmittelbare Probleme auf:

- Da bei der Ausbildung von Softwareentwicklern der Aspekt der Softwareenergieeffizienz gänzlich fernbleibt, haben Programmierer keine bis kaum Kenntnisse darüber, welchen Einfluss ihre Arbeit auf diese hat. Beste Vorgehensweisen, sog. *Best Practices*, sind unbekannt, auch *wie* Software konkret Einfluss auf den Computerenergiehaushalt hat, ist den allermeisten Programmierern fremd [49].

- Besonders beim agilen Softwareentwicklungsprozess, der heute weit verbreitet ist, wird performantes Implementieren von Algorithmen vernachlässigt [63].
- Vermutlich verantwortlich für den genannten Status quo sind Bildungseinrichtungen bzw. deren Verwaltung, die keinen Wert auf Green IT Unterrichtsinhalte legen [18].
- Wirtschaftliche Interessen können Green IT-Grundsätzen entgegenwirken: In der Anwendungsentwicklung für den mobilen Sektor machen werberelevante Softwarewartungsarbeiten 50% des gesamten Wartungsaufwands aus. Werbung macht außerdem im Schnitt 15% des Akkuverbrauchs aus und sogar 98% des Datenverkehrs [55].

Trotz dieser Hindernisse ist der Softwareenergiekonsum aktives Forschungsfeld und Thema einer wachsenden Anzahl von Publikationen. Die Ursachen überflüssigen Rechenaufwands sind meist auf Programmierfehler bzw. nicht optimierte Programmierung zurückzuführen. Häufigste Optimierungsmöglichkeiten sind beispielsweise Applikationen, die nicht benutzt werden (z.B. minimiert sind), zu deaktivieren und bei der Kommunikation mit Datenbanken optimal geschriebene Anfragen (*Queries*) zu nutzen [55].

Eine weitere, in vielen Quellen genannte Möglichkeit der Optimierung von Programmkode besteht in der Parallelisierung. In diesem Zusammenhang stößt man häufig auf den Begriff *Race to Idle* (deutsch: Wettrennen zum Leerlauf). Damit ist gemeint, dass ein Rechenprozess dann am effizientesten (bzw. performantesten) ist, wenn er am schnellsten abläuft und der Rechner sich wieder im Leerlauf befindet. Die vorangegangene Aussage muss, um vollständig akkurat zu sein, um einen Aspekt erweitert werden: Mathematisch lässt sich der Energieverbrauch einer Prozesskette (*schedule*) $E_p(S)$ wie folgt abbilden [2]:

$$E_p(S) = \sum_{i=1}^{n} v_i \frac{P(s_i)}{s_i} \tag{3}$$

In Worten ist der Energieverbrauch einer Prozesskette folglich die Summe aller Produkte der Prozessvolumen v_i mit dem Quotienten aus der Leistung bei einer bestimmten Prozessorgeschwindigkeit $P(s_i)$ und der Prozessorgeschwindigkeit s_i. Tatsächlich zeigt Irani et al. [30], dass der Term $\frac{P(s_i)}{s_i}$ nicht dann am kleinsten ist (und damit das Produkt und folglich der Energieeinsatz), wenn die Prozessorgeschwindigkeit maximal ist, sondern eine Geschwindigkeit s_{crit} existiert, bei welcher der Term minimal wird.

Bei vielen Algorithmen kann der *Race to Idle*-Aspekt auch optimiert werden, indem man sie soweit wie möglich parallelisiert, d.h. auf *Threads* aufteilt, die gleichzeitig an Teilprozessen arbeiten. Benchmarks zeigen, dass ein auf zwei Threads aufgeteilter Prozess etwas mehr als die Hälfte der Zeit benötigt wie ein nicht aufgeteilter Prozess. Bei vier Threads etwa ein Drittel, bei acht Threads etwa ein Viertel. Der zusätzliche Nutzen nimmt mit jedem zusätzlichen Thread ab, jedoch war die Performanz und Energieeffizienz in dem Benchmark mit der höchsten Anzahl Threads auch am größten [59]. Der Nutzen des Einsatzes von Parallelisierung hängt auch davon ab, wie parallelisiert wird. In der Programmiersprache Java beispielsweise arbeiten von `Executor` verwaltete Threads effizienter als gewöhnliche `Threads`. In Java scheint der *Race to Idle* auch eine geringere Rolle zu spielen [51].

Ein für Programmierer in den meisten Programmiersprachen bekämpfbares Problem ist die Fragmentierung von Daten. Diese führt zu ineffizientem Leseverhalten der Datenträger, besonders Festplatten sind davon betroffen. Vermeiden lässt sich das mit dem Definieren fester Blockgrößen beim Schreiben. In der Sprache C# im *.NET-Framework* geht das zum Beispiel via Befehl `setlength`. Große sequenzielle Datenströme arbeiten auch beim Lesen effektiver, wenn das in großen Blöcken geschieht. In der Quelle werden mindestens 8 Kilobyte für die Blockgröße angegeben. Solche Lese- und Schreibzugriffe sollten zusätzlich nach Möglichkeit in einen gesonderten Thread gelegt werden für größtmögliche Performanz [59].

Dank der Modellierung von Softwareverhalten mittels Programmteilung (*Program Slicing*) und Funktionsteilung (*Feature Slicing*) ist es möglich, die Energieeffizienz separater Programmfunktionen (*Features*) zu bestimmen, wobei es sogar möglich ist, die Leistung einzelner Komponenten (CPU, RAM, HDD) zu messen [14, 31]. So hat beim Design der GUI[24] beispielsweise das Scrollverhalten Einfluss auf die Energieeffizienz. Den größten Energieeinsatz benötigt hier automatisches Scrollen, gefolgt vom Scrollen per Mausrad. Nutzen der *Bild-auf-/Bild-ab*-Tasten zeigt die besten Ergebnisse [43].

Für die Softwareentwicklung im Gebiet der Computerspiele ist es relevant zu wissen, dass im Bereich 3D nicht die Grafik selbst den größten Rechenaufwand verursacht, sondern die Physiksimulation (auch *Physik-Engine*), gefolgt von der Pfadfindung computergesteuerter Akteure und an dritter Stelle erst die eigentliche Grafik. In diesem Bereich der Softwareentwicklung wird Parallelisierung schon intensiv eingesetzt und zeigt eine große Steigerung der Effizienz [6].

Eine häufige Datenstruktur, die in der Programmierung verwendet wird, sind *Container*. Container sind Sammlungen und enthalten Werte beliebiger an-

[24] Graphical User Interface (deutsch: grafische Nutzeroberfläche)

derer Datentypen. Es gibt verschiedene Implementierungen, die sich von Programmiersprache zu Programmiersprache unterscheiden. Zu den grundlegenden zählen `Array`, `HashSet` und `LinkedList` sowie verschiedene Varianten und Kombinationen von ihnen. Die Typen variieren in der Effizienz bei der Ausführung bestimmter Prozesse (z.B. über die gesamte Sammlung iterieren, das Element an Position x aufrufen). Abgesehen davon, dass hier Softwareentwickler oft nicht wissen, welcher Datentyp für ihre Zwecke der am besten geeignete ist, lässt sich ihr Einsatz durch Datenstrukturtransformation und -adaption noch weiter optimieren [44].

Leider ist Laufzeiteffizienz heute in der Regel kein kritisches Thema. Martin Reiser hat darüber treffend geäußert: *"The hope is that the progress in hardware will cure all software ills. However, a critical observer may observe that software manages to outgrow hardware in size and sluggishness."* [56]

Green IT für Unternehmen

Im Unternehmensbereich mag Green IT putativ eine größere Rolle spielen als für Privathaushalte, da mit Energieeinsparungen potenziell auch Kosteneinsparungen verbunden sind. Außerdem könnte Green IT sich auch firmenimagespezifisch positiv auswirken. Welche Opportunitäten sich für Unternehmen tatsächlich ergeben und was für eine Rolle Green IT in Unternehmen praktisch spielt, wird in diesem Kapitel behandelt. Nicht für jedes Unternehmen, das IT-Systeme einsetzt, sind alle Green IT-Aspekte gleich bedeutend, weshalb dieses Kapitel in Teilbereiche unterteilt ist, welche die relevanten Gesichtspunkte von Green IT für Unternehmen abdecken.

Etwa die Hälfte der Unternehmen weltweit haben überhaupt keine Green IT- Maßnahmen implementiert, genauso groß ist der Anteil der Unternehmen, die den Strombedarf ihres IT-Sektors nicht separat messen [26].

