Bachelorarbeit

Markus Jansen

Energieeffiziente Betriebssysteme am Beispiel von Linux

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Markus Jansen

Energieeffiziente Betriebssysteme am Beispiel von Linux

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Bernd Kahlbrandt
Zweitgutachter : Prof. Dr. Gunther Klemke

Abgegeben am 19. September 2009

Markus Jansen

Thema der Bachelorarbeit

Energieeffiziente Betriebssysteme am Beispiel von Linux

Stichworte

Energieverwaltung, Betriebssysteme, Linux, ACPI, OSPM, DPM, DVS, Betriebssystem-gesteuerte Energieverwaltung, Dynamic Power Management, Dynamic Voltage Scaling

Kurzzusammenfassung

In der Informatik gewinnen energiebewusste Technologien immer mehr an Gewicht. Diese Arbeit zeigt, wie die Energieeffizienz von typischen Desktop-PCs mit Linux-Betriebssystemen gesteigert werden kann. Die Optimierungen betreffen die Betriebssystemebene und zielen vor allem auf den Bereich der CPU, der Festplatte und der Interrupts, sowie der Erstellung eines, hinsichtlich des Speicherbedarfs und der installierten Software, möglichst minimalen Systems ab. Dabei wird der Energieverbrauch reduziert, ohne die Systemleistung in einem Maß zu verschlechtern, das den Benutzer bei der Erledigung alltäglicher Aufgaben negativ beeinflusst. Je nach Verwendungszweck des Computers wird dabei der Energieverbrauch zwischen ca. 4% und ca. 8% reduziert.

Markus Jansen

Title of the paper

Energie-efficient Operating Systems using the example of Linux

Keywords

Power Management, Operating Systems, Linux, ACPI, OSPM, DPM, DVS, Operating System Directed Power Management, Dynamic Power Management, Dynamic Voltage Scaling

Abstract

Energy-Awareness is getting more and more important in Information- and Communication-Technology. This paper demonstrates how energy efficience of typical Desktop-PCs using linux operating systems can be improved. The optimization concerns the operating system layer and are targeting especially CPU-, hard disk- and interrups-issues as well as minimizing the sytem concerning memory requirements and the software installed. In doing so, the energy consumption will be reduced without affecting the overall system performance in a manner, that constraints users to do their routine tasks. According to the sytem usage the energy consumption can be reduced between about 4% and 8%.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einleitung

1.1 Motivation

Der erste rein elektronische Universalrechner ENIAC wurde 1946 der Öffentlichkeit vorgestellt. Er hatte 17.000 Röhren und verbrauchte 174 kW Strom [1]. Mit der Einführung der Transistoren ging der Stromverbrauch erheblich zurück, so dass der Energieverbrauch von Computern lange Zeit keine große Rolle spielte [2]. Mit der stärkeren Verbreitung von mobilen Geräten, die auf eine lange Akkulaufzeit angewiesen sind, rückten stromsparende Technologien wieder mehr in den Fokus der Computerindustrie, da trotz umfangreicher Forschung die Laufzeiten von Batterien nicht in dem Maße erhöht werden konnten, wie es nötig gewesen wäre, um dem höheren Verbrauch gerecht zu werden [2]. Durch steigende Energiepreise und gleichzeitig wachsendem Energiebedarf hat das Thema auch bei Desktop-PCs und Servern an Relevanz gewonnen. Insbesondere für Rechenzentren ist der Energieverbrauch ein großer Kostenfaktor. Ein weiterer wichtiger Faktor ist die Wärmeentwicklung der Hardware, die maßgeblich von deren Stromverbrauch abhängt.

Gegenwärtig werden auch Forderungen nach einem verantwortungsbewussten Umgang mit unserer Umwelt in der gesellschaftlichen Diskussion immer präsenter. Angesichts des Klimawandels, knapper werdender natürlicher Ressourcen, höherer Emissionswerte und des gleichzeitig ständig wachsendem Bedarfs an Informations- und Kommunikationstechnologien in sämtlichen Lebensbereichen hat dieses Thema auch die IT-Branche erreicht. In der öffentlichen Diskussion liegt der Fokus derzeit insbesondere auf Rechenzentren. 2006 verbrauchten laut Lewis Curtis, einem Strategie-Infrastruktur-Architekt von Microsoft, die ca. 6000 Rechenzentren in den USA 61 Billionen kWh. Das sind laut dem US-Amerikanischen „Department of Energie“ (DOE) 1,5 % des US-Gesamtverbrauch an Elektrizität, der jährlich um 12 % steigt [3]. Einer Studie des Marktforschungsunternehmens Gartner zufolge werden 2% der CO2-Emissionen weltweit durch Informations- und Kommunikationstechnologien verursacht.

Längst ist „Green IT“ auch zu einem Label für viele Unternehmen und Organisationen geworden. Für diese ergeben sich eine Reihe wirtschaftlicher Gründe für den Einsatz von energiesparenden und somit kostensparenden Technologien. Ebenso kann aus Imagegründen eine ökologische Ausrichtung des Unternehmens wichtig sein. Auch die Einhaltung von gesetzlichen Richtlinien, wie es sie derzeit schon vereinzelt im Bereich der Entsorgung gibt, könnte in Zukunft weiter an Relevanz gewinnen [4]. Doch nicht für nur Unternehmen und staatliche Einrichtungen ist Green IT von Bedeutung. Die Endverbraucher können davon ebenso in ideeller sowie wirtschaftlicher Hinsicht profitieren und ihre Bedürfnisse an die Industrie herantragen.

1.2 Problemstellung und Zielsetzung

In der Vergangenheit wurden Betriebssysteme in erster Linie mit dem Schwerpunkt auf Rechenleistung oder, insbesondere in den letzten Jahren der rapiden Entwicklung des Internet Rechnung tragend, in Bezug auf Sicherheit und Bedienkomfort entwickelt. Dabei steht die Forderung nach immer mehr Leistung im direkten Gegensatz zu dem Anspruch, möglichst wenig Strom zu verbrauchen.

Nachdem lange Zeit die Energieeffizienz im Wesentlichen in der Verantwortung der Hardwarehersteller lag, wuchsen seit Beginn der 1990er Jahre mit APM und später vor allem mit ACPI die Einflussmöglichkeiten des Betriebssystems auf den Energieverbrauch eines Computers. Mit der Zeit gewannen energieeffiziente Technologien an Gewicht beim Entwurf und der Implementierung von Betriebssystemen. Dieser Trend hält bis heute an.

Diese Arbeit befasst sich mit der Fragestellung, welchen Einfluss das Betriebssystem auf die Energieeffizienz eines Computers nehmen kann und inwiefern auf der Ebene des Betriebssystemkerns der Stromverbrauch gesenkt werden kann. Der Begriff Energieeffizienz bedeutet in diesem

Zusammenhang die Reduzierung des Stromverbrauchs, ohne gleichzeitig die Leistung unverhältnismäßig stark einzuschränken.

Ziel dieser Arbeit ist, Strategien für typische Benutzerprofile zu entwerfen, die den Energieverbrauch eines Linux-Systems reduzieren. Die Strategien sollen dabei nicht auf minimalen Stromverbrauch ausgelegt sein, sondern sinnvoll zwischen Leistung, Bedienkomfort und Energiebedarf balanciert sein und für den Benutzer in dem Sinne transparent sein, dass er die verwendeten Strategien nicht wahrnimmt.

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich in fünf aufeinander aufbauende Teile.

In der Einleitung wird die Motivation, die dieser Arbeit zu Grunde liegt, erläutert sowie der inhaltliche Rahmen eingegrenzt.

Die Grundlagen bilden das Fundament zum Verständnis der Arbeit.

Der dritte Teil befasst sich mit der Analyse. Dabei werden zunächst die Hardwarekomponenten und Peripheriegeräte eines Computers identifiziert, die den größten Energieverbrauch haben. Davon ausgehend wird untersucht, welche Technologien, Algorithmen und Strategien auf Betriebssystemebene angewandt werden, um den Energiebedarf eines Computers zu senken. Darauf aufbauend wird anschließend ein Linux-Kernel dahingehend untersucht, welche Technologien zur Steigerung der Energieeffizienz dort implementiert sind

Nach Abschluss der Analyse werden in der Synthese Strategien entwickelt, die den Stromverbrauch eines Linux-Systems reduzieren sollen. Dazu müssen verschiedene Szenarien berücksichtigt werden, da die Art der Nutzung nicht nur direkte Auswirkungen auf den Stromverbrauch hat, sondern auch unterschiedliche Möglichkeiten bieten und Grenzen aufzeigen. An Hand einer geeigneten Testumgebung wird der Stromverbrauch der jeweiligen Szenarien gemessen und analysiert. Auf Grundlage der Testdaten werden Strategien zur Optimierung auf Systemebene entworfen und diese anschließend evaluiert.

Den Schluss bildet ein kritischer Rückblick auf die Arbeit und fasst die wesentlichen Erkenntnisse zusammen, um abschließend einen Ausblick auf die Zukunft von Betriebssystemen hinsichtlich dessen Energieeffizienz zu bieten.

1.4 Einordnung

Nachdem sich zunächst die Mehrzahl der Forschungsarbeiten im Bereich „Green IT“ auf die Reduzierung des Energieverbrauchs von Hardwarekomponenten auf Schaltkreis-Ebene befasste, verschob sich später der Fokus auf das Power Management einzelner Komponenten, insbesondere das der CPU, der Festplatte, des Monitors und des Speichers, wo besonders die Verwendung in Embedded- und Echtzeit-Systemen im Vordergrund stand. Mit der der größeren Verbreitung von Notebooks, deren Nutzen auch stark von der Akkulaufzeit abhängt, gewinnt ein generisches Power Management auf Software- und Betriebssystemebene mehr Einfluss, dass nicht für genau eine bestimmte Hardware vorgesehen ist. Diese Arbeit knüpft daran an und untersucht den Einfluss des Betriebssystems auf die Energieeffizienz von Desktop-PCs, die in erster Linie nicht auf Energiesparsamkeit ausgerichtet sind.

1.5 Abgrenzung

Neben einem allgemeinen Teil, der sich grundsätzlich mit der Energieeffizienz von Betriebssystemen auseinandersetzt, wird in einem weiteren Teil die Implementierung des Betriebssystemkerns eines Linux-Systems in Augenschein genommen. Gegenstand der Untersuchungen ist dabei der Linux Kernel 2.6. Anwendungsprogramme und andere Betriebssysteme werden nicht berücksichtigt.

Als Hardwareplattform werden typische Desktop-PCs verwendet, wie sie beispielsweise an einem Büro-Arbeitsplatz Verwendung finden. Embedded- oder Echtzeit-Systeme sowie spezielle Hardware[1] werden nicht berücksichtigt.

Weiterführenden Themen, welche die Architektur von IT-Systemen hinsichtlich deren Energieeffizienz in den Fokus rücken, kann diese Arbeit möglicherweise als Anknüpfungspunkt dienen. In dieser Arbeit spielt die Architektur aber keine Rolle, da die Energieeffizienz von Betriebssystemen an Hand von einzelnen und voneinander unabhängigen Computern betrachtet wird.

[1] Wie z.B. herstellerspezifische Energiesparfunktionen im BIOS

2. Grundlagen

2.1 Definitionen / verwendete Begriffe

2.1.1 Stochastischer Prozess

Ein stochastischer Prozess beschreibt die Zustände eines zufallsbeinflussten Systems in aufeinander folgenden Zeitpunkten. Ein stochastischer Prozess ist demnach eine zeitliche Abfolge von Zufallsvariablen, z.B. $X_0, X_1, X_2, \ldots$.

Definition laut [5]: „Für einen stochastischen Prozess benötigt man einen Wahrscheinlichkeitsraum $(\Omega, \mathcal{A}, P)$, einen Zustandsraum Ω' , auch Zustandsmenge genannt, einen Bildbereich T , meist mit $T \in R$ und für jeden Zeitpunkt $t \in T$ eine Zufallsvariable $X_t : \Omega \to \Omega'$, die den Zustand zum Zeitpunkt t angibt. Dann heißt $(X_t) := (X_t, t \in T)$ ein stochastischer Prozess“.

2.1.2 Markov-Kette

Eine Markov-Kette mit der Ordnung m ist ein stochastischer Prozess, speziell die Folge von Beobachtungen (bzw. der Koordinaten-Variablen) $X_0, X_1, X_2, \ldots$ in einem unendlich-stufigen Versuch mit abzählbarer Zustandsmenge I , bei dem die zukünftige Entwicklung des Systems nur von den letzten m beobachteten Werten abhängt. Üblicherweise spricht man von einer Markov-Kette, wenn es sich um eine Markov-Kette der Ordnung 1 handelt, also der weitere Verlauf nur vom aktuellen Zustand abhängt, weshalb sich die Modellierung als Übergangsgraph anbietet [6].

2.1.3 Energieeffizienz

Unter Energieeffizienz wird in dieser Arbeit die Optimierung eines System auf einen möglichst geringen Energieverbrauch unter Berücksichtigung der Systemperformanz, die nicht unter ein Niveau fallen darf, unter dem typische Aufgaben nicht in akzeptabler Zeit erledigt werden können und unter dem insbesondere die Bedienbarkeit des System gewährleistet bleibt.

2.2 CMOS

CMOS ist die seit 25 Jahren führende Halbleitertechnologie für IC- und Chipdesign [7]. Deren Verwendung in Mikroprozessoren und Speicherbausteinen macht sie zu einem wichtigen Faktor für den Energieverbrauch eines Computers. Eine CMOS-Schaltung zeichnet sich unter anderem dadurch aus, dass sie fast keinen statischen Stromverbrauch hat. Die Verlustleistung kann ebenso vernachlässigt werden, so dass der ausschlaggebende Faktor der Kurzschlussstrom ist, der beim Umschalten, also dem Umladen der Gatter-Kapazität, entsteht. Somit ist der durchschnittliche Energieverbrauch eines einzelnen Gatters durch

$$P = \alpha C_G V_{dd}^2 f$$

gegeben, wobei α die Schalthäufigkeit[2], C_G die Kapazität des Gatters, V_{dd} die Betriebsspannung (üblicherweise 3,3 oder 5 Volt) und f die Taktfrequenz ist [8]. (Die Kapazität der Leitungen innerhalb der Schaltung wirkt sich zwar auch auf den Stromverbrauch aus, kann aber an dieser Stelle vernachlässigt werden, weil deren Auswirkungen sehr gering sind.) Daraus ergeben sich nach [8] drei grundsätzliche Ansatzpunkte zur Steigerung der Energieeffizienz, die im Folgenden kurz erläutert werden:

1. *Activity-Based System Shutdown* setzt die Taktrate ($f = 0$) oder Betriebsspannung ($V_{dd} = 0$) auf 0, wenn die Schaltung momentan nicht verwendet wird.

[2] Häufigkeit eines 0 → 1 Übergangs pro Taktzyklus

2. *Supply-Voltage Reduction* senkt die Betriebsspannung so weit wie möglich.
3. *Switching Activity Reduction* reduziert die Umschalthäufigkeit. Im Kapitel 3 dieser Arbeit werden diese Punkte aus der Betriebssystemperspektive genauer betrachtet werden.

3. Analyse

3.1 Hauptverbraucher eines PC-Systems

Es gibt unterschiedliche Angaben darüber, wie groß der Anteil bestimmter Hardware-Komponenten für den gesamten Energieverbrauch eines PC-Systems ist. Insgesamt ist festzustellen, dass neben dem Display die CPU und die Festplatte als die Komponenten mit dem größten Energiebedarf gelten [9] [10] [11].

Gerade in den letzten Jahren sind auch Grafikkarten immer leistungsfähiger geworden, weshalb davon auszugehen ist, dass diese einen zunehmend gewichtigen Faktor darstellen. Das Gleiche gilt für drahtlose Kommunikation, deren Einfluss gerade in mobilen Geräten in den letzten Jahren zugenommen hat [12].

3.2 Typische Betriebssystemtechnologien

In diesem Kapitel werden Technologien und Strategien vorgestellt, mit denen auf Systemebene eine höhere Energieeffizienz erreicht werden kann. Betriebssystem-gesteuertes Power Management („Operating System Directed Power Management“, OSPM) kann grundsätzlich in zwei Kategorien aufgeteilt werden, das „Dynamic Power Management“ (DPM) und „Dynamic Voltage Scaling“ (DVS) [13]. Das DPM versucht die Zeit, in der Hardware-Komponenten nicht benötigt werden, auszunutzen, in dem sie diese nach bestimmten Regeln in einen energieeffizienteren Zustand versetzen. Die Aufgabe des DVS ist es, die Frequenz und Spannung eines Prozessors an die Auslastung des Systems anzupassen, um unter Berücksichtigung der Leistungsanforderungen möglichst wenig Energie zu verbrauchen.

Anschließend werden die Möglichkeiten eines Betriebssystems aufgezeigt, die Energieeffizienz einzelner Hardware-Komponenten zu erhöhen, um danach eine Blick auf die strukturellen Aspekte von energieeffizienten Betriebssystemen werfen. Zuletzt wird in diesem Kapitel auf APM und vor allem ACPI eingegangen, einen offenen Industriestandard, der von den meisten Betriebssystemen zur Steuerung und Konfiguration ACPI-fähiger Komponenten in Computersystemen eingesetzt wird.

3.2.1 DPM

DPM nutzt den Leerlauf von Hardwarekomponenten zur Reduzierung des Energiebedarfs [14]. Da PC-Systeme in hohem Maße Ereignis-gesteuert sind, verbringen deren einzelne Komponenten viel Zeit im Leerlauf, während sie auf Aktionen des Benutzers warten. Dadurch verbrauchen die Geräte mehr Energie als für den Betrieb notwendig ist, da sie bei Inaktivität eigentlich abgeschaltet werden könnten [8]. Dem DPM liegt nun die Idee zugrunde, momentan ungenutzte Komponenten in einen Zustand zu versetzten, in dem sie weniger oder gar keinen Strom verbrauchen [15]. Das DPM richtet sich also nach der Auslastung einzelner Geräte eines Systems.

3.2.1.1 Power-Management System (PMS)

Ein Power-Management-System besteht aus einem Power Manager (PM) und geeigneten Hardware-Komponenten. Geeignete Komponenten sind in dem Falle solche, die neben dem aktiven Zustand mindestens einen weiteren Schlafzustand mit geringerem Energiebedarf unterstützen. Das Betriebssystem als Schnittstelle zwischen Hardware und Software eignet sich gut für die Implementierung des PM. Dieser besteht aus drei Komponenten (siehe Abbildung 1). Der Observer überwacht den Systemstatus und sammelt Informationen über das Verhalten des Systems. Auf Basis dieser Informationen werden Policies erstellt oder verändert, die den implementierten Algorithmus darstellen [16]. Der Controller wählt eine geeignete Policy aus und weist die Hardware an, ihren System-Status zu ändern. Die Pfeile geben den Kontrollfluss an.