Mögliche Ursache dafür ist, dass aus wirtschaftlicher Sicht Kosteneinsparungen durch Green IT-Maßnahmen in der Regel viel später ihre Gewinnschwelle (*break-even point*) erreichen als herkömmliche Investitionen. Im Kontrast dazu haben eine Reihe von Praxisstudien gezeigt, dass Green IT insgesamt positive Auswirkung auf die Wettbewerbsfähigkeit hat und zur Entwicklung neuer Opportunitäten am Markt führen kann. Momentan ist es jedoch so, dass die Politik generell nicht auf Einführung von Green IT-Maßnahmen drängt, so dass eine solche Einführung nur von höchster Verwaltungsebene in den Firmen kommen kann [64]. Erfolg durch die Einführung von Green IT ist nach Molla et al. [45] abhängig von Einstellung (Attitüde), Regelungen, Ausübung, Technologien und Verwaltung. Ein prinzipielles Problem ist der ansteigende Konsum stark netzwerkbelastender Leistungen, besonders im mobilen Sektor, der zu einer Agglomeration von energieintensiven Datenzentren führt. Der globale Leistungsbedarf von Datenzentren stieg von 2011 auf 2012 um etwa 63% auf 38GW [62], obwohl die Rechenleistung pro Watt stetig zunimmt (vgl Abb. 7).

2.4 Virtualisierung

Durch die Technik der Virtualisierung können mehrere VM[25] auf einem einzelnen Rechner emuliert werden. Auf jeder VM kann Software wie beispielsweise ein Webserver laufen. Ergo können durch dieses Prinzip auf einem Rechner parallel mehrere Webserver betrieben werden, wodurch die Gesamtzahl der notwendigen Server minimiert wird. Mit sinkender Zahl von physischen Rechnern nimmt auch der Wartungsbedarf ab [39].

Ein weiterer Vorteil ist, dass wie im Kapitel Verwendung und Effizienz von Algorithmen gezeigt, die Laufzeiteffizienz des Programmbetriebs steigt, da die virtuellen Maschinen parallelisiert betrieben werden können. Allerdings entspricht die VM gleichzeitig einer weiteren Abstraktionsebene, die zwischen Hardware und Software steht, womit der Effizienzbonus wiederum etwas einbüßen muss, wie im Kapitel Wahl der Programmiersprache erläutert. Insgesamt ist der Nutzen aber deutlich höher. So konnte beispielsweise ein mittelständisches Unternehmen mit 1000 Rechnern in einem Rechenzentrum durch Komplettumstellung auf Servervirtualisierung die Rechnerzahl auf 80 reduzieren[26] und die damit verbundenen Kosten um über 87%[27] senken [39].

Durch die Parallelisierung entsteht ein dienlicher Nebeneffekt, der sich zusätzlich positiv auf die Leistungsausbeute auswirkt und interdependent zu dem in den Kapiteln Stromversorgung, Prozessoren und Verwendung und Effizienz von Algorithmen behandelten Zusammenhang zwischen Auslastung und Energieeffizienz steht:

Die Energieeffizienz von Rechnern ist ungefähr proportional zum Auslastungsgrad wie in den vorangegangenen Kapiteln aufgezeigt ($\eta \propto xAuslastung$).

Typischerweise operieren Server aber mit einem Auslastungsspektrum (Abb. 8), dessen meiste Fläche im Intervall von etwa 20% bis 50% liegt. Mit Auslastungen ab zirka 65% verbringen sie verhältnismäßig wenig Zeit. Auf diese Weise sinkt ihre Gesamteffizienz. Diesem Effekt kann durch Virtualisierung entgegengewirkt werden. Durch eine optimale Anzahl parallel laufender VMs (die genaue Zahl richtet sich nach VM-Software und kann durch Automatisierungen noch gesteigert werden) kann man die CPU-Ausnutzung so regulieren, dass die Energieeffizienz des Serverbetriebs maximal wird [39]. Stehen

[25] Virtuelle Maschine

[26] Das entspricht einer Senkung um 92% ($1-\frac{80 Rechner}{1000 Rechner}$).

[27] $1-\frac{25891\,€}{204405\,€}$ (Werte vgl. [39])

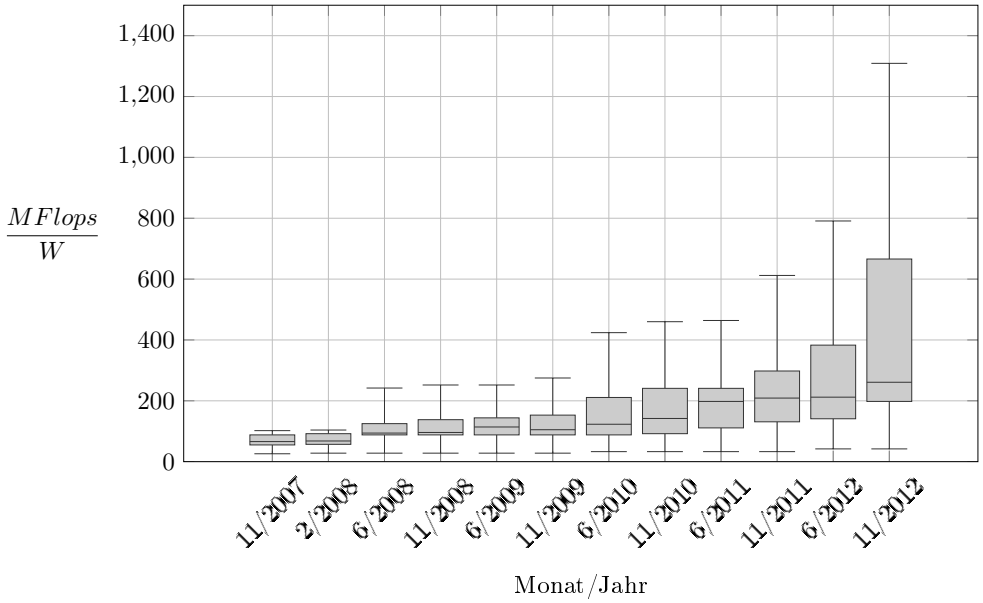


Abbildung 7: Statistische Verteilung der Entwicklung der Gleitkommaoperationen pro Watt in der *Green500*-Rechnerrangliste; ungefähr halbjährig von 2007 bis 2012 [57]

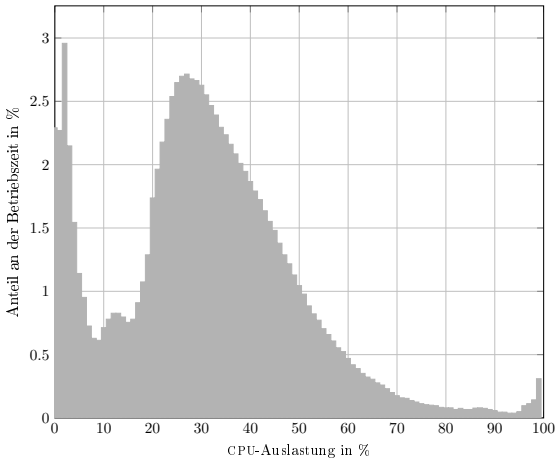


Abbildung 8: Durchschnittliche CPU-Auslastung von Servern nach Anteil an der Gesamtbetriebszeit [11]

einem alle Daten wie in Abb. 8 sowie der Effizienz nach Prozessorauslastung zur Verfügung, lässt sich die Gesamteffizienz berechnen durch:

$$\sum_{i=0}^{100} \eta_i T_i \tag{4}$$

Dabei ist $\eta(x)$ die Effizienz bei der Auslastung x und $T(x)$ die Zeit, die bei derselben Auslastung anteilig an der Gesamtbetriebszeit verbracht wird.

2.5 Rechenzentren

Speziell konzipierte Gebäude für den Betrieb dicht kumulierter Server bzw. Rechner werden Rechenzentren (oder Datenzentren; engl. *data center*) genannt. Diese sind notwendig, um den anwachsenden Speicherplatz-, Rechenkapazitäts- und Netzwerkverkehrbedarf verschiedenster IT-Dienstleistungsunternehmen zu gewährleisten. Sie gelten als *Enabling-Technologie* für die IT-Industrie und hatten 2016 ein alleiniges Marktvolumen von 152 Mrd. US Dollar [15] ($\simeq$ 138 Mrd. € bei einem Kurs von 1,1 USD : 1 EUR).

Ein einzelner Komplex benötigt mitunter soviel Strom wie 25.000 Haushalte [52]. In den USA machen Rechenzentren zirka 2% des Jahresenergiehaushalts aus [37]. Weltweit hatten sie im Jahr 2012 einen Energiebedarf von 270TWh und im Zeitraum von 2007 bis 2012 eine kumulierte jährliche Wachstumsrate von 4,4% [25]. Folglich ist es im ökologischen und ökonomischen Interesse, diesen Energiebedarf so weit wie möglich zu optimieren.

Der prinzipielle Energiefluss im Rechenzentrum ist in Abb. 9 schematisch dargestellt, während der Anteil der (zusammengefassten) Komponenten am Gesamtstrombedarf als Kreisdiagramm (Abb. 10) abgebildet ist. Um die in erster Linie von den *Bladeservern* (auch *Serverblades*)[28] in den *Racks*[29] aufkommende Wärme abzutransportieren, ist eine aufwendige Kühlung nötig, die allein die Hälfte des Strombedarfs der Rechenzentren ausmacht. Umso ef-

[28] In einen *Rack* einschieb- und herausziehbarer Server.

[29] Standardisierte Regale zur kompakten Verstauung möglichst vieler *Serverblades*.

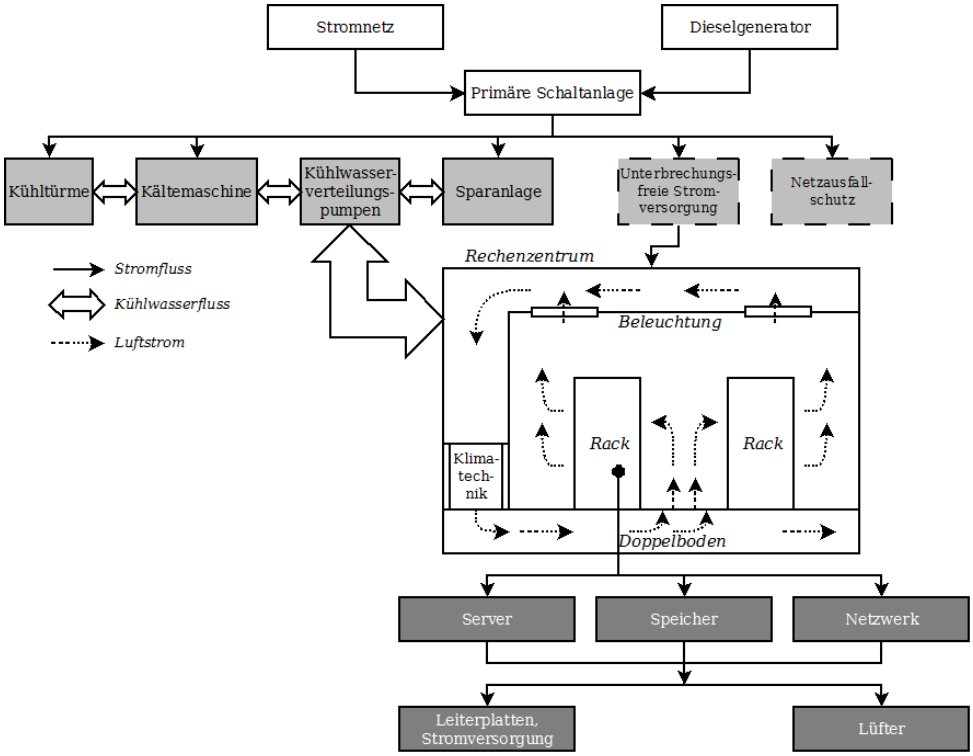


Abbildung 9: Energieflussprinzip im Rechenzentrum [15]

fizienter die Rechnersysteme arbeiten, desto weniger Kühlung würde nötig sein. Die Stromversorgung des Komplexes und der IT selbst ist erheblich verlustreich (in Abb. 10 mit 11% aufgeführt), da die vielen vorhandenen Transformatoren beim Umwandeln der eingehenden Spannung in die benötigten, unterschiedlich hohen Gleichspannungen nicht mit 100% Effizienz arbeiten (vgl. Kapitel Stromversorgung).

2.5.1 Modellierung

Für die Steigerung der Energieeffizienz eines Rechenzentrums wird meist folgende Prozesskette angewandt [15]:

1. Die Energiecharakteristiken der Teilkomponenten werden analysiert, um relevante Stromverbraucher aufzudecken. Hier kommen auch intelligente PDU[30] zum Einsatz, die netzwerkintegriert einzelne *Sockets* vermessen können und mit optionaler Temperatursensorik ausgestattet sind.

2. Mithilfe von Maschinenlernen, Regressionsanalyse oder anderer Methoden wird ein Ist-Modell erstellt. Umso mehr und umso genauer die Parameter bekannt sind, desto akkurater ist eine Modellierung möglich.

3. Das entworfene Modell wird validiert.

4. Liegt ein genügend genaues Modell vor, lässt sich dieses als Basis für Optimierungsansätze verwenden, wie beispielsweise die in den vorangegangenen Kapiteln behandelte Virtualisierung (Kapitel Virtualisierung), DVFS (Kapitel Prozessoren) oder verbesserte Softwarealgorithmen (Kapitel Verwendung und Effizienz von Algorithmen).

Bei der Modellierung werden nicht nur Server autark betrachtet, es werden die separaten Serverkomponenten (CPU, Arbeitsspeicher usw.) berücksichtigt und bei der Analyse von Software einzelne Features und Algorithmen (vgl. Kapitel Verwendung und Effizienz von Algorithmen).

Hauptsächlich werden *additive* und *systemauslastungsbezogene* (engl. *system utilization based*) Modelle genutzt. Additive Modelle versuchen möglichst genau einzelne Komponente zu repräsentieren, deren Energiebedarf zu einem Gesamtbedarf aggregiert wird. Ein einfaches Beispiel ist die Formel

$$E(A) = E_{CPU}(A) + E_{Speicher}(A) \qquad (5)$$

für die Bestimmung des Gesamtenergiebedarfs von Algorithmus A, der sich aus zwei Teilenergiebedarfen von CPU und Speicher errechnet [58]. Auch Modelle vom selben Typ können sich stark unterscheiden. So gibt es welche, die wie in Gleichung 5 den Algorithmus als Abszisse nutzen, aber auch solche, die

[30] Power Distribution Unit (deutsch: Stromverteilereinheit)

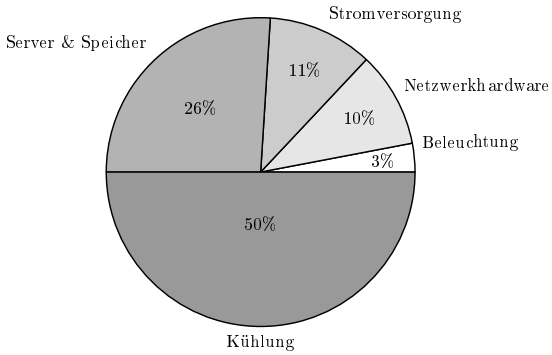


Abbildung 10: Exemplarische Verteilung des Strombedarfs im Rechenzentrum auf Teilbereiche; Werte können von Fall zu Fall variieren [15]

beispielsweise VMs als elementar betrachten, wie [16]:

$$P_{Server} = P_{Basis} + \sum_{i=1}^{n} P_{VM}(i) \tag{6}$$

P_{Basis} ist dabei das Leistungsniveau des Systems ohne VMs.

Systemauslastungsbezogene Modelle konzentrieren sich auf die Modellierung des Auslastungszustands einzelner Komponenten iteriert zum Gesamtsystem. In fast jeder Formel findet sich ein Term, der die Leistung im Leerlauf beinhaltet, wie beispielsweise [15]:

$$P_u = (P_{max} - P_{Leerlauf})(2u - u^r) + P_{Leerlauf} \tag{7}$$

Dabei ist u die Auslastung und r ein experimentell bestimmter Parameter, der die Genauigkeit der Gleichung gegen quadratische Verluste kalibrieren soll.