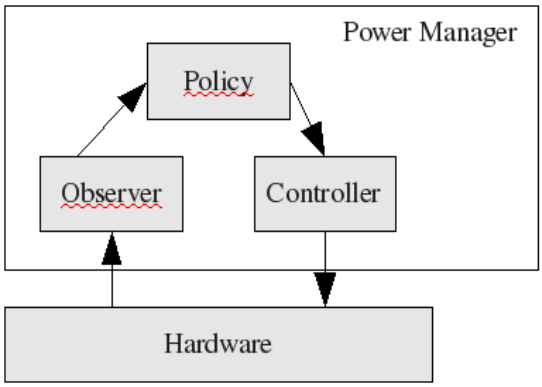

Abbildung 1: Power-Management System

3.2.1.2 Power Manageable Components (PMC)

Ein PMS setzt voraus, dass die zu steuernde Hardware mindestens einen Schlafzustand annehmen kann. Ein Schlafzustand wird im Allgemeinen durch den Kompromiss zwischen Leistung und Energiesparsamkeit charakterisiert, wobei Leistung in diesem Fall die Kürze der Verzögerungsdauer, die für die Wiederaufnahme des Arbeitszustands notwendig ist, darstellt. Der Arbeitszustand, auch aktiver Zustand oder „working state“ genannt, bringt volle Leistung bei hohem Stromverbrauch, der je nach Gerät konstant oder abhängig von aktuell erbrachten Leistung ist. Je mehr verschiedene Zustände unterstützt werden, desto besser kann auf die aktuelle Systemauslastung reagiert werden, sofern die verwendeten Power Management Strategien dies unterstützten. Typische Zustände sind laut [2]:

1. „An“ („*On*“), volle Leistung bei vollem Energieverbrauch.
2. „Schlafen“ („*Sleep*“), das Gerät wird derzeit nicht benötigt, befindet sich aber in Bereitschaft und verbraucht weniger Strom.
3. „Ruhezustand“, („*Hibernate*“) das Gerät ist aus, kann aber seinen Arbeitskontext innerhalb einer gewissen Zeit rekonstruieren.
4. „Aus“ („*Off*“), das Gerät ist aus und muss zur nächsten Verwendung neu gestartet werden.

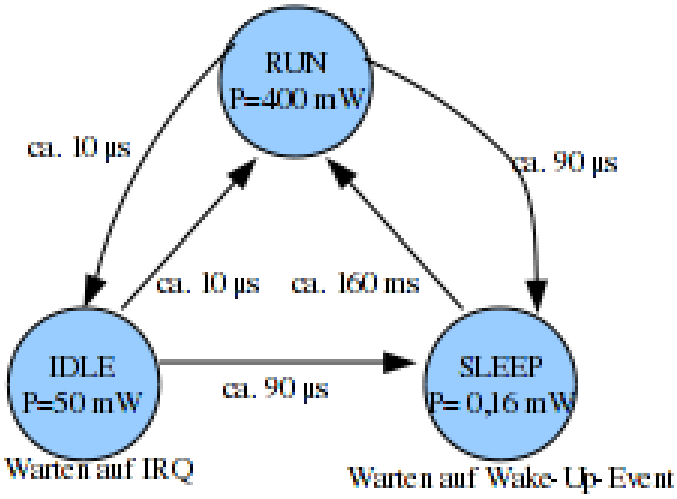

Abbildung 2: Zustandsautomat einer PMC

Dabei ist zu beachten, dass ein Zustandswechsel Overhead verursacht. Zu der bereits erwähnten Verzögerung, die bei einem Zustandswechsel auftritt, können Zustandswechsel auch kurzzeitig zusätzliche Energie verbrauchen. Beide Effekte treten beispielsweise deutlich beim Hochfahren einer Festplatte auf, wenn diese aus dem Ruhezustand erwacht und der Motor, der die Platten bewegt, wieder gestartet werden muss.

PMC können wie in Abbildung 2 an Hand eines endlichen Automaten modelliert werden, in dem die Zustände die verschiedenen Hardware-Zustände („An“, „Schlafen“, …) darstellen und die Übergänge die durch Verzögerung und zusätzlichem Energieverbrauch bei den Zustandsüberführungen

entstehenden Kosten kennzeichnen [17]. Die interne Funktionsweise der Hardware wird aus der Perspektive des Betriebssystems durch eine einheitliche Schnittstelle wie ACPI abstrahiert.

3.2.1.3 Observer

Bei PM-Komponenten kann in intern und extern verwaltet unterschieden werden. Erstere werden auch selbstverwaltete Geräte genannt, da sie selbstständig und ohne Kenntnis des Gesamtsystems und dessen Auslastung ihre Zustände wechseln [17].

Gegenstand dieser Arbeit sind jedoch Komponenten, die extern durch das Betriebssystem verwaltet werden und somit auch Teil des DPM sind. Die Funktion des Observers eines PMS ist das Sammeln von Informationen über alle PMCs wie:

1. Verteilung der Zwischenzeiten zwischen Anfragen an die Ressource und
2. die Verteilung der aus den Anfragen resultierenden Betriebszeiten [17].

In ACPI-kompatiblen Systemen wie Microsoft Windows oder Linux (ab Kernel-Version 2.4) können diese Informationen über ACPI zugänglich sein.

3.2.1.4 Controller

In einem Power Management System ist der Controller für die Steuerung der Hardware zuständig. Abhängig von der Entscheidung der implementierten Policy versetzt er die Hardware in der Policy angeforderten Zustand.

3.2.1.5 Policy

Eine Policy stellt einen Algorithmus dar, der unter Berücksichtigung der vom Observer gelieferten Informationen entscheidet, zu welchem Zeitpunkt und in welchen Zustand (sofern mehrere Schlafzustände vorhanden sind) der Wechsel erfolgt. Würde der Zustandswechsel keine zusätzliche Kosten verursachen, wäre das Finden einer idealen Strategie trivial; sobald ein Gerät nicht mehr benötigt wird, wird es in den tiefstmöglichen Schlafzustand versetzt. Um eine Komponente wieder in den Arbeitszustand zu versetzen, muss allerdings :

1. die Stromversorgung und der Taktgeber eingeschaltet und stabilisiert werden
2. das System reinitialisiert
3. der Arbeitskontext wiederhergestellt werden, was insgesamt möglicherweise eine lange Zeit in Anspruch nehmen kann [17].

Offensichtlich können sich diese Verzögerungen auch negativ auf die Akzeptanz der Anwender auswirken, so dass eine geeignete Strategie auch das Benutzerverhalten berücksichtigen muss, das typischerweise unstet und zumindest zu Beginn einer Benutzersitzung nicht vorhersehbar ist.

Policies werden unterschieden nach prädiktiven, also voraussagenden, und stochastischen Strategien [14] [17] [16]. Als zusätzliche Klassifizierung finden sich Timeout-Policies in [15] [18].

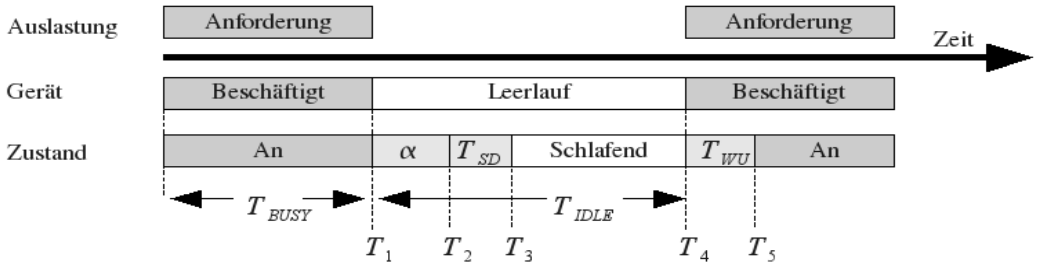

Abbildung 3: Zeitlicher Verlauf beim Statuswechsel aus unterschiedlichen Perspektiven

T_1	Zeitpunkt, ab dem das Gerät nicht mehr beschäftigt ist
T_2	Zeitpunkt, an dem Timeout-Zeit abgelaufen
T_3	Zeitpunkt, an dem Wechsel in den Schlafzustand abgeschlossen ist
T_4	Zeitpunkt, an dem Schlafzustand beendet ist
T_5	Zeitpunkt, nach der Wechsel in den aktiven Zustand abgeschlossen ist
$T_{SD}=T_3-T_2$	Dauer der Überführung in den Schlafzustand
$T_{WU}=T_5-T_4$	Dauer der Überführung in den aktiven Zustand, Verzögerung
$\alpha=T_2-T_1$	Timeout Zeit
$T_{IDLE}=T_4-T_1$	Dauer des Schlafzustands
T_{BUSY}	Zeitraum, in dem das Gerät beschäftigt ist
$T_0=T_{SD}+T_{WU}$	Gesamtverzögerung
$T_{MS}=T_{BE}-T_0$	Minimale Länge des Schlafzustands
T_{BE}	Break-even Zeit
P_W	Stromverbrauch während des Leerlaufs
P_S	Stromverbrauch während des Schafzustands
t_{idle}	Vorausgesagte Leerlaufzeit

Tabelle 1: Policy-Variablen

Timeout Policy

Timeout Policies sind die einfachsten Strategien. Der zeitliche Verlauf eine Statuswechsel wird in Abbildung 2 dargestellt. Ist ein Gerät für eine Zeit $\alpha=T_2-T_1$ (Timeout) unbenutzt, so wird angenommen, dass es sich lohnt, in den nächsten, energiesparenderen Zustand zu wechseln und dieser Zustand mindestens für eine Zeit T_{BE} [3] beibehalten werden kann [19]. T_{BE} gibt also die benötigte Mindestdauer an, die benötigt wird, um den durch den Statuswechsel verursachten zusätzlichen Energieverbrauch zu kompensieren [17], also $P_W*T_{BE}=E_0+P_S*T_{MS}$, mit P_W als den Stromverbrauch während des Leerlaufs, P_S dem Stromverbrauch im Schlafzustand des Gerätes, E_0 als die durch den Zustandswechsel zusätzlich benötigte Energie und T_{MS} als die minimale Zeit in Schlafmodus mit $T_{MS}=T_{BE}-T_0$, wobei T_0 die Verzögerung ($T_{SD}+T_{WU}$) ist. Daraus ergibt sich ebenso $T_{BE}=(E_0-P_S*T_0)/(P_W-P_S)$.

Der Schwachpunkt von Timeout-Policies liegt zum einen darin, dass während des Ruhezustands das Gerät mehr Energie verbraucht als nötig und zum anderen werden kurze Leerlaufzeiten nicht berücksichtigt. Da die Effizienz im Wesentlichen durch den Grenzwert α bestimmt wird, kann die Energieeffizienz durch adaptives Anpassen dieses Wertes erhöht werden (Adaptive Timeout Policy, ATO). Die von [20] vorgestellte adaptive Strategie erhöht α bei einem kleinen Wert für $T_1[i-1]/T_{WU}$ und verringert α bei einem geringen Wert. Ein anderer Ansatz ist, den Grenzwert nicht um einen festen Werte zu reduzieren, sondern diesen zu halbieren [19]. [21] geht davon aus, dass auf lange Betriebszeiten lange Leerlaufzeiten folgen und erhöht folglich α bei großem T_{BUSY} sowie den Grenzwert bei kleinem T_{BUSY} verringert wird.

[3] Break-even-time

Prädiktive Policy

Prädiktive Strategien versuchen mittels heuristischer Methoden die Leerlauf-Zeit T_{IDLE} einer PMC vorherzusagen. Sofern der vorausgesagte Wert t_{IDLE} einen Zustandswechsel rechtfertigt, also $t_{IDLE} > T_{BE}$, wird das Gerät in den nächst-energiesparenderen Zustand versetzt [22]. Ziel ist es, den Timeout Wert α zu eliminieren.

Srivastava et. al. verfolgt den Ansatz, an Hand vorangegangener Werte $T_{IDLE}[1]..T_{IDLE}[i-1]$ für die Leerlaufzeiten und $T_{BUSY}[1]..T_{BUSY}[i-1]$ für die den Leerlaufzeiten vorangegangenen Betriebszeiten eine Annahme für den Wert $T_{BUSY}[i]$ zu machen. Eine andere von Srivastava et. al. entwickelte Strategie namens „L-Shape“ besagt, dass lange Leerlaufzeiten oft auf kurze Betriebszeiten folgen und kurze Leerlaufzeiten oft auf lange Betriebszeiten folgen. In einem zweidimensionalen Diagramm zueinander in Relation gesetzt ergibt dies L-Form (engl. „L-Shape“), weshalb eine PMC nach kurzen Betriebszeiten in den Schlafzustand versetzt wird [8].

Hwang und Wu treffen die Vorhersage mit Hilfe der vorangegangenen vorhergesagten und tatsächlichen Leerlauf-Zeit, formal $t_{IDLE} = \alpha * T_{IDLE}[i-1] + (1-\alpha) * t_{IDLE}[i-1]$ [23]. Diese Methode, Exponential Average (EA), genannt, liefert den Durchschnitt vorangegangener Leerlaufzeiten mit exponentieller Gewichtung, wodurch die Dominanz von außergewöhnlich hohen Leerlaufzeiten verhindert wird.

Chung et. al. stellt eine Methode vor, die mehrere Schlafzustände unterstützt [16]. Vor Beginn der Leerlaufzeit wird an Hand einer geschätzten Leerlaufdauer der geeignetste Zustand des Gerätes mittels einer Datenstruktur namens „Adaptive Learning Tree“ ermittelt. Die Leerlaufzeit wird unter der Annahme, dass sich aus der Länge der aufeinander folgenden Auslastungs- und Leerlaufintervalle wiederkehrende Muster ergeben, geschätzt.

Ein von Gniady et al. ebenfalls unter Verwendung maschinellen Lernens verfolgter Ansatz ist der Program-counter Access Predictor (PCAP) [24]. Dieser entscheidet an Hand erlernter Muster, ob es sich lohnt ein Gerät abzuschalten. Die Muster modellieren die Zugriffe von Anwendungen auf PMCs.

Chung et al. benennt drei Nachteile, die beinahe alle prädiktiven Policies gemeinsam haben. Erstens berücksichtigen sie fast ausnahmslos lediglich einen Schlaf-Zustand. Zweitens kann der Kompromiss zwischen Energiebedarf und durch Verzögerungen entstandene Leistungseinbußen nicht genau genug ausgehandelt werden und drittens berücksichtigen sie keine Warteschlange für eintreffende Anfragen, die gerade nicht bearbeitet werden können [18].

Stochastische Verfahren

Schreibweisen:

- Mengen $\mathcal{M}$: kalligraphisch
- Vektoren $\boldsymbol{v}$: Kleinbuchstaben, fett
- Matrizen $\boldsymbol{M}$: Großbuchstaben, fett
- Skalare Konstanten S : Großbuchstaben, kursiv
- Skalare Variablen x : Kleinbuchstaben, kursiv

Um die Funktionsweise von stochastischen Verfahren zu erläutern wird zunächst das von Benini et al. entworfenen Discrete-Time Markov Model (DM) [25] vorgestellt und darauf folgend kurz auf einige Erweiterungen eingegangen.

Um den Nachteilen von prädiktiven Strategien entgegen zu wirken, wurde eine weitere Kategorie von Policies entwickelt. Stochastische Verfahren formulieren „Policy Optimization“ als stochastisches Optimierungsproblem. Das Ziel ist, aus allen möglichen Policies diejenige herauszufinden, die entweder für ein gegebenes Performance-Level den niedrigsten Energieverbrauch hat oder für einen

gegebenen Energieverbrauch die höchste Leistung bringt [25]. Leistung und Energieverbrauch sind also die Kostenmetriken.

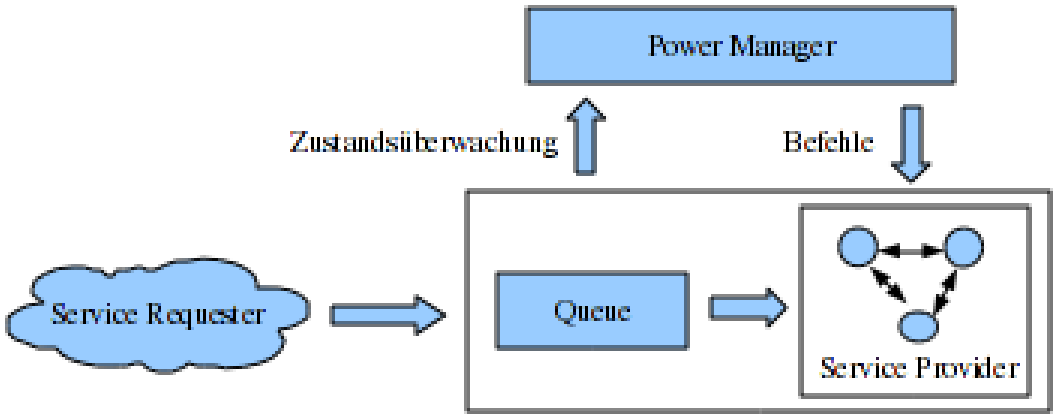

Abbildung 4: Stochastisches Modell

Abbildung 4 zeigt das zu Grunde liegende stochastische Modell. Der Power Manager überwacht den Service Provider (SP), den Service Requester (SR) und die Warteschlange (Service Queue). Der SR kann beispielsweise eine Festplatte sein. In der Queue werden Anfragen gepuffert, bevor sie durch den SP verarbeitet werden. Jede dieser drei Komponenten wird mittels einer Markov-Kette modelliert, die auch als Zustandsdiagramm beschrieben werden kann. Das Charakteristische einer Markov-Kette ist, dass die Zukunft des Systems nur vom aktuellen Zustand abhängt [25].

Service Provider

Der Service Provider (SP) wird formal durch das Tripel $(\mathcal{M}_{SP}(a), b(s,a), c(s,a))$ dargestellt. Die Menge $\mathcal{M}_{SP}(a)$ ist eine stationäre Markov-Kette mit $a \in \mathcal{A}$ als Befehl, den der PM an den SP richtet (bspw.: "s_on", "s_off") und durch den dann mit einer bestimmten Wahrscheinlichkeit der Zustand gewechselt bzw. beibehalten wird sowie den Zuständen $\mathcal{S} = \{s_i | 1, 2, \ldots, S\}$ und der stochastischen Matrix $P^{SP}(a)$. Jeder Kombination aus dem Zustand $s \in \mathcal{S}$ und a wird eine Service Rate $b(s,a)$ als Kostenmetrik mit $b: \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$ und ein Energieverbrauch $c(s,a)$ mit $c: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ zugeordnet. Die Zustände $\mathcal{S}$ sind die Energielevel-Zustände des Gerätes, im einfachsten Fall „An" und „Aus" [25].

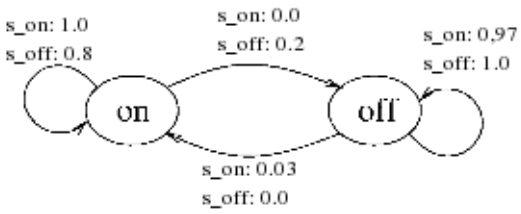

Abbildung 5: Markov-Ketten-Modell des Service Provider

Service Requester

Der Service Requester (SR) modelliert die Systemumgebung und ist durch das Paar $(\mathcal{M}_{\mathcal{R}}, z(r))$ definiert. Der Markov-Prozess $\mathcal{M}_{\mathcal{R}}$ hat die Zustandsmengen $\mathcal{R} = \{r_i | 0, 1, \ldots, (R-1)\}$ sowie die stochastische Matrix P^{SR}. $z(r)$ ist die Funktion $z: R \rightarrow \mathbb{N}$, die die die Anzahl der Anfragen pro Zeitscheibe für jeden Zustand angibt. Die Zustände $\mathcal{R}$ geben die Anzahl der Anfragen, die pro Zeitscheibe auftreten [25].