Ist der Leistungsbedarf P bei einer Auslastung u zum Zeitpunkt t bekannt, lässt sich der Energiebedarf E im Zeitintervall t_0 bis t_1 berechnen durch [41]:

$$E = \int_{t_0}^{t_1} P(u(t))dt \tag{8}$$

2.5.2 Kennzahlen

Bei den zusammenfassenden Kennzahlen für die Effizienzmessung des Rechenzentrums als Ganzes scheinen zwei miteinander mathematisch verknüpfte in der Praxis populär: PUE[31] und DCiE[32]. DCiE gibt an, wie viel Leistung die IT anteilig an der Gesamtleistung des Rechenzentrums benötigt [42, 47]:

$$DCiE = \frac{P_{IT}}{P_{Gesamt}} \tag{9}$$

Die beiden dimensionslosen Kennzahlen sind insofern verwandt, weil eine jeweils reziprok zu der anderen ist:

$$PUE = \frac{1}{DCiE} = \frac{P_{Gesamt}}{P_{IT}} \tag{10}$$

In der Literatur wird für DCiE ein Wert von 0,5 als *Standard*, 0,7 als *gut* und 0,9 als *besser* bezeichnet. Neben den genannten IT-relevanten Kennzahlen gibt es noch weitere für z.B. die Verwendung regenerativer Energien, für die Kühlungs- und Klimasysteme [42]. Interessant für die IT ist möglicherweise noch die Metrik SWaP[33], die Rechenleistung sowie Platzverbrauch der Racks berücksichtigt [57]:

$$SWaP = \frac{Rechenleistung}{Platzverbrauch * Leistung} \tag{11}$$

Die Einheit für SWaP variiert je nach genutzten Benchmarks und je nachdem, wie der Platzverbrauch der Serverracks gemessen wird. So kann beispielsweise eine Einheit wie $\frac{Flops}{W * RU}$ bei der Rechnung herauskommen (RU[34] ist dabei ein Maß für den Platzverbrauch von Racks).

In die Modellierung fließen auch Verluste ein, die unterteilt werden nach Verhältnis zum Gesamtverbrauch des Rechenzentrums. Es gibt Verluste, die *konstant* sind, d.h. deren Wert gleich bleibt, wenn sie angeschaltet sind, unabhängig vom Lastzustand des Betriebs. Ein Beispiel dafür ist die Ladestanderhaltung der unterbrechungsfreien Stromversorgung. *Proportionale* Verluste treten bei der Kühlung auf, da ihre Intensität erhöht werden muss, wenn auf den Rechnersystemen stärkere Last liegt. Durch die physikalischen Eigenschaften von Wechselstrom und den damit zusammenhängenden Blindwiderstand bedingt, ist die dritte Art von Verlusten von *quadratischer* Art nach der Gesetzmäßigkeit 12, bei der P Wirkleistung, R Widerstand, U Spannung und φ Phasenverschiebungswinkel der Wechselspannung ist

$$P_{Verlust} = I^2 R = (\frac{P}{U \cdot \cos\varphi})^2 \cdot R \tag{12}$$

[31] Power Usage Effectiveness (deutsch: Leistungsnutzeneffizienz)

[32] Data Center Infrastructure Efficiency (deutsch: Datenzentruminfrastruktureffizienz)

[33] Space, Watts and Performance (deutsch: Raum, Watt und Performanz)

[34] Rack Unit (deutsch: Rackeinheit)

und welche vor allem bei Transformatoren, Leitungen und damit auch der Stromversorgung auftritt [47, 12].

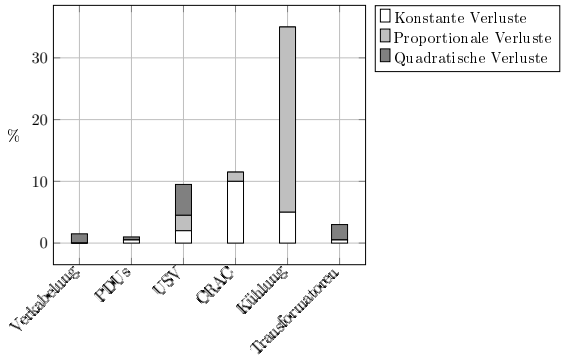


Abbildung 11: Verteilung der Verlusttypen auf verschiedene Bereiche des Rechenzentrums [47]

Wie unterschiedlich groß der Einfluss der drei Verlustarten auf die Bereiche des Rechenzentrums ist, zeigt Abb. 11. Da die Kühlung von proportionalen Verlusten betroffen ist und zusätzlich den größten Anteil des Strombedarfs im Rechenzentrum ausmacht, eignet sie sich als Fokus von Optimierungsüberlegungen.

2.5.3 Kühlung

Bei der Einrichtung der Kühlanlagen stellt sich vorab die Frage, einen zentralisierten Bautyp oder viele verteilte einzelne CRAC[35]-Einheiten zu verwenden. Zwar haben CRACs den Vorteil, einfacher skalierbar zu sein (z.B. bei Vergrößerung der Anlage), aus Energieeffizienzsicht liegt jedoch die Zentralkühlung mit Wasserkreislauf wie in Abb. 9 deutlich vorn. Zudem sollte die Kühlung Technologien enthalten, die ihr es erlauben, sich dynamisch an Umgebungstemperatur und Systemkühlbedarf anzupassen, um zusätzlich effizienter zu arbeiten [42].

Schon bei der Planung lässt sich die Kühleffizienz steigern, indem ein Standort in einer möglichst kalten Klimaregion der Erde gewählt wird. Die dadurch erzielten Vorteile können signifikant sein: Ein Außentemperaturunterschied $\Delta T_{außen} = 14°C$ (Temperaturen: 8°C; 22°C) verursacht eine Differenz bei der Energieeffizienz von fast 30%.[36] Eine andere Quelle nennt für die Wahl eines *angemessenen* (im engl. Originaltext "reasonable") Standorts eine mögliche Energieeffizienzsteigerung von 12% bis 15% [57]. Firmen, die bei der Standortwahl (aus verschiedensten Gründen) weniger flexibel sind, bezeichnen diesen Vorteil als *unfair*, da bei den etablierten Kennzahlen DCiE und PUE Umwelteinflüsse außen vor bleiben [47].

Eine Studie hat gezeigt, dass knapp 40% der befragten Verantwortlichen Maßnahmen veranlassten, die Effizienz durch verbesserte Kühlung zu steigern (32% durch Doppelbodenklimatechnik, 7,7% durch zentralisierte Wasserkühlung). Die Hälfte der Befragten gab an, durch Virtualisierung Energie zu sparen [46]. Meine Supposition ist, dass der Aufwand für eine Umstellung auf Virtualisierung geringer ist als eine Umstellung auf zentralisierte Wasserkühlung (wofür wohl in vielen Fällen ein Komplettumbau nötig wäre). Außerdem ließe ich der Kühlbedarf auch durch reduzierte Rechenlasten verringern.

In manchen Fällen lässt sich das Problem des hohen Energiebedarfs von Rechenzentren komplett beseitigen, nämlich dann, wenn der Komplex unabhängig von externer Stromversorgung ist und seinen eigenen Strom generiert, aus Green IT-Sicht nach Möglichkeit durch regenerative Energien wie beispielsweise Solar-, Wind- und Wasserkraft. Auch aus abgeleiteter Wärme kann ein Teil der Wärmeenergie wieder nutzbar gemacht werden [42].

2.6 Software Reuse

Wiederverwendung von bestehendem Quelltext wird in der Softwareentwicklung als *Reuse* bezeichnet. Reuse hat zum Ziel, Entwicklungsaufwand zu minimieren. Mit geringerem Entwicklungsaufwand lassen sich Kosten und damit Energie sparen. Werden zusätzlich besonders effiziente Algorithmen durch Wiederverwendung in neue Software eingebaut, kann auch die Laufzeiteffizienz von ihr profitieren. Auch könnten Schwächen effizienter Programmiersprachen, die aber mehr Programmzeilen benötigen als höher abstrahierte Sprachen, ausgeglichen werden (vgl. Kap. Wahl der Programmiersprache). Gefahren, die bei der Wiederverwendung von Programmkode auftreten können, sind [38]:

a) Der wiederverwendete Quelltext enthält Fehler, die die neue Software infiltrieren.

b) Der wiederverwendete Quelltext ist so lizenziert, dass es zu juristischen Konflikten durch Reuse kommen kann.

c) Der wiederverwendete Quelltext ist in der vorliegenden Form nicht eins zu eins übernehmbar und Anpassungen führen zu größerem Aufwand als es ohne Reuse der Fall gewesen wäre.

Eine Methodik von Software-Reuse ist es, den eigenen Quelltext nach wiederverwendbaren Teilen zu durchforsten. Hier kann der Aufwand des Suchens den Nutzen übersteigen. Zusätzlich können in diesem Fall a) und c) für diesen Programmteil zutreffen [38].

[35] Computer Room Air Conditioner (deutsch: Rechnerraumklimaanlage)

[36] $\Delta\eta = \frac{P_1}{P_2} - 1 = \frac{490524W + 1467842W}{348087W + 1163344W} - 1 = \frac{1958366W}{1511431W} - 1 = 0,2957$ (Werte vgl. [47])

In der Praxis überwiegen aber die Vorteile. So gab zum Beispiel *Hewlett-Packard* an, durch Software-Reuse-Projekte höhere Programmqualität, kürzere Produkteinführungszeit und Kostenvorteile erzielt zu haben [65].

Die Wiederverwendung von kleineren Quelltextstücken, gemessen in SLOC[37], ist effizienter als mit großen Portionen. In der objektorientierten Programmierung hat das Wiederverwenden von Klassen zusätzlich den Vorteil, dass der geschriebene Kode selbst sich leichter wiederverwenden lässt [4].

Heute gibt es eine große Anzahl quelloffener (*open-source*) Bibliotheken (*libraries*), von denen viele auch in der Produktion verwendet werden. Die Entwicklungsaufwandsersparnis liegt in der Praxis durch Verwendung von Reuse etwa bei 30% [65].

3 Im mobilen Sektor

Im mobilen Bereich ist man als Smartphone-Nutzer quasi selbst betroffen von Green IT: Die Betriebszeit des Akkus bis zur vollständigen Entladung schwankt, je nach Nutzungsverhalten und welche Energiesparoptionen getroffen werden. Smartphones mit dem Android-Betriebssystem beispielsweise geben Rückmeldung und zeigen Statistiken (Abb. 12) darüber an, wie sich die Batterie über die Zeit entlädt und welche Applikationen (Apps) und Systeme (Display, Datenverkehr etc.) daran in welchem Umfang beteiligt sind. Außerdem stehen dem Nutzer in Android (von Version zu Version) verschiedene Möglichkeiten zur Verfügung, den Stromverbrauch des Handys zu minimieren. Beispiele hierfür sind:

- Die Nutzung von 2G statt 3G oder 4G im Ruhezustand.
- Die Nutzung von 2G statt 3G oder 4G generell.
- Optionen zur Verringerung des Datenverkehrs (z.B. keine Bilder laden).
- Geringere Bildschirmhelligkeit.
- WLAN[38] im Ruhezustand deaktivieren.

Der Strombedarf aller Smartphones auf der Welt ist weniger gravierend aus Green-IT-Sicht als der Datenverkehr, den sie verursachen, da der Leistungsbedarf von Smartphones eher gering ist (<1W), der Leistungsbedarf von Rechenzentren für die Bereitstellung der Daten jedoch sehr hoch (vgl. Kap. Rechenzentren) [7]. 4G ist von allen Datenübertragungsarchitekturen der mit dem höchsten Stromverbrauch, an zweiter Stelle 3G, gefolgt von WLAN und letztlich 2G. Im Vergleich zu WLAN ist 4G (LTE[39]) 23-mal weniger energieeffizient [29].

Die Netzwerkgenerationen 3G und 4G arbeiten mit drei Energiezuständen: *IDLE*, *FACH*[40] und *DCH*[41]. Der IDLE-Zustand entspricht einem Ruhezustand, in dem kein Strom gebraucht wird. DCH ist ein Modus für hohen Datendurchsatz, der den höchsten Energiebedarf hat und FACH ist ein intermediärer Zustand. Nachdem ein Smartphone im DCH- oder FACH-Modus Datenpakete empfangen hat, verweilt es eine feste Zeitspanne, die sog. *Tail Time*, weiterhin in demselben Modus bis zum Ablauf einer vom Netzbetreiber definierten Zeitbeschränkung [22, 27]. Bei allen Mobilfunknetzarchitekturen sowie beim WLAN macht die *Tail Time* etwa 50% des Energiebedarfs des Datenverkehrs aus [29]. Eine Optimierungsmethode der Energieeffizienz im mobilen Bereich nutzt diese Technik aus, indem der Transfer von Datenpaketen so zeitlich geplant wird, dass das System möglichst wenig Zeit in DCH und FACH verbringt und nach Möglichkeit viel Zeit in IDLE. Ein Teil des Datenverkehrs fällt auf sog. Kontrollpakete der jeweiligen Netzwerkprotokolle. Eine Studie hat gezeigt, dass TCP[42] den Transfer besonders vieler Kontrollpakete hervorruft, die das System vom Ruhezustand auf DCH oder FACH schalten, ohne dass weitere Kommunikation erfolgt. TCP ist also im mobilen Bereich für die Energieeffizienz von 3G-/4G-Netzen ein ungünstiges Kommunikationsprotokoll [22]. 15% der Akkuladung und 98% des Datenverkehrs entfallen bei Smartphones auf Werbeanzeigen [55] (vgl. Kap. Verwendung und Effizienz von Algorithmen). Problematisch dabei ist, dass diese in den letzten Jahren stetig zugenommen haben. Prognosen gaben an, dass sie 2017 die Ausgaben für Fernsehwerbung übertreffen würden. Außerdem fallen $\frac{1}{3}$ aller Werbemittel bereits auf den Smartphone-Sektor und etwa 50% aller Apps enthalten Werbung [23].

Aus Green-IT-Sicht wäre es günstig, würde das durch Handywerbung verursachte Datenvolumen abnehmen. Bei einigen Nutzern beträgt dieses täglich bis zu 1000MB, während bei wiederum einigen Nutzern (nicht zwingend einer Schnittmenge der Nutzer mit 1000MB/d Datenvolumen) nur 10% der Datentransfers während aktiver Nutzung abgewickelt wurden [19].

Werbeanzeigen verursachen nicht nur hohen Datenverkehr, sondern können auch sog. *Energy Bugs* enthalten: Fehlerhaft geschriebener Programmkode, der verhindert, dass das Smartphone in den Ruhemodus schaltet. Bei der Untersuchung einer Vielzahl von Applikationen hat sich ergeben, dass bis auf wenige Ausnahmen alle Energy Bugs enthalten, die meist durch ungenutzte Hintergrunddienste, oft auch Programmkode, der für die Verwaltung von Werbung zuständig ist, resultieren. Langfristig führt eine zu ho-

[37] Source Lines of Code (deutsch: Quelltextzeilen)

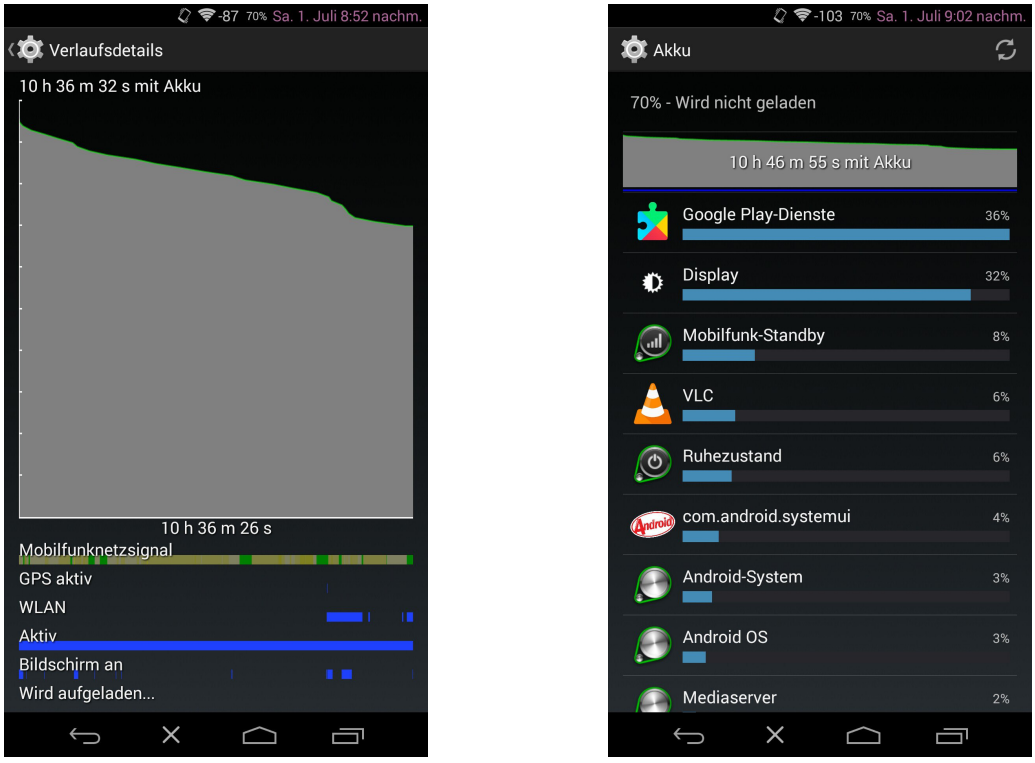
[38] Wireless Local Area Network (deutsch: kabelloses lokales Netzwerk)

[39] Long-Term Evolution (deutsch: langfristige Evolution)

[40] Forward Access Channel (deutsch: Voraus Zugriffskanal)

[41] Dedicated Channel (deutsch: dedizierter Kanal)

[42] Transmission Control Protocol (deutsch: Übertragungssteuerungsprotokoll)



(a) Verlauf Akkuladestand

(b) Rangliste der Verbraucher

Abbildung 12: Bildschirmfotos von der Energieverwaltung unter Android 4.4.4; Messwerte entsprechen exemplarisch denen des Smartphones vom Verfasser und geben keinesfalls Durchschnittswerte an.

he Leistungsaufnahme zu erhöhtem Verschleiß beim Akku [9].