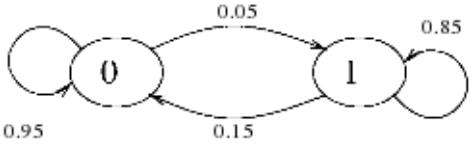

Abbildung 6: Markov-Ketten-Modell des Service Requesters

Service Queue

Die Service Queue (SQ) $\mathcal{M}_{\mathcal{SQ}}(a,s,r)$ mit den Zuständen $Q=\{q_i|0,1,\ldots,(Q-1)\}$, Übergangsparameter der Menge $\mathcal{A}\times\mathcal{S}\times\mathcal{R}$ und der stochastischen Matrix $P^{SQ}(a,s,r)$ modelliert die Warteschlange, in welcher die Anfragen an eine Ressource zunächst eingereiht werden. Die Zustände Q repräsentieren die Anzahl der Anfragen, die noch verarbeitet werden müssen.

Der Power Manager PM schickt in jeder Zeiteinheit t_n einen Befehl $a\in\mathcal{A}$ an den Service Provider und kontrolliert damit die Übergänge zwischen den Zuständen. Wie bereits erwähnt, beobachtet der PM die Komponenten des PMS und speichert diese Informationen als Systemhistorie $\mathcal{H}_n\in(\mathcal{S}\times\mathcal{R}\times Q)^n$, die alle mögliche Sequenzen des n-Tupels (s,q,r) enthält und als Entscheidungsgrundlage für den zu initiierenden Befehl $a\in\mathcal{A}$ dient. Das Element $H_n\in\mathcal{H}$ repräsentiert dabei die Historie für n Zeitscheiben [25].

Das ganze System kann als eine Komposition aus den drei Markov-Ketten des SP, SR und der SQ aufgefasst werden und mit der Zustandsmenge $\mathcal{X}=\mathcal{S}\times Q\times\mathcal{R}$ und der stochastischen Matrix $P(a)$ definiert werden. Die Übergangswahrscheinlichkeit vom Zustand x_i nach x_j abhängig vom Befehl a ist ein Element $p_{(x_i,x_j)}(a)$ der Matrix $P(a)$ und hat die Form:

$$
\begin{aligned}
& p_{(x_i,x_j)}(a) \\
&= Prob(x_j=(s',r',q')|x_i=(s,q,r),a) \\
&= p^{SP}_{(s,s')} * p^{SQ}_{(r,r')} * p^{SQ}_{(q,q')} \\
&= \begin{cases} p^{SP}_{(s,s')}(a) * p^{SQ}_{(q,Q')} * b(s,a), \text{ für } q'=q+z(r)-1 \\ p^{SP}_{(s,s')}(a) * p^{SQ}_{(q,Q')} * (1-b(s,a)), \text{ für } q'=q+z(r) \\ 0, sonst \end{cases}
\end{aligned}
\qquad [25]
$$

Eine Policy ist eine Sequenz von Befehlen $a\in\mathcal{A}$. Die Aufgabe des PM ist nun, eine optimale Policy π mittels linearer Optimierung nach festgelegten Leistungs- oder Energieverbrauchsbedinungen ($b(s,a)$ bzw. $c(s,a)$) zu finden. Dieser Vorgang wird *Policy Optimization* genannt [25].

Sofern die Wahrscheinlichkeiten zu jeder Zeit identisch und somit insgesamt unveränderlich sind, spricht man von einem stationären Modell. Stationäre Modelle haben den Nachteil, dass die Wahrscheinlichkeiten a priori festgelegt werden und sich nicht zur Laufzeit an veränderte Umgebungen anpassen, was nicht dem typischen Benutzerverhalten vieler Systeme entspricht. Chung et al erweitert daher das stationäre Modell um unbekanntes und nicht-stationäres Benutzerverhalten [18]. Qui et al. verbessert das Modell dahingehend, dass anstatt diskreten Zeiträumen von fester Länger, stetige Zeitverläufe verwendet werden, wodurch das Modell realitätsnaher und flexibler wird [26]. Eine weitere Beschränkung ist, dass Markov-Prozesse geometrisch oder exponentiell verteilt sind. Eine Studie hat jedoch herausgefunden, dass beispielsweise die Anfragen aus drahtloser Kommunikation besser mit der Pareto-Verteilung modelliert werden können, so dass Simunic et al. das Time-Index Semi-Markov-Modell entwickelt haben [27] Dieses Modell ist außerdem Ereignis-gesteuert, so dass im Gegensatz zu den Discrete-Time-Markov-Prozesse der aktuelle Zustand des SP nicht periodisch abgefragt werden muss, sondern die Zustandsübergänge durch Ereignisse ausgelöst werden [28], wodurch wiederum durch Reduzierung von unnötigem Polling der Energieverbrauch reduziert wird.

Vergleich der Strategien

Hu et al. liefert einen Vergleich verschiedener Strategien und bewertet diese nach verschiedenen Kriterien mit dem Ergebnis, dass die Wahl der Policy ein Kompromiss ist zwischen Energiebedarf,

Rechenleistung, Interaktivität und dem Ressourcenbedarf wie Arbeitsspeicher und CPU [22]. In wie fern sich eine Policy eignet ist insbesondere von der Hardware abhängt, für welche die Stromsparstrategie angewandt werden soll. Die Exponential Average-Strategie führt beispielsweise zu vielen Shutdowns, weshalb sie sich nicht für Festplatten eignet, da gerade dort der Overhead besonders groß ist. Nicht stationäre Policies sind effizienter als stationäre, weil sie sich an eine veränderte Arbeitslast anpassen können, verbrauchen aber dafür mehr Rechenzeit und vor allem mehr Speicherplatz. Generell wurde beobachtet, dass stochastische Strategien die Benutzer-Interaktivität besonders gut berücksichtigen.

Weitere Vergleiche würden den Rahmen dieser Arbeit sprengen, weshalb auf [29], [30] und [31] verwiesen sei.

Das „Fuzzy Decision Support System“ findet automatisch aus einem Satz von verschiedenen Policies die am besten geeigneten heraus [32]. Das Verfahren kann beeinflusst werden, in dem die Merkmale wie Energieverbrauch, Geschwindigkeit und Rechenleistung unterschiedlich gewichtet werden, so dass unter Berücksichtigung der eigens festgelegten Bedingungen die günstigste Strategie gewählt wird.

3.2.2 DVS (Dynamic Voltage Scaling)

DVS ist eine sehr effektive Methode, um den Energieverbrauch einer CPU zu reduzieren [33]. Die meisten Computersysteme zeichnen sich dadurch aus, dass sie über verschiedene Zeiträume hinweg unterschiedlich ausgelastet sind und der Prozessor somit oft mit einer höheren Frequenz als nötig läuft [34]. Die Frequenz eines Prozessors zu reduzieren, mindert zwar auch den Energieverbrauch des Prozessors insgesamt, allerdings bleibt der des einzelnen Prozesses gleich, da dieser dafür entsprechend länger dauert. Eine Reduzierung der Spannung erhöht zwar die Energieeffizienz, führt aber auch dazu, dass die maximale Frequenz des Prozessors ungefähr proportional abnimmt [35]. Auf der anderen Seite führt das quadratische Verhältnis von Energieverbrauch und Spannung dazu, dass auch geringe Spannungsminderungen großen Einfluss auf den Energieverbrauch haben [36] (siehe dazu Abschnitt Fehler: Referenz nicht gefunden) Die Aufgabe von DVS ist es nun, die Frequenz und die Spannung an die aktuelle Systemauslastung so anzupassen, dass ein Maximum an Energieeffizienz erreicht wird [33]. In [37] wurde gezeigt, dass durch einen DSP mit variabler Spannungsversorgung durch adaptives Anpassen der Spannung und der Frequenz der Energieverbrauch wesentlich verringert werden kann.

DVS findet sowohl auf Hardware-, als auch auf Softwareebene statt. Abbildung 7 zeigt die Hardware-Architektur eines skalierbaren Prozessors.

Der Energieverbrauch eines CMOS kann mit $E = N_{OPS} C V^2$ berechnet werden (siehe Kapitel 2.2), wobei N_{OPS} die Anzahl der Operationen, C die elektrische Kapazität pro Operation und V die Betriebsspannung ist. Auf der Mikro-Architektur-Ebene liegt der Fokus auf der Reduzierung von C. Ein DVS-fähiger Prozessor erfordert wesentliche Änderungen und Optimierungen im Chipdesign im Vergleich zu einem nicht skalierbaren Prozessors [38].

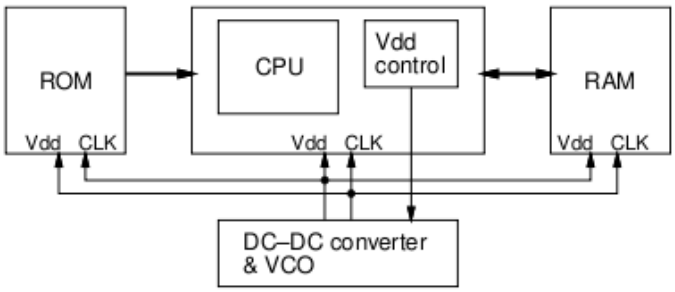

Abbildung 7: Architektur eines DVS-Prozessors

Auf Softwareebene kann grundsätzlich zwischen compilergesteuerten Strategien [39] und Laufzeit-Strategien unterschieden werden [40]. Während sich Erstere auf die Optimierung von N_{OPS} fokussieren, ist für Laufzeit-Strategien das Betriebssystem verantwortlich, dass durch einen *Voltage Scheduler* die Frequenz und die Spannung V des Prozessors kontrolliert und an die Systemauslastung anpasst [33].

Das Ziel des Voltage Schedulers ist, die Arbeitsleistung des Prozessors zu verringern, ohne die Deadline von Tasks zu überschreiten (siehe Abbildung 8) [41]. Es ist energieeffizienter, eine Aufgabe (engl. „task“) langsamer auszuführen, um das zur Verfügung stehende Zeitintervall möglichst ganz auszuschöpfen, als die Aufgabe so schnell wie möglich zu beenden und den Rest der Zeit im Leerlauf zu verbringen[42].

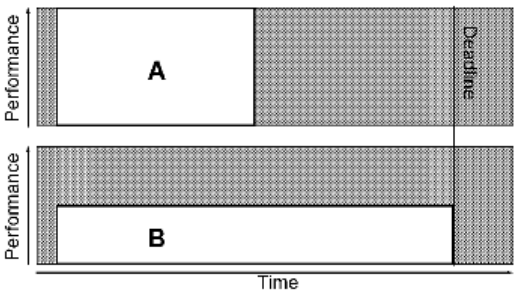

Abbildung 8: DVS und Deadlines

Algorithmen, welche die Arbeitsleistung steuern, können in zwei Kategorien, und zwar ob sie Echtzeit-Bedingungen unterliegen oder nicht, unterteilt werden [43]. In Echtzeit-Systemen müssen sich die Algorithmen an die vom Betriebssystemkern vorgegebenen Deadlines halten. Um die Echtzeit-Garantien zu erfüllen, nehmen die Algorithmen zunächst den Worst-Case, dass der Task die maximale Prozessorleistung benötigt, an. Da aber genau dieser Fall nicht immer eintritt, gibt es verschiedene Ansätze, die nicht benötigte Rechenzeit sinnvoll zu verwerten, um Energie zu sparen [36] [44] [45]. Weitere Arbeiten dazu finden sich beispielsweise bei [46] und [47].

Da Echtzeit-Systeme nicht Gegenstand dieser Arbeit sind, sei hiermit nur auf die zweite Kategorie von Algorithmen eingegangen, die Deadlines auf Grund von Beobachtungen der CPU-Auslastung der Vergangenheit automatisch festlegen (intervall-basierte Algorithmen). Eine der Grundideen stammt von Weiser et al., die einen intervall-basierten Algorithmus namens PAST entwickelten [48]. Dieser teilt die Zeit in Intervalle von fester Länge auf, innerhalb denen die Arbeit fertiggestellt werden soll. Die CPU-Frequenz wird an Hand der Leerlauf-Quote des vorhergehenden Intervalls so festgelegt, dass die Frequenz für das folgende Intervall erhöht wird, wenn die Arbeit in diesem Zeitraum nicht vollständig erledigt werden konnte und die Frequenz wird dementsprechend verringert, wenn im letzten Intervall eine hohe Leerlauf-Zeit gegeben war. Dieser Algorithmus ist zwar sehr einfach, hat aber auch den Nachteil, dass lediglich ein Intervall, und zwar das unmittelbar vorhergegangene, zur Berechnung herangezogen wird. Dies hat zur Folge, dass die Performance-Level zu oft wechseln, weil sich die Frequenz dauernd ändert. Außerdem werden dadurch die eher selten auftretenden Spitzenbelastungen unverhältnismäßig stark gewichtet. Govil et al. stellen, auf den Algorithmus von Weiser et al. aufbauend, eine Reihe weitere Algorithmen vor und fanden heraus, dass eine sanfte Anpassung der Prozessorgeschwindigkeit effektiver ist als häufige Geschwindigkeitswechsel, die auf Basis ausgefeilter Vorhersagestrategien initiiert werden [42]. Es ist also sinnvoller, möglichst lange eine akzeptablen Leistungslevel einzuhalten, als den Level in zu kurzen Zeitabständen exakt an die prognostizierte Arbeitsauslastung anzupassen.

Allen diesen Algorithmen wohnt der Nachteil inne, dass sie Annahmen über die zukünftige Entwicklung der Arbeitsauslastung machen. Pering et al. zeigte, dass intervall-basierte Algorithmen zwar gut bei gleichmäßiger CPU-Auslastung funktionieren (wie z.B. Audio-Playback), aber bei interaktiven Systemen mit stark schwankender Auslastung keine guten Ergebnisse erzielen [49] . Zu ähnlichen Ergebnissen kam auch Grundwald et al. [50]. Lorch et al. verbesserten intervall-basierte Algorithmen mittels statistischen Methoden und entwickelten den PACE-Algorithmus (Processor Acceleration to Conserve Energy). Dieser Algorithmus basiert auf der Wahrscheinlichkeitsverteilung der Anforderungen an die CPU und passen die Prozessorgeschwindigkeit auf Grundlage dieser Informationen während der Ausführung eines Jobs an [51]. Dabei startet der Job mit geringer CPU-Geschwindigkeit, die dann während der Ausführung soweit erhöht wird, wie es nötig ist, um den Job innerhalb der Deadline fertigzustellen. Einen anderen stochastischen Ansatz verfolgen Simunic et al. sowie Sinha und Chandrakasan, welche die Geschwindigkeit zu Beginn eines Jobs festlegen, und diese auch während der Ausführung konstant beibehalten [52] [34].

Bei allen genannten Strategien ist die Deadline fest vorgegeben. Flautner et al. gehen dagegen davon aus, dass die Verschiedenartigkeit von Aufgaben durch unterschiedliche Deadlines besser berücksichtigt werden kann und klassifizieren deshalb Abschnitte in interaktive, periodische Erzeuger und periodische Verbraucher mittels der Analyse von Kommunikationsmustern im Kernel des Betriebssystems. An Hand dieser Klassifikation werden die Deadlines für die Ausführungsabschnitte

automatisch vorhergesagt. Diese Methode richtet sich vor allem an interaktive Systeme. Durch die Berechnungen entsteht ein zu vernachlässigender Overhead. Insgesamt konnten Flautner et al. geringe Leistungseinbußen bei höherer Energieeffizienz nachweisen. Des weiteren schlagen sie vor, dass Anwendungsprogramme durch das Signalisieren von kritischen Abschnitten Einfluss nehmen können, um genauere Vorhersagen zu ermöglichen [41].

3.2.3 PM von verschiedenen Hardwarekomponenten

CPU

Das Power Management der CPU befasst sich neben dem DVS (siehe Kapitel 3.2.2) mit den der ACPI-Spezifikation entnommenen Performance States (P-States, siehe Kapitel) und Sleep-States (C-States, siehe Kapitel).

Speicher

Gegenstand der Forschung nach energieeffizienter Verwendung des Arbeitsspeichers sind im Wesentlichen die Halbleiter-Ebene, mit dem Chip-Design, die Architekturebene, die Speicherhierarchie betreffend, und die Datenorganisation [53]. Im Folgenden sei auf die letzten beiden Punkte eingegangen.

Um den Anforderungen moderner Computersysteme gerecht zu werden, sind Speicher typischerweise hierarchisch organisiert. An der Spitze stehen die CPU-Register, auf die mit voller Prozessorgeschwindigkeit zugegriffen werden kann, gefolgt von einer oder mehreren Cache-Ebenen von der Größe einiger weniger Megabytes. Anschließend folgt der Arbeitsspeicher in der Größenordnung zwischen einigen Megabyte bis zu einigen Gigabyte. Als nicht-flüchtiger Speicher folgt ein Hintergrundspeicher zur permanenten Speicherung von Daten und auf der untersten Ebene befinden sich optische Medien oder Bänder, die beispielsweise zu Sicherungs- oder Archivierungszwecken genutzt werden. Neben dem wachsenden Speicherplatz charakterisiert die Speicherhierarchie die sinkende Zugriffsgeschwindigkeit, den sinkenden Preis pro Byte [54] sowie den steigenden Energieverbrauch [55] mit tiefer werdender Hierarchieebene. Der Lokalitätseigenschaft Rechnung tragend sollen häufig verwendete Daten möglichst nah, also dort, wo der Aufwand des Speicherzugriffs möglichst gering ist, beim Prozessor gespeichert werden. Durch das Lokalitätsprinzip wird die Leistung des Speichersystems erhöht. Bellosa zeigt, dass der Datentransfer zwischen der CPU und dem 2nd Level Cache, der sich außerhalb der CPU befindet, erheblichen Einfluss auf den Gesamtenergieverbrauch des Systems hat und eine effiziente Cache-Verwaltung und die Vermeidung von Cache-Misses auch gleichzeitig energieeffizient sind [56]. Um das Lokalitätsprinzip zu unterstützen, sollten die Lebensdauer temporärer Daten möglichst gering gehalten werden, um den Speicherbedarf zu verringern [55].

Analog zu Prozessoren haben auch moderne DRAM-Speicherchips verschiedene Energiezustände, die pro Speicherchip einzeln gesteuert werden können. Die Steuerung übernimmt der Memory Controller, der auch für die Zustände der einzelnen Chips verantwortlich ist. Nur wenn der Chip sich im Aktiv-Zustand befindet, kann lesend oder schreibend auf ihn zugegriffen werden. Die anderen Schlafzustände unterscheiden sich darin, wie groß der Energiespareffekt ist und wie groß die Verzögerung ist, die durch das Versetzen in den Aktiv-Zustand entsteht .