Für die Ausführung von Programmen auf Smartphones gelten generell dieselben Effizienzprinzipien wie auch für ortsfeste Rechner (Kap. Prozessoren): So spielt der *Race to Idle* (Kap. Verwendung und Effizienz von Algorithmen) ebenfalls eine Rolle und wird begünstigt durch die Verwendung von Mehrkern-CPUs, wobei mehr Kerne (untersucht wurde bis zur Anzahl 4) effizienter arbeiten [67].

Abgesehen vom Datenverkehr spielt der Produktlebenszyklus von Smartphones für Green IT auch eine große Rolle. IKT-Systeme machen etwa 2% der globalen CO_2-Emissionen aus, bis zum Jahr 2020 sollen es etwa 2,8% werden. Die Produktion eines Handys verursacht dabei zirka 24kg CO_2-Emissionen. Negativ wirkt sich vor allem der kurze typische Lebenszyklus von 1 bis 2 Jahren pro Gerät aus [21].

4 Politische Situation

Überraschenderweise sind IT-relevante Standards für Energieeffizienz in der EU beinahe nicht-existent. Sporadisch wird für die Kennzeichnung effizienter Geräte der aus den USA stammende *ENERGY STAR™* -Aufkleber verwendet. Das EU-Äquivalent dazu wäre vielleicht die Richtlinie 2012/27/EU, die sog. Energieeffizienzrichtlinie. In dieser werden zwar Reglementierungen für verschiedene technische Anlagen und beispielsweise auch Gebäude und deren Wärmeisolierung (somit indirekt, aber wenig relevant auch für Rechenzentren), nicht jedoch für Rechner, Laptops, PSUs o.ä. definiert. Gleichermaßen die Richtlinie zur Kennzeichnung des Energieverbrauchs, aktuellste Version von 2010 ist die EU-Richtlinie 2010/30/EU, von der eine Reihe von Haushaltsgeräten betroffen sind (auch Fernseh-fähige Monitore), die mit einer *Energieeffizienzklasse* benotet werden. Rechner, Laptops usw. werden von ihr jedoch ebenfalls nicht berücksichtigt. Von einer neuen, für das Jahr 2020 von der EU geplanten Richtlinie 20/20/20 werden möglicherweise Rechenzentren betroffen sein. Ihr Ziel soll es sein, den Einsatz von 400V Gleichstrom als Eingangsversorgung für Rechenzentren vorzuschreiben und somit verlustreiche Transformatoren zu umgehen (vgl. Kap. Stromversorgung & Rechenzentren), die Energieeffizienz zu steigern, den Stromverbrauch zu senken und ultimativ den CO_2-Ausstoß um 20% zu reduzieren [17].

4.1 Elektronischer Abfall

Etwas strenger hat die EU den Umgang mit in der IT vorkommenden Gefahrstoffen reglementiert:

In der Verordnung (EG) Nr. 1907/2006 (REACH[43]-Verordnung) wird die Verwendung von Chemikalien begrenzt und geregelt, EU-Richtlinie 2011/65/EU (RoHS[44]) beschränkt die Verwendung von Gefahrstoffen explizit im IT-Bereich und WEEE[45]-Richtlinie 2012/19/EU enthält Regelungen zur Entsorgung von elektrischen und elektronischen Geräten.

Paradox ist, dass die Länder, die die meisten Vorgaben und Gesetzte für dem Umgang mit Elektroschrott haben, das sind mitunter USA und China, nicht die besten Recycling-Raten für diese Abfälle haben (siehe Tabelle 2). Ursache dafür sind mög-

Region	Anzahl relevanter Verordnungen in Kraft	Recyclingrate
Japan	4	65,7%
EU	5	37,0%
USA	87	29,0%
China	17	18,2%

Tabelle 2: Vergleich der Recyclingraten von Elektroschrott [33]

licherweise unterschiedlich harte Strafen, die in Japan, das die Rangliste der Recycling-Raten anführt, am höchsten sind, während die USA, welche weltweit am meisten Elektroschrott produzieren, diesen noch nicht als *ernsthaftes Problem* ansehen [33].

Elektroschrott ist die weltweit am stärksten anwachsende Art von Abfall und kann im Vergleich zu allen anderen Abfallarten (beispielsweise Autobatterien, Blechdosen, Reifen, Glas, Holz, Papier) schlechter recycelt werden. Nach eine UN-Studie werden für die Produktion eines Rechners mit Bildschirm 240kg fossiler Brennstoffe, 22kg Chemikalien und 1,5t Wasser verbraucht. IT-Geräte werden prinzipiell nicht so entworfen, leicht recyclebar zu sein. Eine Möglichkeit Produzenten zu motivieren leichter recyclebare Geräte herzustellen wäre daher, sie die Kosten für die Entsorgung übernehmen zu lassen [32]. Auch geplante Obsoleszenz könnte so gekontert werden.

4.2 Bildungssituation

Häufigster Faktor für die Entscheidung gegen eine Implementierung Green IT oder gegen Green IT-Prinzipien, obwohl umfassende Kenntnisse über die Thematik bestehen, scheinen wirtschaftliche Interessen, vor allem auch Konsumorientierung, zu sein. Um Green IT praktisch zu realisieren, müssen Kenntnisse und Bewusstsein dafür vorhanden sein [8, 24]. Beides kann auch erzieherisch erlernt werden und gleichzeitig Green IT in den globalen Zusammenhang des Klimawandels und der Nachhaltigkeit platzieren, womit sich eine Gruppe an der *Königlich Technischen Hochschule Stockholm* beschäftigte [66].

Eine kritische Rolle spielt Green IT für alle Studierenden, die nach ihrem Studium Green IT-relevante Entscheidungen treffen werden. Betroffen sind dabei nicht nur Studenten, die verwaltende Positionen in IT-Unternehmen erlangen könnten, sondern auch Programmierer, die im Allgemeinen schlecht über Green IT Bescheid wissen (vgl. Kap. In der Softwareentwicklung). Green IT benötigt daher einen Platz im Curriculum solcher Studiengänge [18].

5 Fazit

In den hier untersuchten Bereichen spielt Green IT unterschiedliche Rollen: Im Unternehmensbereich beispielsweise werden Energieeffizienzsteigerungen oft zur Gewinnoptimierung angewandt, als langfristige Investition mit der letztendlich Stromkosten gespart werden können. Vor allem im Rechenzentrum ist diese Perspektive allgegenwärtig. Manche Zweige der IT sind weniger aufgeklärt und sich weniger bewusst über die Auswirkungen, die sie auf die Energieeffizienz beim Betrieb von Rechnersystemen haben. In der Softwareentwicklung ist dies nachweislich der Fall. Für die Umsetzung von Green IT spielen wirtschaftliche Interessen eine Schlüsselrolle. IT-Bereiche, die selbst kaum etwas davon spüren, wenn sie ineffiziente Systeme betreiben oder hohen Energieeinsatz verursachen, wie Smartphones bei Rechenzentren, haben wenig Anreiz, Green IT zu implementieren. Die Politik hätte hier die Möglichkeit zum Beispiel das monatliche Datenvolumen von Mobilfunknutzern gesetzlich zu begrenzen, falls sie dies für nötig hielte. An anderer Stelle hat sich gezeigt (Kap. Politische Situation), dass eine weitere Schwierigkeit für die Implementierung von Green IT ist, dass Studienabsolventen, die für diesen Bereich verantwortlich sein werden, gar nicht über ihn an den Hochschulen aufgeklärt werden. Parallel steht im Studiengang Betriebswirtschaftslehre in Deutschland Wirtschaftsethik nicht im Curriculum. Die Politik hätte hier keinen Handlungsbedarf, wenn der IKT-Sektor eine vernachlässigbare Rolle im Umweltschutz und in der Nachhaltigkeit spielte. Im Rahmen dieser Arbeit hat sich das Gegenteil erwiesen und es wurden an allen relevanten Stellen, auch in der Softwareentwicklung, Optionen zur Energieeffizienzsteigerung aufgezeigt.