Neben energiebewussten Optimierungen auf Halbleiter- und Speicherarchitekturebene kann laut Delaluz et al. zwischen zwei Ansätzen unterschieden werden, sich die verschiedenen Zustände von energiebewussten Speicherchips zu Nutze zu machen [53]. Beim ersten Ansatz entscheidet der Memory Controller an Hand von intern implementierten Policies selbstständig über die Zustände der Chips. Das Betriebssystem kann darauf keinen Einfluss nehmen. Fan et al. fanden heraus, dass es unter vielen Bedingungen am günstigsten ist, die Chips sofort in Schlafzustand zu versetzen, sobald eine Leerlaufphase auftritt und auf komplexe Vorhersagestrategien verzichtet werden sollte [57].

Der zweite Ansatz ist compiler- bzw. software-gesteuert. Auf Grundlage einer statischen Analyse des Programmverhaltens werden Leerlaufzeiten a priori erkannt, um den selektiven Zustandswechsel einzelner Speicherchips zu begünstigen. Weitere softwarebasierte Ansätze finden sich beispielsweise bei Wuytack et al. und da Silva et al., die mittels energieeffizienter Datenstrukturen den Energieverbrauch des Speichers senken ([58] und [59]), liegen aber außerhalb des Rahmen dieser Arbeit. Im Allgemeinen wirken sich viele Speicheroptimierungen auch positiv auf die Energieeffizienz aus [56].

Wie Fan et al. herausfanden, kann das Betriebssystem bei der Verwendung von virtuellem Speicher den Memory Controller durch intelligente Seiten-Allokation (eng. „page file allocation“) unterstützen. Sie entwickelten eine Strategie, mit der die Lokalität des Working Sets erhöht wird, was sich günstig auf die effiziente Ausnutzung der Schlafzustände einzelner Speicherchips auswirkt [60]. Ebenso zeigten Fan et al., dass mit der Kombination von DVS und energiebewusstem Speicher eine signifikante Menge Energie eingespart werden kann. Sie regen auch dazu an, weitere Synergieeffekte dadurch zu nutzen, dass der DVS-Scheduler die energiesparenden Funktionen des Speichers berücksichtigt und der Memory Controller auch auf Frequenzänderungen reagiert.

Weitere Arbeiten auf dem Gebiet der Energieeffizienz, auf die an dieser Stelle nicht weiter eingegangen sei, beschäftigen sich mit Scheduling auf Compiler-Ebene [61], der Register & Speicher-Allozierung [62] [63], der Reduzierung des Bus-Traffics [64], speziellem Memory Mapping für Speicherriegel mit mehreren Speicherbänken [65] und Caching [66].

Anzeige

Moderne LCD-Displays bieten dem Betriebssystem die Möglichkeit, die Intensität der Hintergrundbeleuchtung über eine Schnittstelle wie beispielsweise ACPI zu steuern. Dadurch kann eine große Menge an Energie eingespart werden, da die Anzeige eine der größten Energieverbraucher eines Computersystems ist.

Flinn et al. stellen in einer Studie eine Methode namens „zone backlighting“ vor, mit der sie den Stromverbrauch eines LCD-Displays um 7-29 % reduzieren. Dabei wird der Bildschirm in verschiedene Zonen aufgeteilt, die ihre Hintergrundbeleuchtung separat regeln können. In Analogie zu der Power-State-Verwaltung von Geräten steuert das Window-Manager-Subsystem des Betriebssystems die voneinander unabhängigen Zonen, in dem es beispielsweise die Fenster so auf dem Display platziert, dass es über möglichst wenige Zonen reicht. Eine weitere Möglichkeit ist, nur das Fenster, dass gerade den Fokus hat, voll zu beleuchten, und den Rest des Bildschirms zu dimmen oder auszuschalten [67]. Dem Autor ist allerdings zu diesem Zeitpunkt keine Hardware bekannt, welche diese Technik unterstützt.

Festplatte

Das Power Management von Festplatten ist ein weit erforschtes Feld, dass sich hauptsächlich mit der Verwalten von Schlafzustand von Festplatten in mobilen Geräten befasst und dort gute Ergebnisse erzielt, beispielsweise in [9] [68] [10] [20] [29] [19] [69] [70]. Gegenstand dieser Studien ist das Finden geeigneter Policies (siehe Kapitel 3.2).

Die Hersteller erkannten früh, dass die Festplatte eines der größten Energieverbraucher eines Computersystems ist und empfahlen, die Festplatte herunterzufahren, wenn sie sich über einen bestimmten Zeitraum im Leerlauf befindet [71]. Im einfachsten Fall lässt sich der Motor, der die drehenden Platten antreibt, abschalten. Charakteristisch für das Power Management von Festplatten ist, dass der Overhead, der beim Aktivieren der Festplatte aus einem Schlafzustand heraus entsteht, sehr hoch ist. Das Hochfahren des Motors (engl. „spin-up“) dauert verhältnismäßig lange und verbraucht relativ viel Energie, so dass der Zeitpunkt des spin-downs wohlüberlegt gewählt werden will (das sogenannte *Spindown-Problem*) [72]. Moderne Festplatten unterstützen deshalb mehrere Zustände mit unterschiedlichem Energieverbrauch und Aktivierungszeit [73]. Der ATA-Standard sieht vier Zustände vor:

- *Active*: Die Festplatte führt im Moment Operationen aus, z.B. Lesen oder Schreiben
- *Idle Mode*: Die Scheiben rotieren, aber der Lesekopf befindet sich in Parkposition
- *Standby Mode*: Der Festplattenmotor ist aus, aber die Festplatten-Schnittstelle ist immer noch aktiv
- *Sleep Mode*: Sowohl der Motor als auch die Schnittstelle sind aus. Um die Festplatte wieder zu aktivieren ist ein Reset notwendig.

Durch eine Reduzierung der Festplattenzugriffe, beispielsweise durch einen großen Cache-Speicher und einer Reduzierung der Cache-Miss-Rate, kann der Energieverbrauch direkt reduziert werden. Auch das Power Management der Festplatten wird indirekt begünstigt, da längere Leerlaufzeiten entstehen [2]. Insbesondere in Serverumgebungen, wo typischerweise keine langen Leerlaufzeiten auftreten und Leistungseinbußen durch die Spin-Up-Verzögerung nicht akzeptabel sind, ist Cache-Management eine bessere Alternative zum Power Management durch Zustandswechsel [74]. Einen weiterer Ansatz, der ohne Zustandswechsel auskommt, beschreiben Gurumurthi et al. Dabei wird die Rotationsgeschwindigkeit angepasst, anstatt den Festplattenmotor komplett auszuschalten [73]. Eine weitere Möglichkeit, die Festplattenaktivität zu verringern, ist Daten, die wahrscheinlich bald benötigt werden, im Arbeitsspeicher zwischenzuspeichern bevor die Festplatte in den Schlafzustand geht [12]. Eine ähnliche Idee verfolgt das von Satyanarayanan et al., entwickelte Coda-Dateisystem, dass Daten eines von mehreren Clients gemeinsam genutzten Dateisystem zwischenspeichert, solange ein mobiles Gerät mit dem Dateisystem verbunden ist. Dadurch kann auch wenn die Verbindung getrennt wurde weitergearbeitet werden kann [75].

Während zum Beispiel bei der Energieverwaltung von CPUs das Glätten der Leistung, also das Finden eines geeigneten Durchschnittslevels und die Vermeidung von großen Leistungsschwankungen, eine sinnvolle Strategie ist, beschäftigen sich andere Studien mit der Energieeffizienz von gegenteiligen Strategien. Pathanasiou et al. fanden heraus, dass die Erhöhung des Datendurchsatzes von Festplatten in der Zeit, in der die Festplatte aktiv ist, die darauf folgende Leerlaufzeit verlängert, was sich wiederum unterstützend auf des Festplatten-Power Management auswirkt. Die Technik stützt sich auf das Laden von Daten im Voraus sowie das Verschieben von nicht dringenden Zugriffen, wodurch bei Anwendungen mit relativ leicht vorhersehbarem Datendurchsatz wie Streaming der Energieverbrauch der Festplatte um 78,5 % gesenkt werden konnte [76].

Kommunikation

Drahtlose Kommunikation ist, wie bereits gezeigt, vor allem in mobilen Geräten ein großer Energieverbraucher. Aus Sicht von Betriebssystemen sind vor allem ad hoc-Netzwerke interessant, da durch das Bilden spontaner Netze die Aufgabe des Routens auf das Betriebssystem zurückfällt. Mit der Entwicklung von energieeffizienten Routing-Protokollen befassen sich Singh et al. [77]. Um den Energieverbrauch von drahtloser Kommunikation zu reduzieren, schlagen Vahdat et al. vor, die Reichweite von Funknetzwerken durch Reduzierung der Sendeleistung zu verkleinern. Eine Verkürzung der Reichweite um 30 % führt zu einer Energieeinsparung von 50 %. Dem zugrunde liegt die Beobachtung, dass fünf Sprünge zu verschiedenen Access Points über eine Distanz von zehn Metern weniger Energie verbrauchen als zehn Sprünge über jeweils fünf Meter. Der Nachteil ist allerdings, dass die Übertragung insgesamt länger dauert [78]. Auch hier ist also wieder ein Kompromiss zwischen Leistung und Energieeinsparung zu finden. Zu einem ähnlichen Schluss, und zwar dass kleine Netze energieeffizienter sind, kommen Singh et al bei ihrer Untersuchung von ad hoc-Netzwerken [77].

Andere Studien beschäftigen sich mit *Remote Processing.* Beim Remote Processing wird Energie dadurch gespart, dass bestimmte Aufgaben nicht lokal, sondern auf einem entfernten Rechner ausgeführt werden und der Client mit den fertigen Resultaten weiterarbeiten kann, ohne diese zuvor selber berechnet haben zu müssen [79] [80] [78]. Ein Beispiel dafür sind Web-Services.

3.2.4 Strukturell

Flinn et al. zeigten in einer Studie, dass durch Komforteinschränkungen, wie z.B. das Verringern der Framerate beim Abspielen von Videos, das Hardware-basierte Power-Management begünstigt wird. Die Effizienz dieser Methode ist stark von der Art der Daten und der Anwendung abhängig. Flinn et al. kommen zu dem Schluss, dass mit Unterstützung von Daten aus den Anwendungsprogrammen ein besserer Kompromiss zwischen Benutzerkomfort und Energieeffizienz gefunden werden kann [67]. Durch dynamische Umwandlung der Codierung von Bildern reduzierten Chandra et al. den Speicherplatzbedarf und Energieverbrauch in einem mobilen Gerät [81].

Vahdat et al. kommen zu dem Schluss, dass auch die interne Struktur von Betriebssystemen überdacht werden muss. Beispielsweise werden viele Aufgaben, wie z.B. *Garbage Collection,* periodisch ausgeführt. Das kann sich unter Umständen ungünstig auf die Energieeffizienz auswirken, da Leerlaufzeiten verkürzt werden, was wiederum das Power Management erschwert [78]. Auf der anderen Seite können periodische Aufgaben, da die Zeitpunkte der Ausführung bekannt sind, zusammengefasst ausgeführt werden, was auch im Linux Kernel bereits berücksichtigt wird.

3.2.4.1 Batterieverwaltung

Policies aus dem Bereich der Batterieverwaltung verfolgen einen anderen Ansatz als Policies, die sich an der Systemauslastung orientieren. Die Batterieverwaltung macht das Policy Management vom Ladezustand der Batterie abhängig. Eine mögliche Strategie für die Verlängerung der Laufzeit von Notebooks besteht darin, dass die Leitung des Systems bei niedriger Restlaufzeit gedrosselt wird, während bei vollständig oder fast vollständig geladener Batterie das System mit voller Leistung läuft.

3.2.5 ACPI & APM

Das Advanced Configuration and Power Interface (ACPI) ist ein, von Microsoft, Intel, Toshiba, Hewlett Packard und Phoenix entworfener, offener Industrie-Standard zur Konfiguration und Überwachung von Hardware [82]. Für die Energieverwaltung von Desktop-PCs, Notebooks und Servern, die auf Unix-artigen oder Windows-Plattformen betrieben werden, wird ACPI häufig als Schnittstelle zwischen Betriebssystem und Hardware verwendet.

Der Vorgänger von ACPI ist dass 1992 von Intel und Microsoft entwickelte APM (Advanced Power Management). Im Gegensatz zu ACPI ist APM ein BIOS-basierter Ansatz. Im BIOS oder der Firmware werden einfache Timeout-Policies implementiert und die Hardware nach Ablauf eines definieren Leerlauf-Zeitraums in den Schlafzustand versetzt oder ausgeschaltet. Für die Verwaltung der Energiesparmethoden ist die Firmware der Geräte also weitgehend selber verantwortlich. Dadurch wird die Implementierung von ausgefeilteren Policies erschwert, da es schwierig ist, einen komplexen BIOS-Code zu schreiben und die Größe der Firmware in der Regel stark limitiert ist [83].

Aus Sicht des Betriebssystems ist die Verwendung von APM zwar sehr einfach, aber auch sehr unflexibel, da nur sehr wenig Einflussmöglichkeiten gegeben sind. Des Weiteren fragt das Betriebssystem alle Informationen, wie beispielsweise die restliche Batterielaufzeit, über Real-Mode BIOS-Aufrufe ab, was mit zunehmender Verbreitung des 32-Bit Protected Mode zum Problem wird, da der Real-Mode aus Sicherheitsgründen in modernen Betriebssystemen problematisch ist. Ein weiterer Nachteil entsteht dadurch, dass die Kontrolle über die Geräte an deren Firmware abgegeben wird, was sich somit potentiell negativ auf die Systemstabilität, -effizienz und -konsistenz auswirken kann [84].

Um den Nachteilen von APM entgegen zu wirken wurde 1996 mit ACPI ein fortgeschrittener Standard vorgestellt, der das als fehlerhaft und unzuverlässig bekannte APM ablösen und die Basis für betriebssystem-gesteuertes Power Management („Operating system directed power management“, OSPM) schaffen soll. Die Aufgabe von OSPM ist es, die Plattform optimal zu konfigurieren und den Energieverbrauch, die Leistung und die Wärmeentwicklung des Systems unter Berücksichtigung der Benutzerpräferenzen und der Dienstgüte (QOS) des Betriebssystems [82] zu steuern. Im Gegensatz zu

APM ist ACPI selber kein Power Management, sondern übergibt dem Betriebssystem die Kontrolle über die, für die Energieverwaltung relevanten, Funktionen mittels einer standardisierten Schnittstelle [83], was einen Paradigmenwechsel von Firmware- zum betriebssystem-gesteuertes Power Management bedeutet.

Die der Spezifikation 3.0a entnommenen Ziele von ACPI und OSPM sind [82]:

1. Allen Computersystemen soll es möglich sein, Motherboard-Konfigurationen und Power Management Funktionen unter Berücksichtigung geeigneter Kosten- /Funktion–Kompromissen zu implementieren.
2. Bereitstellen eines robusten Rahmenwerkes für das Power Management. Die Policies werden vollständig im Betriebssystem implementiert, was einfacher ist als die Implementierung in der Firmware. Die Vereinheitlichung vermeidet Konflikte zwischen Betriebssystem und der Firmware, was die Zuverlässigkeit des Systems verbessert. Unterstützt wird das Betriebssystem dabei durch Power Management Informationen von Benutzern, Anwendungen und der Hardware.
3. Vorantreiben der industrieweiten Implementierung eines Power Managements. Betriebssysteme sollen unabhängig von der Hardware entwickelt werden können, wodurch alle ACPI-kompatiblen Computer vom Power Management profitieren können. Durch die Verlagerung des Power Managements in das Betriebssystem sollen redundante Investitionen vermieden werden.
4. Bereitstellen einer robusten Schnittstelle zur Konfiguration von Motherboard-Komponenten

3.2.5.1 Abstraktion

Um die Abhängigkeit zwischen der Firmware und dem Betriebssystem zu verringern, führt ACPI mit der *ACPI Source Language* (ASL) und der *ACPI Machine Language (AML)* einen Abstraktionsmechanismus ein. Die Funktionalität des Power Managements wird nun nicht mehr in der Firmware, sondern im Betriebssystem mittels ASL geschrieben. ASL ist eine plattform- und betriebssystemunabhängige Programmiersprache und wird von einem Compiler in einen plattformabhängigen Bytecode AML übersetzt. Diese Hardwarezugriffsbeschreibung wird von einem Interpreter, den das Betriebssystem bereitstellt, ausgeführt. AML abstrahiert das OSPM von der Hardware. Das OSPM erhält die volle Kontrolle, aber es werden die Hardwaredetails hinter Standard-Funktionsaufrufen verborgen [84]. Umgekehrt können Hardwarehersteller ASL-Codes unabhängig von dem eingesetzten Betriebssystem entwickeln.

BIOS- und Chipsatz-Hersteller benutzen ASL, um Systemkomponenten und Kontrollmethoden (engl. *Control Methods*) zu beschreiben. Diese Informationen werden in der Tabelle „Differentiated-System-Description-Table“ (DSDT) bereitgestellt und bilden einen baumartigen Namensraum (ACPI Namespace). Eine *control method* wird beispielsweise vom Betriebssystem aufgerufen, um die Temperatur eines Geräts zu erfragen oder einen Zustandswechsel zu veranlassen. Durch diesen Abstraktionsmechanismus können Hardware, Firmware und Betriebssysteme unabhängig voneinander entwickelt werden, da aus Sicht des OSPM die Implementierungsdetails der Hardwaresteuerung durch wohldefinierte Kontrollmethoden gekapselt werden. Das OSPM muss sich nur noch darum kümmern, wann der Aufruf der Methoden erfolgen soll, wodurch die Implementierung von komplexeren Power Management-Strategien ermöglicht wird. Abbildung 9 zeigt die ACPI-Schnittstelle zum Betriebssystem und zur Hardware. Weitere Vorteile sind laut [84]:

- Bessere Wartbarkeit, weil Fehler im AML einfacher als in der Firmware gefunden werden können
- Paralleles Ausführen mehrerer Aufrufe durch Threads, wodurch auch Blockierungen durch Interrupts, die gerade nicht behandelt werden, vermieden werden können

– Aufhebung der engen Speicherplatzgrenzen, die bei der Implementierung von Firmware im Allgemeinen gegeben sind.

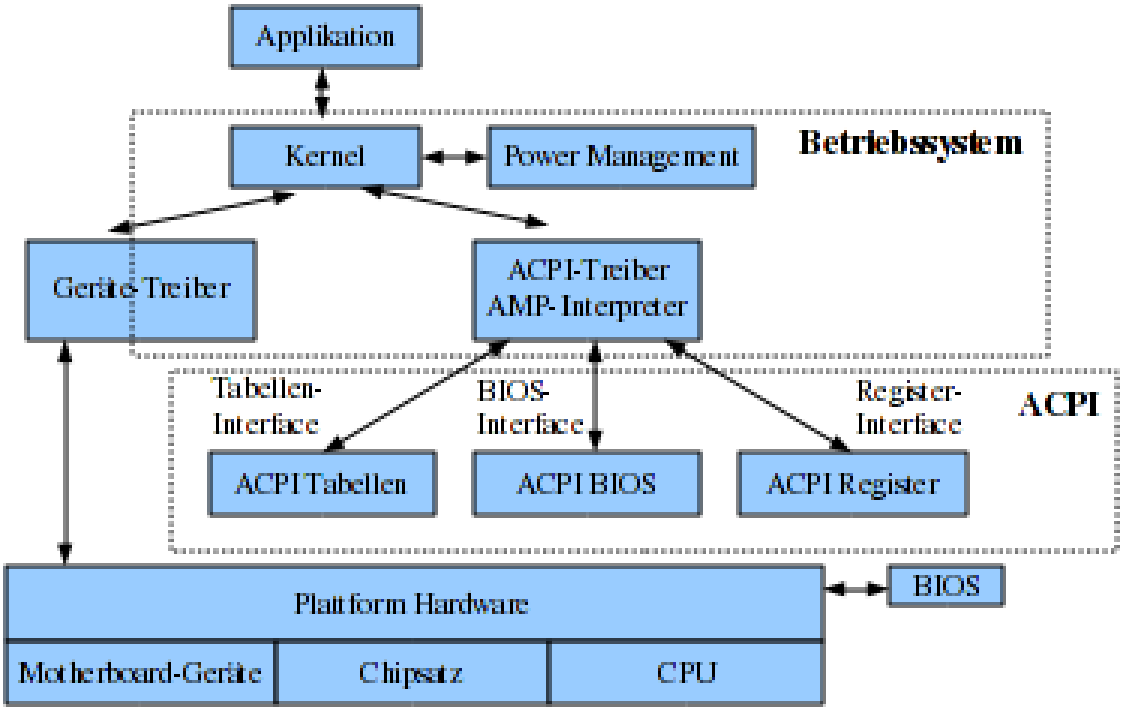

Abbildung 9: ACPI-Schnittstelle und PC-Plattform

3.2.5.2 Übersicht

ACPI-kompatible Plattformen bieten dem OSPM direkten und vor allem exklusiven Zugriff über die Energieverwaltung und die Motherboard-Konfiguration. Im Folgenden werden die für das Power Management relevanten Funktionsbereiche des ACPI low-level Interface vorgestellt [82].

System Power Management

Das vollständige Darstellung der ACPI Schlafzustände zeigt Abbildung 10, die der ACPI-Spezifikation entnommen wurde. ACPI kennt vier globale Zustände G0 bis G3, die für das gesamte Computersystem gelten und direkt für den Benutzer sichtbar sind. Während der Computer arbeitet und Anwendungen ausgeführt werden, befindet sich das System im „Working State“ G0. Wenn das System gerade nicht benötigt wird, kann es in den Zustand G1 („Sleeping“) überführt werden. In diesem Zustand werden keine Prozesse mehr ausgeführt. Der Computer kann aber mit verhältnismäßig geringem Aufwand wieder in den Arbeitszustand versetzt werden. Der Aufwand hängt von der Tiefe des Schlafzustands ab, der durch die vier Zustände S1 bis S4 innerhalb von G1 definiert wird. Je tiefer der Schlaf, desto weniger Energie wird verbraucht, aber desto länger dauert es auch, das System wieder in den Zustand S0 zu versetzen. Charakteristisch für die einzelnen Zustände ist auch, wie viel von dem Systemkontext verloren geht. Während bei S1 (Standby) noch der komplette Gerätekontext samt CPU-Kontext behalten wird, wird dieser im Zustand S3 („Suspend-To-RAM“) im Arbeitsspeicher gesichert. In S4, auch *„Non-Volatile-Sleep“, „Hibernate“* oder *„Suspend-To-Disk“* genannt, wird der gesamte Systemkontext auf einen nicht-flüchtigen Speicher geschrieben, so dass alle Geräte abgeschaltet werden können. Die Wiederherstellung des Systemkontextes zur Überführung in G0 dauert dann dafür auch entsprechend lange und kann auch durch ACPI-kompatible Geräte initiiert werden; zum Beispiel durch den Netzwerkadapter eines Druckservers, der beim Eintreffen

bestimmter Pakete, die beispielsweise einen Druckauftrag signalisieren, den Server wieder in den Arbeitszustand versetzt. Dazu aktiviert das Betriebssystem die „*wake* on“-Funktion des Netzwerkadapters und versetzt diesen anschließend in einen passenden Zustand. Dieser Zustand ist dadurch gekennzeichnet, dass er gerade noch soviel Energie verbraucht, wie das Gerät benötigt, um Signale zu empfangen und das System wieder in den Arbeitszustand zu versetzen. Alle anderen Geräte, die keine „wake-on“-Funktionen wahrnehmen können oder sollen, werden ausgeschaltet (Gerätezustand D3) und anschließend wird das System in den globalen Schlaf-Zustand (G1) versetzt.

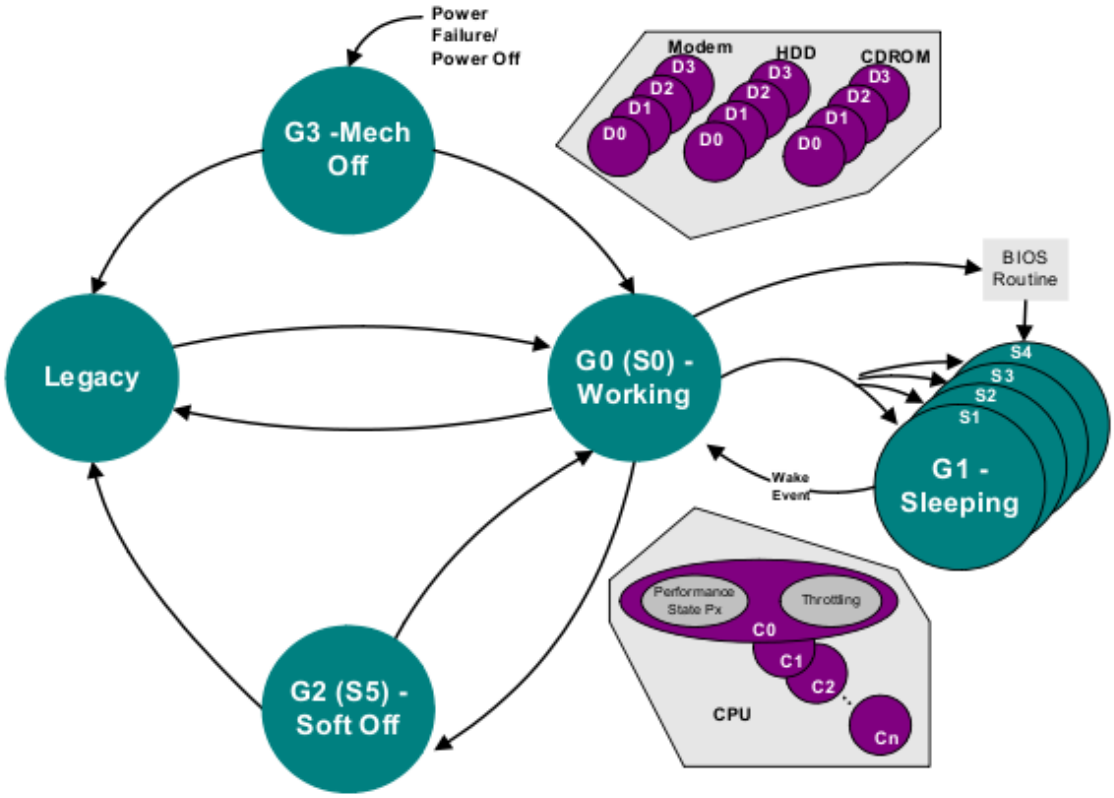

Abbildung 10: ACPI Power-Zustände

Der tiefste Schlafmodus ist S5 und entspricht dem globalen Zustand G2. Ähnlich wie bei S4 wird der Systemkontext dauerhaft gespeichert. Der Computer ist allerdings komplett ausgeschaltet. Wenn der Arbeitszustand wieder aufgenommen wird, muss das System neu gestartet und der Arbeitskontext wiederherstellt werden. Der Benutzer findet das System trotz eines Neustarts in dem gleichen Zustand vor, wie es beim Verlassen von G0 war. Der Energieverbrauch liegt bei nahezu Null Watt.

Im Zustand G3 ist der Computer vollständig ausgeschaltet und benötigt keinen Strom. Der Arbeitskontext geht verloren und das System kann nur durch den Power-Knopf wieder angeschaltet werden. Elektronische Aufweck-Mechanismen gibt es in diesem Zustand keine.

Device Power Management

Wenn sich das Gesamtsystem im Zustand G0 („*Working State*“) befindet, verwaltet das OSPM die Zustände D0 (vollständig eingeschaltet) bis D3 (vollständig abgeschaltet) der einzelnen Geräte. Nicht jedes Gerät unterstützt alle vier Power-Modi; vielmehr werden für allgemeine Geräteklassen (z.B. Modem, Festplatte, Netzwerkadapter usw.) Kriterien festgelegt, die beispielsweise den Stromverbrauch oder die benötigte Zeit, um ein Gerät wieder in den Zustand D0 zu versetzen, benennen. Des Weiteren werden Spezifikationen für verschiedene Busse, die die I/O-Geräte miteinander verbinden festgelegt, so dass beispielsweise ein Bus abgeschaltet werden kann, wenn alle verbundenen Geräte auch ausgeschaltet sind.

Das Betriebssystem bekommt die Informationen, welche Power Management Funktionen von einem Gerät unterstützt werden, über einer von BIOS bereitgestellten Beschreibung, dem „Differentiated Definition Block“ (DDB). Diese enthält unter anderem Informationen darüber, welche Stromquellen das Gerät in welchem Zustand benötigt und welche notwendig sind, um das Gerät aufzuwecken.

Prozessor Power Management

Wie jedes ACPI-kompatible Gerät besitzt auch die CPU vier Zustände (C0 bis C3), die im G0-Zustand zum Tragen kommen. In den Schlafzuständen C1, C2 und C3 führt die CPU kein Instruktionen aus. Durch Interrupts, wie zum Beispiel der Scheduler-Timer-Interrupt des Betriebssystems, wird der Prozessor wieder in den Arbeitszustand C0 überführt. Zu welchem Zeitpunkt die CPU in welchen Zustand versetzt wird, hängt von den im OSPM verwendeten Policies ab.

Device and Processor Performance States

Während sich die Geräte und Prozessoren im Arbeitszustand D0 bzw. C0 befinden, können sie zusätzlich verschiedene *Performance States* (P-States) (P1, P2, P3, …, Px) annehmen. Der Energiebedarf wird durch eine Verringerung der Leistung der jeweiligen Komponenten reduziert. Eine Festplatte kann beispielsweise reduzierte Datendurchsatzraten mit geringerem Energieverbrauch anbieten. Das OSPM wählt kann dann unter Berücksichtigung der Vorlieben des Benutzers oder der aktuellen Systemumgebung den geeignetsten Kompromiss aus.

Weniger effizient als die Performance States sind die *Throttling-States* (*T-States*). Diese Zustände passen nicht die Frequenz oder Betriebsspannung des Prozessors an, sondern versetzen die CPU innerhalb eines Zeitraums eine bestimmte Zeit lang in die Leerlaufphase, so dass beispielsweise zu 30% der Zeit gar nicht arbeitet und zu 70% unter Volllast.

Configuration / Plug and Play

ACPI stellt eine Schnittstelle zur Konfiguration von Geräten sowie Funktionen für Plug and Play bereit, die aber für das Power Management nicht spezifisch und somit auch nicht Gegenstand dieser Arbeit sind.

System Events

Die ACPI-Spezifikation beinhaltet ein Event-Model für Plug and Play, Thermal- und Power Management. Wird ein Event durch ein Gerät ausgelöst, ruft das Betriebssystem die dazugehörige *control method* auf um festzustellen, welches Event von welchem Gerät ausgelöst wurde. Die control method ist eine in AML-kodierte Funktion aus plattformspezifischen Bytecode, die vom Betriebssystem interpretiert und ausgeführt wird, um eine einfache Hardware-Aufgabe, wie z.B. das Auslesen des Batteriekapazität, zu lösen. Ein typisches Ereignis liegt beispielsweise vor, wenn die Batterie einen niedrigen Ladezustand signalisiert, so dass das OSPM angemessen darauf reagieren kann.

Battery Management

ACPI unterstützt zwei verschiedene Modelle des Batteriemanagements, die es dem Betriebssystem ermöglichen, an Information wie den Ladezustand oder die Entladungsrate der Batterie zu gelangen sowie über Ereignisse wie einen Wechsel der Stromquelle benachrichtigt zu werden.

- Die *Control Method Battery* – Schnittstelle stellt alle Informationen und Kontrollmöglichkeiten als AML *control method* bereit.
- Das *Smart Battery* – Subsystem wird durch einen *Embedded Controller* (EC) via SMBus gesteuert und bietet erweiterte Möglichkeiten des Batteriemanagements.

Beiden Modellen ist gleich, dass die Batterie Daten über die vom Hersteller angegebene Kapazität, die Kapazität bei der letzten vollständigen Ladung und die aktuell verbleibende Kapazität liefern muss. Die Daten werden entweder in Entladestrom mA und elektrische Ladung mAh oder Leistung mW und Energie mWh angegeben. Das Betriebssystem berechnet den Ladezustand im einfachsten Fall an Hand der Formel:

$$Ladezustand[\%] = \frac{\text{Verbleibende Kapazität [mAh / mWh]}}{\text{Letzte vollständige Ladung [mAh / mWh]}}$$

An Hand der aktuellen Entladungsrate kann die verbleibende Batterielaufzeit berechnet werden:

$$Verbleibende\,Laufzeit[h] = \frac{\text{Verbleibende Kapazität [mAh / mWh]}}{\text{Aktuelle Entladungsrate [mA / mW]}}$$

Thermal Management

ACPI erlaubt dem Betriebssystem auch Einfluss auf die Wärmeentwicklung des Systems zu nehmen. Die entscheidende Rolle dabei spielen *Thermal Zones*, durch die das Gesamtsystem in verschiedene Regionen eingeteilt wird. Für jede *Thermal Zone* können unterschiedliche Strategien festgelegt werden. Im einfachsten Fall ist das komplette System eine einzige Zone.

Es wird zwischen zwei Arten der Kühlung unterschieden. Bei der passiven Kühlung reduziert das OSPM den Energieverbrauch durch eine Reduzierung der Leistung, beispielsweise durch DVS, um die Temperatur zu verringern. Im Gegensatz dazu wird bei aktiver Kühlung zusätzliche Energie benötigt, um einen Lüfter zu betreiben. Die Leistung bleibt bei aktiver Kühlung unbeeinflusst. Ein Nebeneffekt der aktiven Kühlung ist die Lärmentwicklung, die sich negativ auf die Akzeptanz der Benutzer auswirken kann.

3.2.6 Fazit und Vorgehen bei der Evaluierung von Strategien

Bei der Evaluierung von Power Management-Strategien sollte der Fokus auf dem Gesamtersparnis des kompletten Systems liegen, da eine Betrachtung einzelner Komponenten auf Grund von Seiteneffekten nicht zwangsläufig Rückschlüsse auf den Gesamtverbrauch zulässt. Wichtig ist auch zu beachten, dass Optimierung hinsichtlich des Energieverbrauchs oft zu Einbußen bei der Bedienbarkeit des Systems und auf die Leistung des Systems insgesamt führen. Gerade bei zeitkritischen Anwendungen muss dies beachtet werden.

Bei mobilen Computern können Strategien des Power Managements außerdem die Tatsache berücksichtigen, dass die Batteriekapazität veränderlich ist.

Folglich bietet sich folgendes Vorgehen bei der Evaluierung von Power Management-Strategien an [12]:

1. Prüfen, wie hoch die Einsparung der untersuchten Komponente ist.
2. Prüfen, wie hoch der Anteil des Verbrauchs der Komponente im Gesamtsystem ist (im Durchschnitt).
3. Prüfen, wie stark wird der Energieverbrauch anderer Komponenten durch die einzelner Komponenten eingeschränkt wird.
4. Bei batteriebetriebenen Geräten prüfen, in wie fern die Batteriegesamtkapazitäten durch die Optimierungen beeinflusst werden.

3.3 Energieeffizienz in Linux 2.6

3.3.1 ACPI

3.3.1.1 ACPI-Subsystem

Das ACPI-Subsystem ist Bestandteil eines ACPI-fähigen Betriebssystems und besteht aus der ACPI Component Architecture (ACPICA) und der Serviceschicht OSL [85], die im Folgenden näher erläutert werden.

ACPICA

Die ACPI Component Architecture stellt eine betriebssystemunabhängige Referenzimplementierung für ACPI dar. Der ACPICA-Quellcode kann unverändert als Subsystem direkt in das Betriebssystem integriert werden. Die ACPICA macht keinerlei Annahmen über das Host-Betriebssystem. Die Kommunikation zwischen dem ACPICA-Subsystem und dem Host-Betriebssystem findet über eine zu implementierende Serviceschicht (OS Service Layer, OSL) statt. Dadurch wird ACPICA isoliert, um dessen Unabhängigkeit von dem Betriebssystem zu gewährleisten. Die Serviceschicht bietet Schnittstellen an, über die der ACPICA-Code, ACPI Core Subsystem genannt, auf Betriebssystemdienste zugreifen kann. Die Ausführung der Systemcalls obliegt alleine der Service Layer, die mit der ACPICA das ACPI-Subsystem bildet (siehe Abbildung 11).

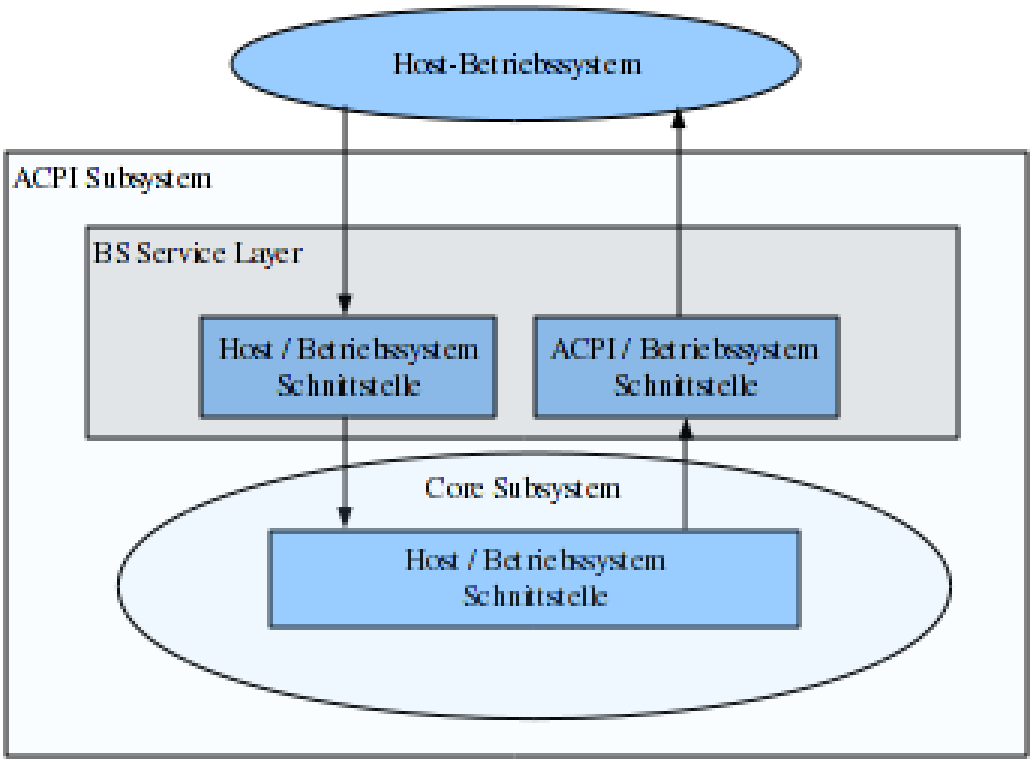

Abbildung 11: ACPI-Subsystem und Interaktion

Das ACPICA-Subsystem besteht aus mehreren Komponenten (siehe Abbildung 12). Der *AML-Interpreter* ist die grundlegende Komponente des ACPI-Core-Subsystems und für das Parsen und Ausführen des AML-Bytecodes zuständig. Die anderen Komponenten bauen auf den AML-Interpreter auf. Das *ACPI-Table Management* ist verantwortlich für die Verwaltung aller ACPI-Tabellen. Das *Namespace-Management* erzeugt und verwaltet den internen ACPI-Namensraum. Die Komponente

dient ebenfalls der Nummerierung der vorhandenen, ACPI-fähigen Geräte. Oberhalb das Namespace Managements und des AML Interpreters stellt das *Ressource Management* Abfrage und Konfigurationsmechanismen für die verfügbaren Ressourcen bereit. Das *ACPI Hardware Management* ist für den Zugriff auf die ACPI-fähige Hardware zuständig. Für die Verwaltung des ACPI System Control Interrupts (SCI), der den ACPI Timer und die ACPI Events bündelt, ist das *Event Handling* zuständig.