[43] Registration, Evaluation, Authorisation and Restriction of Chemicals (deutsch: Registrierung, Evaluation, Autorisierung und Beschränkung von Chemikalien)

[44] Restriction of Hazardous Substances Directive (deutsch: Gefahrstoffrestriktionsanordnung)

[45] Waste of Electrical and Electronic Equipment (deutsch: Abfall elektrischer und elektronischer Geräte)

Literatur

[1] Aebischer, Bernard ; Huser, Alois: *Energy Efficiency of Computer Power Supply Units*. Ittigen, Schweiz : Bundesamt für Energie, 2002

[2] Albers, Susanne ; Antoniadis, Antonios: Race to Idle: New Algorithms for Speed Scaling with a Sleep State. In: *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*. Philadelphia, PA,

USA : Society for Industrial and Applied Mathematics, 2012 (SODA '12), S. 1266–1285

[3] Albert, Marco: Stromverbrauch: 40 LC-Displays von 19 bis 30 Zoll. In: *PC Games Hardware* 07 (2010)

[4] Anguswamy, Reghu ; Frakes, William B.: A Study of Reusability, Complexity, and Reuse Design Principles. In: *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. New York, NY, USA : ACM, 2012 (ESEM '12), S. 161–164

[5] Aruoba, S. B. ; Fernández-Villaverde, Jesús: A Comparison of Programming Languages in Macroeconomics. In: *Journal of Economic Dynamics and Control* 58 (2015), S. 265–273

[6] Asaduzzaman, Abu ; Lee, Hin Y.: GPU Computing to Improve Game Engine Performance. In: *Journal of Engineering and Technological Sciences* 46 (2014), Nr. 2

[7] Balasubramanian, Niranjan ; Balasubramanian, Aruna ; Venkataramani, Arun: Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. In: *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*. New York, NY, USA : ACM, 2009 (IMC '09), S. 280–293

[8] Bandi, Rajendra K. ; Bose, Anik K. ; Saxena, Ashay: Exploring Green IT Awareness and Adoption Among Indian Students. In: *Proceedings of the 2015 ACM SIGMIS Conference on Computers and People Research*. New York, NY, USA : ACM, 2015 (SIGMIS-CPR '15), S. 87–96

[9] Banerjee, Abhijeet ; Chong, Lee K. ; Chattopadhyay, Sudipta ; Roychoudhury, Abhik: Detecting Energy Bugs and Hotspots in Mobile Apps. In: *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. New York, NY, USA : ACM, 2014 (FSE 2014), S. 588–598

[10] Barroso, Luiz A.: Comprehensive Optimization of Energy Consumption and Delay Performance for Green Communication in Internet of Things. In: *ACM Queue* 3 (2005), S. 49–53

[11] Barroso, Luiz A. ; Hölzle, Urs: The Case for Energy-Proportional Computing. In: *Computer* 40, Nr. 12 (2007), Dezember, S. 33–37

[12] Benedict, E. ; Collins, T. ; Gotham, D. ; Hoffman, S. ; Karipides, D.: *Losses in electric Power Systems*. West Lafayette, IN, USA : Purdue University, 1992

[13] Bhattacharya, Pamela ; Neamtiu, Iulian: *Assessing Programming Language Impact on Development and Maintenance: A Study on C and C++*. New York, NY, USA : ACM, 2011

[14] Chowdhury, Shaiful A. ; Hindle, Abram: GreenOracle: Estimating Software Energy Consumption with Energy Measurement Corpora. In: *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR), Austin, TX* (2016), S. 49–60

[15] Dayarathna, Miyuru ; Wen, Yonggang ; Fan, Rui: Data Center Energy Consumption Modeling: A Survey. In: *IEEE Communications Surveys and Tutorials* 18, Nr. 1 (2016), S. 732–794

[16] Elnozahy, E. N. ; Kistler, Michael ; Rajamony, Ramakrishnan: Energy-Efficient Server Clusters. In: *Power-Aware Computer Systems: Second International Workshop*. Berlin, Heidelberg : Springer, 2003 (PACS 2002), S. 179–197

[17] Europäisches Parlament: *Richtlinie 2012/27/EU*. `http://eur-lex.europa.eu/legal-content/DE/TXT/?uri=celex%3A32012L0027`. Version: 2012

[18] Evangelou, Evangelos K. ; Pagge, Jenny: The urge for GREEN IT courses at Universities and Technical Institutes. In: *International Conference on Interactive Mobile Communication Technologies and Learning (IMCL)* (2015), S. 390–392

[19] Falaki, Hossein ; Mahajan, Ratul ; Kandula, Srikanth ; Lymberopoulos, Dimitrios ; Govindan, Ramesh ; Estrin, Deborah: Diversity in Smartphone Usage. In: *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*. New York, NY, USA : ACM, 2010 (MobiSys '10), S. 179–194

[20] Folegnani, Daniele ; Gonzalez, Antonio: Power Efficient Hardware Transactional Memory: Dynamic Issue of Transactions. In: *28th Annual International Symposium on Computer Architecture* (2001)

[21] Forster, Colin ; Dickie, Ian ; Maile, Graham ; Smith, Howard ; Crisp, Malcolm: Understanding the Environmental Impact of Communication Systems / eftec. Version: 2009. `https://www.ofcom.org.uk/__data/assets/pdf_file/0026/31886/environ.pdf`. 2009 (Version 03). – Final Report

[22] Govindarajan, C. ; Sengupta, S. ; De, P. ; Mitra, B. ; Chakraborty, S.: Role of network control packets in smartphone energy drainage. In: *8th International Conference on Communication Systems and Networks*, 2016 (COMSNETS 2016), S. 1–2

[23] Gui, Jiaping ; Li, Ding ; Wan, Mian ; Halfond, William G. J.: Lightweight Measurement and Estimation of Mobile Ad Energy Consumption. In: *Proceedings of the 5th International Workshop on Green and Sustainable Software*. New York, NY, USA : ACM, 2016 (GREENS '16), S. 1–7

[24] Håkansson, Maria ; Sengers, Phoebe: Beyond Being Green: Simple Living Families and ICT. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA : ACM, 2013 (CHI '13), S. 2725–2734

[25] Heddeghem, Ward V. ; Lambert, Sofie ; Lannoo, Bart ; Colle, Didier ; Pickavet, Mario ; Demeester, Piet: Trends in worldwide ICT electricity consumption from 2007 to 2012. In: *Computer Communications* 50 (2014), S. 64–76

[26] Hildebrand, Knut (Hrsg.): *IT im Mittelstand*. Heidelberg : dpunkt Verlag, 2012

[27] Hoque, Mohammad A. ; Siekkinen, Matti ; Khan, Kashif N. ; Xiao, Yu ; Tarkoma, Sasu: Modeling, Profiling, and Debugging the Energy Consumption of Mobile Devices. In: *ACM Computing Survey* 48 (2015), Nr. 3, S. 39:1–39:40

[28] Hotta, Y. ; Sato, M. ; Kimura, H. ; Matsuoka, S. ; Boku, T. ; Takahashi, D.: Profile-based optimization of power performance by using dynamic voltage scaling on a PC cluster. In: *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*, 2006

[29] Huang, Junxian ; Qian, Feng ; Gerber, Alexandre ; Mao, Z. M. ; Sen, Subhabrata ; Spatscheck, Oliver: A Close Examination of Performance and Power Characteristics of 4G LTE Networks. In: *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*. New York, NY, USA : ACM, 2012 (MobiSys '12), S. 225–238

[30] Irani, Sandy ; Shukla, Sandeep ; Gupta, Rajesh: Algorithms for Power Savings. In: *ACM Trans. Algorithms* 3 (2007), November, Nr. 4

[31] Islam, Syed ; Noureddine, Adel ; Bashroush, Rabih: *Measuring Energy Footprint of Software Features*. New York, NY, USA : IEEE, 2016

[32] Jaragh, Mansour ; Boushahri, Jenan: The e-Waste Impact. In: *Proceedings of the First Kuwait Conference on e-Services and e-Systems*. New York, NY, USA : ACM, 2009 (eConf '09), S. 4:1–4:7