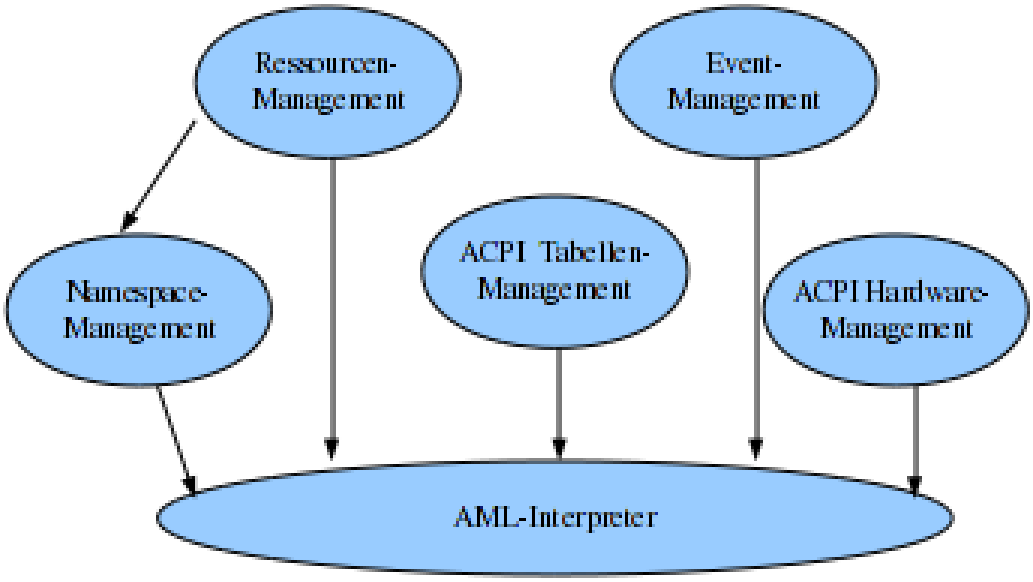

Abbildung 12: Interne Module des ACPICA Core Subsystems

Operating System Service Layer (OSL)

Während das ACPI-Core-Subsystem betriebssystemunabhängig ist, muss die Serviceschicht OSL für jedes Ziel-Betriebssystem explizit implementiert werden. Der OSL hat hauptsächlich drei Aufgaben:

1. Initialisierung des gesamten ACPI-Subsystems, einschließlich des OSL und ACPI-Core-Subsystems, sowie das Erkennen von ACPI-fähigen Geräten und das Laden deren Treiber.
2. Übersetzen der ACPI-Dienst-Anfragen vom Host-Betriebssystems in Aufrufe an das ACPI-Core-Subsystems.
3. Implementierung der von der ACPI-Spezifikation vorgegebenen Schnittstellen, die das ACPI-Core-System benutzt, um Betriebssystemdienste, wie z.B. Speicherverwaltung oder Thread-Scheduling, anzufordern.

3.3.1.2 Implementierung im Linux Kernel

Diese Kapitel bezieht sich auf den Linux Kernel v2.6.29 und ist dessen Dokumentation entnommen [86]. Auf dem ACPI-Subsystem aufsetzend (OSL und ACPI-Core-Subsystem) symbolisiert die Box „Linux/ACPI“ den Linux-spezifischen ACPI-Code. Im ACPI-Subsystem sind keinerlei Policies implementiert. Diese befinden sich in Modulen wie Button, Battery oder Processor. Dadurch wird gewährleistet, dass ausschließlich das Betriebssystem für die Implementierung von Power Management Strategien verantwortlich ist. Wie bereits erwähnt, besitzt ACPI selber keiner Power Management Funktionen, sondern stellt eine entsprechende Spezifikation für Hardware- und Betriebssystemhersteller bereit.

Der Daemon acpid (Advanced Configuration and Power Interface Event Daemon) ermöglicht es, im Userspace auf ACPI-Events zu reagieren. Abbildung 13 Zeigt zeigt die ACPI-Architektur unter Linux.

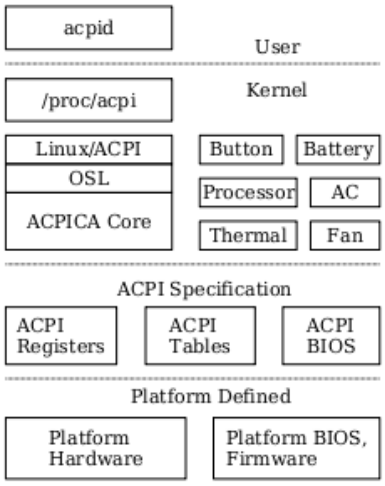

Abbildung 13: Linux ACPI-Architektur

Gerätetreiber und Linux Power Management

Grundsätzlich gibt es zwei verschiedene, sich nicht gegenseitig ausschließende Modelle des Device Power Managements, die Gerätetreiber nutzen können. Das System Sleep Model implementiert die ACPI-Zustände S1. S3 und S4, während das Runtime Power Management Model die Schlafzustände D1 bis D3 der Systemgeräte und -busse zur Laufzeit, während sich das Gesamtsystem im Zustand S0 befindet, repräsentiert.

System Sleep Model

Wird das gesamte System in einen Schlafzustand wie „suspend-to-RAM“ oder „suspend-to-disk“ versetzt, können die Treiber die Systemgeräte und -Busse ebenfalls in einen Schlaf-Zustand versetzen, um weniger Energie zu verbrauchen. Wichtig ist dabei, dass bei Wiederaufnahme des aktiven Systemzustands S0 der Gerätekontext verlustfrei wiederhergestellt wird. Dazu ist eine Kollaboration der Geräte-, Bus- und Geräteklassen-Treiber durch die Implementierung diverser rollenspezifischer Methoden notwendig. Zusätzlich müssen die Treiber berücksichtigen, ob ihre Geräte dazu in Lage sein sollen, das System durch Hardware-wakeups wieder in den aktiven Zustand zu versetzen. Diese Funktion kann in den entsprechenden /sys/devices/.../power/wakeup – Dateien aktiviert bzw. deaktiviert werden (Unter der Voraussetzung, dass sysfs unter */sys* eingehängt ist). Dabei ist zu beachten, dass die Aktivierung dieser Option für das gewählte Gerät auch einen höheren Stromverbrauch zur Folge hat. Auf der anderen Seite kann es dem System durch die flexiblere Konfiguration unter Umständen öfter ermöglicht werden, in Niedrigverbrauchs-Zustände zu wechseln. Ein Druckserver könnte beispielsweise nachts in einen Schlafzustand versetzt werden, wenn der Netzwerkadapter in der Lage ist, beim Empfang von Druckaufträgen den Server wieder aufzuwecken. Unterstützt der Netzwerkadapter diese Funktion nicht, muss der Server, um auf Druckaufträge reagieren zu können, ständig im Arbeitszustand sein.

Der Linux Kernel kennt drei systemweite Schlafzustände:

- Standby / Power-On Suspend,

  ACPI-Status: S1, String: „standby“.

  Das System schläft, aber der CPU-Kontext bleibt erhalten. Der Energiespareffekt ist sehr gering, dafür ist für die Wiederherstellung des Systems nur wenig Zeit notwendig. Sofern die Geräte den dazu äquivalenten ACPI-Zustand D1 unterstützen, wird dieser versucht zu aktivieren. Unterstützen die Geräte kein D1 oder schlägt die Überführung fehl, verbleiben diese in D0. Um das System wieder in den aktiven Zustand zu versetzen, sollten nicht mehr als 1-2 Sekunden notwendig sein.

- Suspend-to-RAM (STR)

  ACPI-Status: S3, String: „mem“

  Das gesamte System, und damit auch alle seine Komponenten, wird in einen Niedrigverbrauchszustand versetzt und nach Möglichkeit abgeschaltet (Zustand D3). Die einzige Ausnahme bildet der Arbeitsspeicher, in dem alle Informationen gespeichert werden, die notwendig sind, um den ursprünglichen Systemzustand wieder herzustellen. Die Wiederherstellung sollte nicht mehr als 3-5 Sekunden dauern.

- Suspend-to-Disk (STD)

ACPI-Status: S4, String: „disk“

Dieser Zustand hat den geringsten Energiebedarf. Es werden allerdings in der Regel auch mehr als 30 Sekunden benötigt, um den aktiven Systemzustand S0 wieder herzustellen. Der Zustand ähnelt Suspend-to-RAM, nur das als zusätzlicher letzter Schritt der Inhalt des Arbeitsspeicher auf einen Hintergrundspeicher, im Regelfall die Festplatte, geschrieben wird. Sobald der Systemkontext gesichert ist, kann der PC entweder in einen Niedrigverbrauchszustand analog zu Suspend-to-RAM versetzt werden oder es ganz ausgeschaltet werden, je nachdem, ob das System nur noch durch den Power-On-Knopf angeschaltet werden soll oder ob es auch möglich sein soll, das System durch Ereignisse wie das Öffnen der Laptop-Klappe aufzuwecken (Hardware-Wakeup).

Der Wechsel zu Suspend-to-RAM oder Suspend-to-disk erfolgt in mehreren Schritten. Zunächst werden mittels *sys_sync()* alle im Seiten-Cache (eng. „page cache“) befindlichen Daten auf den Hintergrundspeicher geschrieben. Bevor die Threads eingefroren werden, werden Suspend-Notifier an die registrierten Komponenten versendet, so dass diese gegebenenfalls noch Arbeiten erledigen können, bevor sie schlafen gelegt werden. Nachdem die Threads eingefroren wurden ruft Linux die Suspend-Methoden der Geräteklassen-Objekte auf, anschließend die der Treiber und zuletzt die des Busses. Bei Computern mit Mehrkernprozessoren werden nun alle Kerne, bis auf den Ersten, abgeschaltet und danach alle Interrupts deaktiviert. Anschließend folgt der Aufruf der an die Geräteobjekte gekoppelten *„suspend_late()“*-Methoden, um zuletzt alle Systemgeräte zu deaktivieren. Um das System wieder in den Zustand S0 zu versetzen, werden diese Schritte in umgekehrter Reihenfolge durchlaufen.

Alle PCI-Geräte unterstützen standardmäßig die Zustände D0 (vollständig eingeschaltet) und D3 (vollständig abgeschaltet), auch wenn sie nicht die ab PCI Version 2.2 eingeführte PCI PM Spezifikation erfüllen. Dadurch können alle sich am PCI-Bus befindlichen Geräte in den Zustand D3 versetzt werden, sofern ein systemweites suspend angefordert wurde.

Runtime Power Management Model

Befindet sich das Gesamtsystem im aktiven Zustand (S0), können Treiber die Geräte, die momentan nicht benötigt werden, in einen Zustand versetzten, in dem sie weniger Energie benötigen. Diese Zustände entsprechen D1 bis D3 der ACPI-Spezifikation, wobei nicht alle ACPI-fähigen Geräte alle Zustände unterstützen. Besonders effektiv ist das Runtime Power Management für Geräte, die selten benutzt werden. Entscheidend für die Einsparung einer signifikanten Menge von Energie ist, dass die Treiber vom systemweiten Power Management entkoppelt sind. Wenn beispielsweise ein Treiber ein Gerät zur Laufzeit in einen Schlafzustand D1 versetzt und später das komplette System in einen Schlafzustand S4 übergeht, wird auch das Gerät ausgeschaltet, also nach D3 überführt. Nach Wiederaufnehmen des systemweiten aktiven Zustand S0 soll sich das Gerät im Zustand D1 wiederfinden, also dem letzten gerätespezifischen Zustand vor dem systemweiten Schlaf-Zustand.

3.3.1.3 Userspace

ACPID

Der acpid (ACPI Event Daemon, [87]) ist dafür zuständig, Programme im Userspace über ACPI-Events zu informieren. Im Verzeichnis /etc/acpi/events können Konfigurationsdateien abgelegt werden, die Regeln zu bestimmten Events festlegen, um auf die Ereignisse zu reagieren [88]. Dadurch kann beispielsweise beim Entfernen des Netzsteckers ein Skript ausgeführt werden, dass die Hintergrundbeleuchtung des Displays reduziert.

SYSFS /sys/power

Des Weiteren stellt das Linux Power Management Subsystem im Userspace ein einheitliches sysfs-Interface im Verzeichnis */sys/power* bereit (wieder unter der Annahme, dass sysfs unter */sys* eingehängt ist). Im Folgenden sei auf die wichtigsten Dateien eingegangen:

Die Datei */sys/power/states* kontrolliert den systemweiten Status. Diese Datei enthält fix codiert alle verfügbaren Status als Zeichenkette, die da wären:

- „standby“ (Power-On-Suspend, S1)
- „mem“ (Suspend-to-RAM, S3)
- „disk“ (Suspend-to-disk, S4)

Ersetzt man den Inhalt der Datei durch eine dieser Zeichenketten, wird das System in den angegebenen Zustand überführt.

Die Datei */sys/power/disk* beeinflusst den suspend-to-disk-Mechanismus und legt fest, wie die Überführung in S4 initiiert werden kann. Das Suspend-to-disk kann beispielsweise durch einen Plattform-Treiber wie ACPI oder das Betätigen des Aus-Knopfs am PC-Gehäuse ausgelöst werden. Des Weiteren werden einige Optionen zum Testen bereitgestellt.

Die angestrebte Größe des Abbilds, das beim suspend-to-disk erzeugt wird, wird in der Datei */sys/power/Image_sie* festgelegt. Wenn es nicht möglich ist, die Größe einzuhalten, wird dennoch in den Zustand S4 gewechselt und versucht, dass Abbild möglichst klein zu halten. Der Standardwert liegt bei 500 MB.

3.3.2 Tickless Kernel

Ältere Linux-Kernels benutzten einen periodischen Timer-Interrupt, oft „timer tick“ genannt, der mit einer Frequenz zwischen 100 und 1000 Hz ausgelöst wird. Der Timer Tick wird zur Erledigung folgender Aufgaben benötigt:

- Inkrementierung der Jiffies, einer kernelinternen Zeit-Notation. Jiffies werden an vielen Stellen verwendet und repräsentieren die Anzahl der Ticks seit dem Starten des Systems.
- Prüfen, ob verzögerte Aufgaben auszuführen sind. Das können periodische Aktivitäten (bspw. eine Aufgabe alle 30 Sekunden auszuführen) oder Timeouts (nach 15 Sekunden Inaktivität eine bestimmte Aufgabe auszuführen) sein.
- Prozessabrechnung. Bei jedem Timer Tick wird die CPU-Benutzung der aktiven Prozesse aktualisiert.
- Prozess-Scheduling. Prüfen, ob die Zeitscheibe des aktiven Prozesse abgelaufen ist und gegebenenfalls die nächste Zeitscheibe einem anderen Prozess zuteilen.

Der große Nachteil des Timer Ticks besteht darin, dass der Timer-Interrupt immer ausgelöst wird, auch dann, wenn er gar nicht benötigt wird. Das hindert die CPU daran, in einen C-Zustand zu wechseln, der weniger Energie verbraucht bzw. es wird die Zeit verkürzt, in der die CPU in einem solchen Zustand verbleiben kann, da mit jedem Interrupt die CPU aufgeweckt und in den Zustand C0 überführt wird.

Mit dem Linux Kernel 2.6.21 wurde der Tickless Kernel eingeführt, der den Timer-Interrupt nur bei Bedarf auslöst. Da Prozessabrechnung und Prozess-scheduling während einer Leerlaufphase nicht notwendig sind, mussten bei der Entwicklung des Tickless Kernels nur für die ersten beiden Punkte, die Inkrementierung der Jiffies und die Ausführung verzögerter Aktivitäten, Alternativen gefunden werden.

Zum Aktualisieren der Jiffies wird nun ein Hardware-Timer verwendet. Anstatt die Jiffies bei jedem Timer Tick zu inkrementieren, wird der Wert an Hand der von einem Hardware-Timer gelieferten Zeit ermittelt. Die zuverlässigsten Ergebnisse liefert dabei der HPET (High Precision Event Timer).

Um periodische Ereignisse steuern zu können, bedient sich der Kernel einer softwaregesteuerten Ereignisquelle, also einem Gerät, dass zu einem festlegbaren Zeitpunkt einen Interrupt auslöst. Der Kernel berechnet die Zeit, wann der nächste Timer abgelaufen sein wird und weist ein solches Gerät an, genau zu diesem Zeitpunkt einen Interrupt auszulösen. Anschließend kann die CPU in einen Niedrigverbrauchszustand wechseln und verbleibt dort, bis sie durch den eigens initiierten Interrupt aufgeweckt wird.

Das Problem dieses Ansatzes ist, dass die Effizienz stark von der Anzahl der auftretenden Interrupts abhängt. Diese Anzahl ist typischerweise sehr hoch, weil sich bis zur Entwicklung des Tickless Kernels, auf Grund des periodischen Timers, unnötige Interrupts nicht negativ ausgewirkt haben. Dies hat zur Folge, dass Softwareentwicklerinnen und Softwareentwickler wenig sorgfältig mit der Verwendung von Interrupts umgegangen sind. Damit der Tickless Kernel effizient arbeiten kann, muss also die Anzahl der ausgelösten Interrupts reduziert werden. Eine Möglichkeit ist, unnötiges Polling zu vermeiden und die Intervalle, in denen periodisch auftretende Aktivitäten stattfinden, so zu verlängern, dass kein erkennbarer Nachteil entsteht. Eine weitere Möglichkeit ist, Interrupts möglichst zeitgleich auftreten zu lassen. Man macht sich dabei den Umstand zu nutze, dass es für viele Ereignisse nicht wichtig ist, wann genau der Interrupt ausgelöst werden soll, sondern nur der Intervall, beispielsweise alle zwei Sekunden, von Bedeutung ist. Die Kernel API stellt deshalb Funktionen bereit, die die Jiffies auf die volle Sekunde runden, so dass alle Timer zu gleichen Zeit, also zur vollen Sekunde, ausgelöst werden. Für den Userspace wird in der Bibliothek „glibc“ eine ähnliche Funktion angeboten.

3.3.3 Anwendung

3.3.3.1 CPU Frequency Scaling – Subsystem (cpufreq)

Das cpufreq-Subsystem ist verantwortlich für die Verwaltung der *P-States* (Performance States), die während des systemweiten Zustands G0 und des Arbeitszustand C0 des Prozessors verfügbar sind. Die P-States unterscheiden sich in der Frequenz und der Betriebsspannung. Niedrigere Frequenzen erlauben eine niedrigere Betriebsspannung, weshalb durch P-States der Energiebedarf des Prozessors signifikant reduziert werden kann. Eine cpufreq-Policy besteht zunächst aus einem Intervall [min, max], dass den Bereich der zulässigen Frequenzen angibt. Bei CPUs, die in der Lage sind, unabhängig und ohne Einwirken des Betriebssystems ihre Frequenz anzupassen, wird dieser lediglich über das cpufreq-Subsystem der minimale und der maximale Frequenzwert mitgeteilt. Bei allen anderen Prozessoren wird die Zielfrequenz, die innerhalb des angegebenen Intervalls liegen muss, durch den sogenannten Governor gewählt. Der Governor entspricht dem in Kapitel 3.2 verwendeten Begriff der Policy. Die Konfiguration des cpufreq-Subsystem, wie z.B. die Auswahl der zu verwendenden Policy, die Festlegung des Frequenzintervalls oder des CPU-Treibers, erfolgt über das sysfs-Interface.

Im Gegensatz zu den P-States wird bei *T-States* (Throttling-States) nicht die Frequenz oder die Betriebsspannung angepasst. Im Gegensatz dazu wird die CPU bei maximaler Leistung betrieben und für bestimmte Zeiträume komplett abgeschaltet, wodurch der Energiespareffekt nur gering ist. T-States werden im Linux Kernel lediglich zur Reaktion auf thermische Ereignisse, wenn der Prozessor zu heiß wird, verwendet.

Seit dem Linux Kernel 2.6.18 ist es durch den *sched_mc_power_savings*-Parameter möglich, dass Scheduling bei Multicore-Prozessoren zu beeinflussen. Aktiviert man diesen Parameter über das sysfs-Interface (/sys/devices/system/cpu/sched_mc_power_savings), so versucht der Scheduler so wenig CPUs wie möglich zu benutzen. Erst wenn alle verwendeten CPUs vollständig ausgelastet werden, wird eine weitere CPU aktiviert. Nicht aktive CPUs können derweil in energiesparenden Schlafzuständen verweilen. Dieses Verfahren ist besonders bei geringer Systemauslastung (CPU-

Load < Anzahl der CPUs), interessant, da es sich auf die Arbeitsleistung einiger Anwendungen auswirken kann. Der CPU-Load entspricht der aktuellen Länge der Run-Queue des CPU-Schedulers.

3.3.3.2 Festplatte

Für das Power Management von Festplatten ist entscheidend, die interne Power-State-Verwaltung durch Reduzierung der Zugriffe zu unterstützen, sofern diese ein Power Management anbieten. Außerdem können Festplatten auch manuell in einen Schlafzustand versetzt werden, was in der Regel mit dem Werkzeug „hdparm“ geschieht [89].

Caching erlaubt es der Festplatte, möglichst viel Zeit im Leerlauf bzw. bei entsprechendem Power Management in Schlafzuständen zu verbringen, da I/O-Anfragen für eine gewisse Zeit aufgeschoben werden können.

Durch das Aufschieben von I/O-Anfragen kann neben der Leistung auch die Zeit, in der sich eine Festplatte in einem Niedrigverbrauchszustand befindet, verlängert werden. Das VM-Subsystem im Linux Kernel nutzt einen Zwischenspeicher für Schreibzugriffe, so dass die Zugriffe gebündelt erfolgen. Wie lange die Schreibzugriffe zwischengespeichert werden, hängt von dem Wert in „/proc/sys/vm/dirty_writeback_centisecs“ ab und liegt standardmäßig bei 5000 Millisekunden. Der Nachteil ist, dass bei einem Ausfall der Festplatte die zwischengespeicherten Daten verloren gehen. Ein ähnliches Caching-Verfahren wird durch das Werkzeug *Laptop-mode-tools [90]* angeboten. Weitere Möglichkeiten, Festplattenzugriffe zu reduzieren, bestehen durch den noatime-Parameter. Dieser gibt an, dass kein Zeitstempel für die Zugriffszeit („Access Time“) einer Datei gespeichert wird, was zur Folge hätte, dass alle Lesezugriffe auf eine Datei implizit zu Schreiboperationen führen würde. Weiterhin kann das Logging von (System)-Ereignissen reduziert werden, um unnötige Schreibzugriffe zu vermeiden.

Auf Dateisystemebene bieten sich die Möglichkeiten, bei ausreichend großem Arbeitsspeicher die Auslagerungsdatei zu deaktivieren und temporäre Dateien auf einer Ramdisk (temporary file storage, tmpfs) zu speichern.

4. Anwendung

4.1 Konzept

4.1.1 Testumgebung

4.1.1.1 Hardwareausstattung

Als Testrechner wird ein üblicher Desktop-PC verwendet, der keine besondere, auf Energiesparsamkeit ausgelegte, Hardware verwendet, mit folgender Ausstattung:

Mainboard	ASUS K8V Rev. 1.xx
Prozessor	AMD Athlon 64 3000+ (Single Core)
L1-Cache	64 kB intern
L2-Cache	512 kB extern
Southbridge	VIA VT 8237
Northbridge	VIA K8T800
Arbeitsspeicher	2 x 512 MB DIMM SDRAM
Festplatte	ATA Seagate ST340014A 40 GB
Grafikkarte	Radeon RV100 QY [Radeon 7000/VE]
DVD-Laufwerk	DBD-ROM DVD 16X5H
Soundkarte	VT8233/A/8235/8237 AC 97 Audio Controller (onboard)
Netzwerkadapter	Marvel Technology Group Ltd. 88E8001 Gigabit Ethernet Controller (onboard)
USB Controller	VT82xx UHCI USB 1.1 Controller

Tabelle 2: Hardwareausstattung des Testrechners

Alle Komponenten, die für die verwendeten Szenarien nicht benötigt werden, wie zusätzliche USB- und PCI-Steckplätze, werden entweder entfernt oder im BIOS deaktiviert. In allen Szenarien wird die gleiche Hardware in der gleichen Konfiguration verwendet. Eine detailliertere Auflistung der verwendeten Hardware befindet sich im Anhang A.

4.1.1.2 Messgerät

Als Messgerät kommt ein EMU10 Memo Set zum Einsatz. Das Gerät zeichnet sich dadurch aus, dass es neben Stromspannung und Stromstärke auch die Energie mit einer Genauigkeit von 0,1 Watt misst. Die gesammelten Daten werden im Speicher des Geräts vorgehalten und können über eine PC-Schnittstelle ausgelesen werden. Die Messungen werden mittels eines Adapters direkt zwischen Netzanschluss und dem zu messen Gerät vorgenommen.

4.1.2 Szenarien

Ziel der Szenarien ist es, ein hinsichtlich Energieeffizienz optimiertes Systeme andere Systeme gegenüber zu stellen, die eher universal ausgerichtet sind und einen möglichst breiten Nutzen bieten sollen. An Hand der Szenarien werden die möglichen Optimierungen analysiert und deren Auswirkungen untersucht.

4.1.2.1 Betriebssystem und Anwendungssoftware

Die Grundlage aller Szenarien bilden verschiedene Linux-Betriebssysteme, da sie nicht zuletzt auf Grund ihrer Quelloffenheit eine große Bandbreite an Konfigurations- und Analysemöglichkeiten bieten. Außerdem nimmt die Verbreitung von Linux im Bereich der Desktop-PCs und Notebooks weiter zu, so dass auch dessen Energieeffizienz weiter an Relevanz gewinnt. Die Betriebssysteme werden auf separaten, baugleichen, Festplatten installiert. Als grafische Arbeitsumgebung wird jeweils „Gnome“ [91] eingesetzt. Die für die Arbeitsabläufe benötigten Anwendungsprogramme sind für alle Szenarien gleich und werden, sofern nicht bereits in der verwendeten Linux-Distribution vorhanden, nachträglich installiert. Dadurch unterscheiden sich die unterschiedlichen Szenarien nur in der verwendeten Distribution und dem Linux-Kernel.

Eine detaillierte Auflistung der verwendeten Software findet sich in der folgenden Tabelle:

Anwendung	Version	Verwendung
xorg-server	1.5.1	X-Window System
Gnome	2.24.1	Arbeitsumgebung / grafische Benutzeroberfläche
Mozilla Firefox	3.0.1	Webbrowser, Internet-Anwendungen
Mozilla Thunderbird	2.0..21	E-Mail-Client
Evince	2.24.2	PDF-Dokumentenbetrachter
OpenOffice	3.0.0	Textverarbeitung (writer) und Tabellenkalkulation (calc)
gcc	4.3.2-r3	Softwareentwicklung / Kompilation
Nautilus	2.24.2-r3	Datei-Browser

Tabelle 3: Software des Testrechners

Szenario I:

Im ersten Szenario wird die Distribution „Gentoo Linux“ [92] benutzt um ein möglichst energieeffizientes System zu schaffen. Gentoo eignet sich als quellcode-basierte Linux-Distribution, da es viele Optimierungs- und Konfigurationsmöglichkeiten bietet und es wenig automatisierte Abläufe gibt [93]. Wegen seiner nahezu unbeschränkten Anpassungsfähigkeit wird Gentoo von der Gentoo-Community auch oft als Meta-Distribution bezeichnet [94].

Aus diesen spezifischen Eigenschaften ergeben sich auch die Nachteile von Gentoo. Zum einen nimmt die Installation des Grundsystems viel Zeit in Anspruch, da es keine geführte Installation mittels automatisierten Setup-Routinen gibt, wie es bei vielen Distributionen üblich ist, zum anderen müssen alle Pakete vor der Installation zunächst kompiliert werden. Dieses aufwändige Verfahren garantiert jedoch ein Maximum an Kontrolle und ermöglichst es, die Installation von unnötigen und ungewollten Paketen zu vermeiden. Für dieses Szenario ist ein möglichst minimales System angestrebt.

Durch die Auslieferung der Pakete im Quellcode müssen diese, unterstützt durch den Paketmanager „portage“ [95], zunächst kompiliert werden, was wiederum sehr Zeitaufwändig und Rechenintensiv ist und die nachträgliche Installation von Anwendungen, im Vergleich zu Distributionen mit binären Software-Paketen, unkomfortabel macht. Gentoo eignet sich deshalb nicht für Systeme, an denen oft Änderungen an der Softwareauswahl vorgenommen werden. Die Kompilationsdauer des gesamtem Testsystems liegt bei ca. 45 Stunden.

Im Fokus liegt die Konfiguration des Kernels (Version 2.6.29-r5). Der Kernel soll zum einen minimal und zum anderen auf Energieeffizienz optimiert sein. Es werden die in der Analyse gewonnen Erkenntnisse berücksichtigt, wonach sich das Power Managements, insbesondere das des Prozessors, und damit einhergehend die Anzahl der ausgelösten Interrupts und das Prozess-Scheduling, auf den

Energiebedarf des stark Gesamtsystems auswirken kann. Zusätzlich soll das Power Management der CPU durch geeignete Policies optimiert und das Power Management der Festplatte begünstigt werden.

Szenario II:

Als Betriebssystem des zweiten Szenarios wird OpenSuse 11.1 eingesetzt. Es werden bei der Installation keine besonderen Konfigurationen oder Änderungen vorgenommen. Über den Yast-Paketmanager [96] wird das System auf den neuesten Stand gebracht. Der Kernel und die Anwendungsprogramme werden aktualisiert, um die Szenarien bestmöglich vergleichen zu können.

Szenario III:

Das dritte Szenario ähnelt dem zweiten Szenario, da die Plattform bis auf den Kernel identisch ist. Als Kernel wird jedoch das optimierte Exemplar aus Szenario I verwendet. Dieses Szenario soll dazu dienen, den Einfluss des Betriebssystemkerns detaillierter untersuchen zu können.

4.1.3 Benutzerprofile

Jedes dieser Szenarien wird an Hand unterschiedlicher Benutzerprofile untersucht, um eine möglichst hohe Bandbreite an unterschiedlichem Benutzerverhalten berücksichtigen zu können. Es ist zu erwarten, dass verschiedene Arten der Nutzung zu unterschiedlichen Ergebnissen führen, da bereits in der Analyse der Arbeit der Einfluss von Interaktion in ereignisgesteuerten Systemen wie Büro-PCs gezeigt wurde. Die folgenden Tabelle zeigt die zwei verschiedenen Benutzertypen in ihren grundlegenden Eigenschaften. Der genaue Ablauf der Benutzersimulation wird im Anhang B in Form von Shell-Skripten aufgeführt.

Benutzertyp	Nutzung	Beschreibung
Light User	Benutzt normalerweise immer nur eine Anwendung und schließt diese vor der Nutzung einer anderen Anwendung. Typische Programme sind Web-Browser und Textverarbeitungsprogramm.	- geringe Benutzeraktivität - lange Leerlaufphasen - geringe Anzahl parallel laufender Prozesse - wenige genutzte Anwendungen insgesamt
Medium User	Benutzt zwei oder drei Anwendungen gleichzeitig, z.B. E-Mail-Clients, Office-Anwendungen und Internetradio oder Compiler	- höhere Benutzeraktivität - kürzere Leerlaufphasen - mehr parallel laufende Prozesse - höhere Anzahl auch gleichzeitig genutzter Anwendungen.

Tabelle 4: Benutzerprofile

4.1.4 Optimierungsziele

Grundlage für die Optimierungen bilden die für die Benutzerprofile aus Testläufen gewonnenen Systemdaten über die CPU- und Speicherauslastung, die Anzahl der ausgelösten Interrupts, usw. Dass diese Daten a priori bekannt sind entspricht nicht der Realität. Es wird aber davon ausgegangen, dass diese hinreichend repräsentativ für die verwendeten Benutzerprofile und Computersysteme sind. Diese Arbeit diskutiert Optimierungsstrategien für verschiedene Benutzerprofile, ausgehend von einer klar definierten Ausgangssituationen und möchte dahingehend Anknüpfungspunkte für weitere Untersuchungen bieten.

Ausgehend von der in der Analyse gewonnen Erkenntnisse zielt die Energieeffizienz-Optimierung auf die drei folgenden Kernaspekte:

1) CPU: Bessere Ausnutzung der CPU-Schlafzustände C1-Cx sowie der Performance-Level P0-Px
2) Festplatte: Unterstützung des festplatteneigenen Power Managements, insbesondere durch Reduzierung der Festplattenzugriffe
3) Interrupts: Entlastung der CPU und anderer Komponenten durch Reduzierung vermeidbarer Interrupts.

Die Optimierungen beziehen sich somit in erster Linie auf die größten Energieverbraucher eines PC-Systems, wobei die Anzeige außen vor gelassen wird. Dabei müssen neben den Wechselwirkungen der einzelnen Optimierungszielen auch die Beeinträchtigung der Systemleistung, insbesondere wenn sie die Beeinträchtigung negativ auf die Bedienbarkeit des Systems auswirkt, untersucht werden. Alle Ziele haben gemeinsam, dass bei dem Entwurf des optimierten Systems dieses so minimal wie möglich gehalten wird. Dies wirkt sich vor allem bei dem Kernel aus, in dem alle nicht benötigten Optionen, Treiber und Einstellungen deaktiviert wurden. Im Folgenden werden die einzelnen Optimierungsziele im Detail erläutert.

4.1.4.1 CPU

Die Optimierungen hinsichtlich des Energiebedarfs des Prozessors umfassen zwei grundlegende Bereiche.

Zum einen soll der Wechsel in energiesparsame C-States durch Entlastung der CPU begünstigt werden. Es wird untersucht, in wie fern sich ein optimiertes System auf die Belastung der CPU und somit auf die C-State-Verwaltung auswirkt. Außerdem wird der Einfluss verschiedener, im Linux Kernel zur Verfügung stehender, Scheduling-Algorithmen auf die Verweildauer in den einzelnen C-States berücksichtigt.

Zum anderen werden für jedes Benutzerprofil spezifische Policies analysiert, die mittels des cpufreq-Subsystem des Kernels die Performance-Zustände der CPU (P-States) verwalten. Es wird angenommen, dass weniger rechenintensive Profile wie das des „Light Users“ auch mit geringerer CPU-Frequenz die definierten Ziele hinreichend erfüllen, da die Leistungsanforderungen im Mittel geringer sind. Ein wichtiger Gesichtspunkt ist, inwieweit die beiden Herangehensweisen bezüglich der C-States und der P-States zueinander im Gegensatz stehen. Eine sehr granulare Anpassung der CPU-Frequenz an die momentane Leistungsanforderungen kann die Dauer der Prozesse verlängern und somit die Leerlaufzeit verkürzen, was negative Auswirkungen auf die C-States haben kann. Ziel ist es, für die einzelnen Benutzerprofile eine optimale Balance zwischen C-States und P-States zu finden.

Keine Berücksichtigung finden die *Throttling-States* (*T-States*) der CPU, da diese in Linux in erster Linie zur Regulierung der Temperatur in den thermischen Zonen verwendet wird. Außerdem wird, wie bereits erwähnt, der T-States-Verwaltung keine große Einflussmöglichkeit auf den Energieverbrauch zugesprochen.

4.1.4.2 Festplatte

Die in dem Testrechner verwendeten Festplatten vom Typ ST340014A haben laut den Angaben des Herstellers folgenden Stromverbrauch:

Stromverbrauch	Durchschnitt (25 C°) in Watt
Idle	7,5
Idle mit Offline-Aktivitäten[4]	9,3
Lesen/Schreiben	12
Seeking	12,5
Standby / Sleep	0,7
Spin-Up	(unbekannt)

Tabelle 5: Energieverbrauch der Festplatte des Testrechners

Im Standby-Modus ist der Energieverbrauch somit ungefähr zwischen 10 und 17 mal geringer als während der Leerlauf- bzw. Betriebszeiten. Über den Energiebedarf beim Spin-Up macht der Hersteller keine genauen Angaben, so dass keine verlässlichen Aussagen über den Overhead, der bei der Nutzung der Standby-Zustände entsteht, getroffen werden können. Der Stromverbrauch liegt laut Hersteller bei 2,8 Ampere in der Spitze und die durchschnittlichen Dauer für die Überführung vom Standby- und Aus-Zustand in Betriebsbereitschaft beträgt zehn Sekunden. Da die Interaktivität der Benutzerprofile derart große Verzögerung nicht zulässt wird der Fokus auf die Erhöhung der Leerlaufzeit und die Vermeidung von Offline-Aktivitäten gelegt. Das eigene Power Management der Festplatten wird in sofern berücksichtigt, dass mittels des Tools „hdparm“ [97] die Aggressivität des Power Managements festgelegt wird. Dieser Wert wird sich an den Benutzerprofilen und somit an der Auslastung des Laufwerks orientieren, womit auch Synergieeffekte in Bezug auf die Systementlastung durch die Reduzierung der Festplattenzugriffe zu erwarten sind. Die Auswirkungen des Power Managements auf die Leistung der Festplatte in Form des Datendurchsatzes und der Verzögerungen beim Zugriff wird ebenfalls berücksichtigt.