[33] Jesus Cardoso Correia, Auro de ; Oliveira Neto, Geraldo C. ; Silva, Paulo C.: Comparative Analysis of Regulatory Instruments in Reverse Logistics for Electrical and Electronic Wastes. In: *Proceedings of the 7th International Conference on Management of Computational and Collective intElligence in Digital EcoSystems*. New York, NY, USA : ACM, 2015 (MEDES '15), S. 207–213

[34] Joumaa, Chibli ; Kadry, Seifedine: Green IT: Case Studies. In: *Energy Procedia* 16 (2012), S. 1052–1058

[35] Jovanović, Milan M.: Power Supply Technology – Past, Present, and Future. In: *PCIM China 2007, 6th International PCIM China Conference for Power Electronics, Conference Proceedings* 6 (2007)

[36] Kim, Jae H. ; Lee, Myung J.: *Green IT: Technologies and Applications*. Berlin, Heidelberg : Springer, 2011

[37] Koomey, Jonathan: *Growth in Data center electricity use 2005 to 2010*. Oakland, CA, USA : Analytics Press, 2011

[38] Krueger, Charles W.: Software Reuse. In: *ACM Computer Surveys* 24 (1992), Nr. 2, S. 131–183

[39] Lampe, Frank (Hrsg.): *Green-IT, Virtualisierung und Thin Clients*. Wiesbaden : Vieweg+Teubner, 2010

[40] Liang, S.-A.: *Low Cost and High Efficiency PC Power Supply Design to Meet 80 Plus Requirement*. Touyuan City, Taiwan : FSP Technology Inc. FSP Group, 2008

[41] Lim, S. H. ; Sharma, B. ; Tak, B. C. ; Das, C. R.: A dynamic energy management scheme for multi-tier data centers. In: *(IEEE ISPASS) IEEE International Symposium on Performance Analysis of Systems and Software*, 2011, S. 257–266

[42] Lintner, William ; Tschudi, Bill ; VanGeet, Otto: *Best Practices Guide for Energy-Efficient Data Center Design*. Washington, DC, USA : US Department of Energy, 2010

[43] McLachlan, Ross ; Brewster, Stephen: Towards new widgets to reduce PC power consumption. In: *CHI '12 Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA : ACM, 2012 (CHI EA '12), S. 2153–2158

[44] Michanan, Junya ; Dewri, Rinku ; Rutherford, Matthew J.: GreenC5: An adaptive, Energy-aware Collection for Green Software Development. In: *Sustainable Computing: Informatics and Systems* 13 (2017), S. 42–60

[45] Molla, Alemayehu ; Cooper, Vanessa A. ; Pittayachawan, Siddhi: IT and Eco-sustainability: Developing and Validating a Green IT Readiness Model. In: *ICIS 2009 Proceedings* (2009)

[46] Murugesan, San: Harnessing Green IT: Principles and Practices. In: *IT Professional* 10 (2008), S. 24–33

[47] Newcombe, Liam: *Data centre energy efficiency*. Swindon, UK : BCS, 2014 `http://policy.bcs.org/sites/policy.bcs.org/files/data-centre-energy.pdf___L.pdf`

[48] Noureddine, Adel ; Bourdon, Aurelien ; Rouvoy, Romain ; Seinturier, Lionel: *A Preliminary Study of the Impact of Software Engineering on GreenIT*. New York, NY, USA : IEEE, 2012

[49] Pang, Candy ; Hindle, Abram ; Adams, Bram ; Hassan, Ahmed E.: What Do Programmers Know about Software Energy Consumption? In: *IEEE Software* 33, Nr. 3 (2015), Juli, S. 83–89

[50] Pichl, Günter: Netzteile mit 80 Prozent Wirkungsgrad. In: *PC-Magazin* (2008)

[51] Pinto, Gustavo: Do Language Constructs for concurrent Execution have Impact on Energy Efficiency? In: *Proceedings of the 2013 Companion Publication for Conference on Systems, Programming & Applications: Software for Humanity*. New York, NY, USA : ACM, 2013 (SPLASH '13), S. 121–122

[52] Poess, Meikel ; Nambiar, Raghunath O.: Energy cost, the key challenge of today's data centers: a power consumption analysis of TPC-C results. In: *Proceedings of the VLDB Endowment* 1 (2008), Nr. 2, S. 1229–1240

[53] Prager, Manuel: *Laufzeitvergleiche für die Implementierung von Algorithmen in Java und C/C++*, Hochschule Neubrandenburg, Diplomarbeit, 2010

[54] Prechelt, Lutz: An Empirical Comparison of Seven Programming Languages. In: *Computer* 33, Nr. 10 (2000), S. 23–29

[55] Procaccianti, Giuseppe ; Fernández, Héctor ; Lago, Patricia: Empirical Evaluation of two Best Practices for Energy-efficient Software Development. In: *The Journal of Systems and Software* 117 (2016), S. 185–198

[56] Reiser, Martin: *The Oberon System. User Guide and Programmer's Manual*. New York, NY, USA : ACM Press, 1991

[57] Rong, Huigui ; Zhang, Haomin ; Xiao, Sheng ; Li, Canbing ; Hu, Chunhua: Optimizing Energy Consumption for Data Centers. In: *Renewable and Sustainable Energy Reviews* 58 (2016), S. 674–691

[58] Roy, Swapnoneel ; Rudra, Atri ; Verma, Akshat: An Energy Complexity Model for Algorithms. In: *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*. New York, NY, USA : ACM, 2013 (ITCS '13), S. 283–304

[59] Sabharwal, Manuj ; Agrawal, Abhishek ; Metri, Grace: Enabling Green IT through Energy-Aware Software. In: *IT Professional* 15, Nr. 1 (2012), S. 19–27

[60] Sokal, Nathan O. ; Sokal, Alan D.: Class E – A New Class of High-Efficiency Tuned Single-Ended Switching Power Amplifiers. In: *IEEE Journal of Solid-State Circuits* 10, Nr. 3 (1975), S. 168–176

[61] University of California, Berkeley (Hrsg.): *Auto Shutdown Manager*. Berkeley, CA, USA : EnviProt, 2010

[62] Wadhwa, Bharti ; Verma, Amandeep: Energy saving Approaches for Green Cloud Computing: A Review. In: *Recent Advances in Engineering and Computational Sciences (RAECS)*. New York, NY, USA : IEEE, 2014

[63] Woodside, Murray ; Franks, Greg ; Petriu, Dorina C.: *The Future of Software Performance Engineering*. New York, NY, USA : IEEE, 2007

[64] Yunus, S. ; Jailani, S. F. A. K. ; Hairuddin, H. ; Kassim, E. S.: Green IT adoption towards environmental sustainability: The moderating role of top management enforcement. In: *2013 International Conference on Research and Innovation in Information Systems (ICRIIS)*. New York, NY, USA : IEEE, 2013, S. 241–244

[65] Zaimi, Asimina ; Ampatzoglou, Apostolos ; Triantafyllidou, Noni ; Chatzigeorgiou, Alexander ; Mavridis, Androklis ; Chaikalis, Theodore ; Deligiannis, Ignatios ; Sfetsos, Panagiotis ; Stamelos, Ioannis: An Empirical Study on the Reuse of Third-Party Libraries in Open-Source Software Development. In: *Proceedings of the 7th Balkan Conference on Informatics Conference*. New York, NY, USA : ACM, 2015 (BCI '15). – ISBN 978–1–4503–3335–1, S. 4:1–4:8

[66] Zapico, Jorge L. ; Turpeinen, Marko ; Brandt, Nils: Climate Persuasive Services: Changing Behavior Towards Low-carbon Lifestyles. In: *Proceedings of the 4th International Conference on Persuasive Technology*. New York, NY, USA : ACM, 2009 (Persuasive '09), S. 14:1–14:8

[67] Zhang, Yifan ; Wang, Xudong ; Liu, Xuanzhe ; Liu, Yunxin ; Zhuang, Li ; Zhao, Feng: Towards Better CPU Power Management on Multicore Smartphones. In: *Proceedings of the Workshop on Power-Aware Computing and Systems*. New York, NY, USA : ACM, 2013 (HotPower '13), S. 11:1–11:5

[68] Zheng, Hongzhong ; Lin, Jiang ; Zhang, Zhao ; Gorbatov, Eugene ; David, Howard ; Zhu, Zhichun: Mini-rank: Adaptive DRAM architecture for improving memory power efficiency. In: *Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture*. Washington, DC, USA : IEEE Computer Society, 2008 (MICRO 41), S. 210–221