Das Werkzeug „laptop-mode-tools“ [90] wurde entwickelt, um den Stromverbrauch von Festplatten zu reduzieren, wodurch insbesondere die Akkulaufzeit von Notebooks verlängert werden soll. Die Arbeitsweise dieses Werkzeugs kann aber auch auf Desktop-PCs übertragen werden. Es arbeitet sowohl im Kernel als auch im Userspace und zielt darauf ab, die Leerlaufzeit von Festplatten zu maximieren, um die Schlaf-Zustände effizienter ausnutzen zu können.

Die wichtigste Funktion von „laptop-mode-tools“ ist die Arbeitsweise mit Daten, die seit dem letzten Zugriff verändert wurden. Normalerweise schreibt der pdflush-Daemon, sofern nicht schon zuvor z.B. durch den „sync“-Befehl angefordert, Daten aus dem Page-Cache nach Ablauf von 30 Sekunden bei der nächsten Gelegenheit auf die Festplatte. Der Zeitraum kann unter „proc/sys/vm/dirty_expire_centiseconds“ festgelegt werden, was sich auch „laptop-mode-tools“ zu Nutze macht. Es verändert den Wert dahingehend, das Schreiben möglichst lange hinauszuzögern. Der Nachteil dabei ist, dass sich Risiko eines Datenverlusts mit der Verlängerung der Verzögerung proportional erhöht. Des Weiteren verzögert „laptop-mode-tools“ das c*ommit* bei Dateisystemen mit Journal, wie das in dem Testrechner verwendete ext3-Dateisystem. Bei beiden Methoden wird jeweils am Ende der aktiven Phase das Dateisystem synchronisiert, so dass es zu Beginn der Leerlaufphase keine veränderten Daten existieren. Zur Beendigung der aktiven Phase werden die Partitionen der Festplatte mit einem entsprechenden Timeout-Wert für den Wechsel in den Schlaf-Modus neu eingehängt. Das mounten der Festplatte mit dem „noatime“ - Parameter verhindert, dass bei jedem Lesen einer Datei der Zugriffszeitstempel angepasst wird, was zusätzlich einen Leistungsgewinn zur Folge hat. Diese Mount-Option kann nicht verwendet werden, wenn Programme bspw. zu Datensicherungszwecken die Option benötigen. In den in dieser Arbeit verwandten Szenarien werden jedoch keine Programme benötigt, die einen Zugriffszeitstempel für Dateien und Verzeichnisse benötigen.

[4] Offline-Aktivitäten wie S.M.A.R.T. können den Energiebedarf während der Leerlaufphase erhöhen.

Zuletzt passt „laptop-mode-tools“ die Speicherverwaltung von Linux so an, dass beim Ersetzen von Speicherseiten jene Seiten zuerst ersetzt werden, die kein Festschreiben benötigen. Dadurch werden weitere Schreibzugriffe verhindert.

Sowohl das Light-, als auch das Normal-Profil verwendet die gleiche Konfiguration (aus „/etc/laptop-mode/laptop-mode.conf“):

Parameter	Erklärung	Wert
LM_AC_MAX_LOST_WORK_SECONDS	Max. Dauer, die ungespeicherte Daten im Speicher gehalten werden.	Hoch
LM_READAHEAD	Read-Ahead aller Dateien in kB	Hoch
CONTROL_NOATIME	Noatime-Option aktivieren beim Mounten?	Ja
CONTROL_HD_IDLE_TIMEOUT	Soll „laptop mode tools“ die Leerlauf-Timeouts der Festplatte bestimmen?	Ja
LM_AC_HD_IDLE_TIMEOUT_SECONDS	Leerlauf-Timeout-Werte	Niedrig
CONTROL_HD_POWERMGMT	Soll das Festplatten Power Management durch „laptop-mode-tools“ kontrolliert werden?	Ja
LM_AC_HD_POWERMGMT	Aggressivität des Power Managements	Hoch
CONTROL_HD_WRITECACHE	Soll die Schreib-Cache über „laptop-mode- tools“ gesteuert werden?	Ja
LM_HD_WRITECACHE	Schreib-Cache aktivieren?	Ja

Tabelle 6: Laptop-Mode-Tools - Parameter

Des Weiteren werden die Logging-Aktivitäten des syslog-Daemons ([98]), in einer auf die Benutzerprofile bezogene praktikable Weise, eingeschränkt.

4.1.4.3 Interrupts / I/O-Komponenten

Zur Unterstützung des Power Managements von Hardwarekomponenten, insbesondere das der CPU, wird die Anzahl der ausgelösten Interrupts möglichst gering gehalten. Neben der tatsächlichen Aktivität, die durch den Interrupts ausgelöst ist, hat jeder IRQ zur Folge, dass die CPU diesen nur im aktiven Zustand C0 verarbeiten kann. Das halt zur Folge, dass der Prozessor für jeden IRQ aufgeweckt werden muss, wenn er sich in einem Schlafzustand befindet.

Folgende Interrupts werden in dieser Arbeit untersucht:

- Timer-IRQ (Kernel-Tick)
- ide0 & ide1: HD-IRQ (Festplatten-Interrupts)
- eth0
- LOC (Local Timer Interrupt)

Die Interrupts, die durch Tastatur- und Mausaktivitäten ausgelöst werden, können nicht berücksichtigt werden, weil durch die Simulation von Tastatur und Maus durch X-Server/Client-Kommunikation keine IRQs auftreten. Es werden außerdem keine Optimierungen in Bezug auf Netzwerk-IRQs vorgenommen, da diese schwer zu beeinflussen sind und damit den Rahmen dieser Arbeit sprengen würden. Allerdings werden sie protokolliert, um sie gegebenenfalls zur Analyse des Gesamtenergieverbrauchs heranzuziehen.

Die Timer-IRQs werden durch die Verwendung der Tickless-Kernel sowie die Vermeidung unnötiger CPU-Aktivitäten minimiert. Dazu werden im optimierten Szenario alle Anwendungen und Daemons, die nicht zur Erfüllung der in den Benutzerprofilen festgelegten Aufgaben benötigt werden, deaktiviert.

Die Festplatten-IRQs werden durch die im vorherigen Kapitel beschriebenen Maßnahmen reduziert. Auch gilt die hier Annahme, dass sich durch das Bündeln der Festplattenzugriffe die Anzahl der IRQs verringert.

Der *Local Timer Interrupt* (*LOC*) wird von Linux unter Anderem für Timer und periodische Aktivitäten verwendet. Auch hier gilt, dass je weniger vermeidbare Aktivitäten im System stattfinden, desto geringer ist die Anzahl der LOCs und desto größer ist die Wahrscheinlichkeit, die Schlafzustände der CPU zu begünstigen.

4.1.4.4 Abgrenzung

Das Starten und Herunterfahren der Systeme ist nicht Gegenstand der Optimierungen und Messungen. Durch die hohe Beanspruchung der Ressourcen, insbesondere beim Hochfahren, ist offensichtlich, dass der Energieverbrauch dieser Prozesse im Wesentlichen von dessen Dauer und den zu initialisierenden Hardware-Komponenten abhängt. Eine Verkürzung der Dauer des Hochfahrens, beispielsweise durch einen minimalen Kernel, der nur die unmittelbar notwendigen Module und Treiber lädt, ist daher auch eine Optimierung bezüglich der Energieeffizienz. Deshalb ist davon auszugehen, dass das optimierte Szenario I diesbezüglich auch das energieeffizienteste ist. Da die Dauer des Hoch- und Herunterfahrens des Systems im Vergleich zur Gesamtlaufzeit des Systems nur einen geringen Anteil ausmacht und zusätzliche Analyse des Start- und Herunterfahren auf Grund fehlender Daten, die nicht vom Linux-System nicht zur Verfügung gestellt werden, den Rahmen dieser Arbeit überschreiten würde, werden diese Vorgänge nicht berücksichtigt. Die Messungen beginnen also erst, nach dem das System vollständig gestartet und betriebsbereit ist.

Des Weiteren werden thermische Aspekte und deren Wechselwirkung mit dem Energiebedarf eines Systems nicht berücksichtigt. Der im Testsystem zur aktiven Kühlung der CPU verwendete Lüfter wird unter Volllast betrieben, um maximale Kühlung zu gewährleisten und den Energiebedarf konstant zu halten. Dadurch ergeben sich Anknüpfungsmöglichkeiten für weitere Untersuchungen hinsichtlich energiesparsamer Lüftersteuerung unter Berücksichtigung der mit der Wärmeentwicklung in unmittelbarem Zusammenhang stehenden CPU-Auslastung.

Da in dieser Arbeit der Energieverbrauch zur Laufzeit des Systems im Fokus steht, werden die Zustände S1-S4, wie systemweites Sleep oder Ruhezustand, nicht berücksichtigt.

4.1.4.5 Gewährleistung des Bedienkomforts

Es muss sichergestellt werden, dass der Benutzer des Systems durch die Optimierung keine derart großen Einbußen in der Bedienbarkeit hinnehmen muss, dass er bei der Erledigung seiner Aufgaben behindert wird. Der Grad der Bedienbarkeit wird an Hand von potenziellen Flaschenhälsen festgestellt. Folgende Daten werden zur Erkennung von Flaschenhälsen herangezogen und periodische jede Sekunde ermittelt:

- Länge der Run-Queue der CPU (die Anzahl der Prozesse, die darauf warten, eine Zeitscheibe zugeteilt zu bekommen)
- CPU-Auslastung (User, Nice, System, I/O-Wait, Steal, Idle)
- Anzahl der Swap-Pages pro Sekunde (Page-in (Disk → RAM) und Page-Out (RAM → Disk))
- Sehr hoher Datendurchsatz der Festplatte

Des Weiteren wird der Einfluss auf die in dem rechenintensiveren Medium-Benutzerprofile verwendeten Hintergrundaktivitäten, wie das Kompilieren von Quellcode und das Kopieren von

Dateien, gemessen. Interessant ist hierbei, ob sich die Dauer dieser Operationen maßgeblich verlängert.

4.2 Realisierung

4.2.1 Testsuiten

4.2.1.1 Benutzer-Simulation

Die Benutzerprofile werden durch automatisierte Abläufe durch das Makro-Softwarepaket Xnee [99] realisiert. Diese Software ist in der Lage, die Kommunikation zwischen X-Server und X-Client, wie beispielsweise Tastatur- und Mausbefehle, aufzuzeichnen und an Hand der aufgezeichneten Daten Makros zu erstellen. Die Makros simulieren dann den Benutzer durch das Ausführen von Maus- und Tastaturbefehle. Dadurch ist sichergestellt, dass in allen Szenarien die exakt gleiche Befehlsfolge ausgeführt wird.

Jedes Benutzerprofil wird durch ein eigenes Skripts realisiert, dass neben der Steuerung der Xnee-Software Daten aus dem „proc“-Subsystem, wie beispielsweise die CPU-Auslastung und die Verweildauer in den einzelnen C-States, sammelt, welche die Basis für die Evaluation der Messdaten aus den verschiedenen Szenarien bilden. Dieses Skript besteht wiederum aus Unterskripten, die die einzelnen Arbeitsschritte simulieren, beispielsweise das Schreiben eines Briefes mit OpenOffice Writer. Die Simulation behält dabei die Arbeitsgeschwindigkeit des Benutzer bei. Die statistischen Daten werden jeweils am Start und am Ende der Unterskripte gesammelt, um neben der Gesamtlaufzeit des Szenario auch die Dauer der Arbeitsschritte der einzelnen Szenarien miteinander vergleichen zu können. Dadurch erhalten die Messdaten eine feinere Granularität, was der Analyse mehr Möglichkeiten einräumt. Die gesamte Simulation dauert bei beiden Szenarien etwa 20 Minuten. In den Profilen werden auch Anwendungen wiederholt ausgeführt, um das Verhalten des Cachings mit in die Beobachtungen mit einbeziehen zu können. Jede Simulation wird drei Mal ausgeführt und aus den gewonnen Daten wird der Mittelwert ermittelt. Nach jedem Testlauf wird der Rechner neu gestartet, um Seiteneffekte durch Caching und Ähnlichem auszuschließen. Die Skripte befinden sich im Anhang B, die aus den Testläufen gewonnen Daten befinden sich auf der beiliegenden CD.

Da die Kommunikation im X-Window-System durch zusätzliche Software simuliert werden muss und die Makros über Skripte gesteuert werden, entsteht ein Overhead, der allerdings zu gering ist, um die Messergebnisse signifikant zu beeinflussen. Die Auswirkungen des Overheads wurde mit dem System-Profiling-Tool „oprofile“ ([100]) untersucht. Dieses Werkzeug erstellt unter Verwendung der Hardware-Performance-Counter-Register der CPU Statistiken über verschiedene Daten, wie z.B. die Anzahl der benutzen CPU-Zyklen oder die Anzahl der ausgelösten Interrupts pro Prozess.

Bei der Simulation der Benutzer treten zahlreiche Probleme auf. Die Konfiguration des Gentoo- und des OpenSuse-Systems, die für die Szenarien verwendet werden, sind nach der Installation zunächst sehr unterschiedlich, so dass Systemsprache und Erscheinungsbild aufeinander abgestimmt werden müssen. Außerdem muss die Reaktionszeit der Anwendungen berücksichtigt werden, um die Sychronisität zu gewährleisten, so dass die Feinabstimmung der Skripte sehr viel Zeit in Anspruch nimmt.

4.2.1.2 System-Setup

Die 40 GB Festplatte wurde wie folgt partitioniert:

Device	Funktion	Dateisystem	Größe
/dev/sda1	Boot-Partition	ext2	128 MB
/dev/sda2	Swap-Partition	swap	2048 MB
/dev/sda3	Root-Partition	ext3	37,875 GB

Tabelle 7: Festplatten-Partitionierung des Testrechners

Auf Grund der Größe des Arbeitsspeichers von 1024 MB kann die Swap-Partition im Light-User-Profile des ersten Szenarios deaktiviert werden. Zur Erhöhung der Datensicherheit wird für die Root-Partition (/) das Journal-Dateisystem ext3 verwendet.

Als Gerätetreiber wird jeweils die Open Source – Variante verwendet. Alle nicht benötigten Hardware-Komponenten wurden entfernt bzw. falls kein Entfernen möglich ist, im BIOS deaktiviert.

Da das BIOS des Testrechners keine Informationen über die C-States der CPU zur Verfügung stellt, müssen diese über eine veränderte DSDT-Tabelle bereitgestellt werden. Für den C-Stand C1 wurde dabei ein Energieverbrauch von 0,75 Watt und für den Zustand C2 0,5 Watt angegeben. Der Verbrauch im aktiven Zustand C0 ist typischerweise stark schwankend und konnte nicht genau ermittelt werden. Da der Energieverbrauch des gesamten Systems zwischen 55 Watt und 130 Watt schwankt, wird angenommen, dass die CPU je nach Auslastung zwischen 50 und 100 Watt verbraucht, was zwar nur eine sehr grobe Einschätzung ist, aber die die Zwecke dieser Arbeit ausreicht.

4.2.2 Messinfrastruktur

Die Messergebnisse werden zum Einen durch die vom System zur Verfügung gestellten Daten wie z.B: CPU-Auslastung, ausgelöste und bearbeitete Interrupts und Festplattenzugriffe gewonnen, zum Anderen wird der Gesamtverbrauch des Systems über den Zeitraum der Versuche mittels eines digitalen Leistungsmesser (Wattmeter) mit einer Genauigkeit von 0,1 Watt ermittelt.

4.2.2.1 CPU

Über das proc-Dateisystem werden Informationen über die C-States den Prozessoren zur Verfügung gestellt. Die Informationen enthalten Angaben darüber, welche C-States von der CPU unterstützt werden, welcher Zustand der Default-Zustand ist, in den bei Inaktivität gewechselt wird, und wie groß die entstehende Verzögerung ist, die durch die Zeit vom Auftreten eines Interrupts bis zum Beginn dessen Bearbeitung bestimmt ist. Für die Auswertung der Szenarien sind vor allem die statistischen Angaben über die Anzahl der Zustandswechsel sowie die Verweildauer in den einzelnen Zuständen von Bedeutung. Diese Informationen finden sich unter „/proc/acpi/prozessor/CPU0/power“.

Die Effizienz des „tickless Kernels“ kann unter „/proc/timer_list“ untersucht werden. Über diese Schnittstelle werden Angaben über die Anzahl der „Idle Calls“ insgesamt („idle_calls“), also die Zeit, in der der Kernel-Tick ausgesetzt werden soll, die Anzahl der „Idle Calls“, die dann auch tatsächlich zu einem Stopp des Kernel-Ticks führten (idle_sleeps) sowie die Gesamtdauer der Leerlauf-Zeit in Nanosekunden (idle_sleeptime) zur Verfügung gestellt.

Daten über die CPU-Auslastung werden an Hand der Daten, die von dem proc-Dateisystem unter „/proc/stats“ zur Verfügung gestellten werden, erhoben. Folgende Informationen werden zur Analyse herangezogen:

- User: Normale Prozesse, die im User-Mode (Userspace) ausgeführt werden
- System: Prozesse, die im Kernel-Mode (Kernelspace) ausgeführt werden
- Idle: Kernel-Ticks während des Leerlauf-Modus insgesamt
- I/O-Wait: Dauer des Wartens auf die Beendigung von I/O-Operationen.

Zu Protokollierungszwecken wird das Werkzeug „sysstats“ ([101] verwendet, dass es ermöglicht, den aktuellen P-State der CPU auf einfache Weise periodisch im Sekundentakt zu ermitteln.

4.2.2.2 Festplatten

Analog zur CPU werden Daten über die Festplattenzugriffe über das proc-Dateisystem zur Verfügung gestellt. Folgende, unter „/proc/diskstats“ abrufbare Informationen, werden zur Auswertung herangezogen:

- jeweils die Anzahl der Lese- und Schreiboperationen insgesamt
- jeweils die Anzahl der zusammengefügten Lese- und Schreiboperationen
- jeweils die Dauer der Lese- und Schreiboperationen insgesamt (in Millisekunden)
- benötigte Zeit für I/O-Operationen insgesamt
- gewichtete Gesamtdauer der I/O-Operationen

Ein weiterer Indikator für die Effizienz der Szenarien ist die Anzahl der von der Festplatte ausgelösten I/O-Interrupts, auf die im folgenden Abschnitt genauer eingegangen wird.

4.2.2.3 Interrupts

Die Anzahl der ausgelösten Interrupts wird für jede Interrupt-Quelle unter „/proc/interrupts“ aufgelistet, so dass die relevanten Interrupt-Requests der Szenarien einfach verglichen werden können. Für die Evaluation wird die Anzahl der Timer-IRQs, Festplatten-IRQs, Ethernet-IRQs und Local Timer IRQs ermittelt.

4.2.2.4 Gesamtenergieverbrauch

Der Energieverbrauch des Systems wird durch das in Kapitel 4.1.1.2 kurz beschriebene Messgerät ermittelt, dass direkt zwischen Netzanschluss und Testsystem angeschlossen ist. Die gemessenen Werte werden in den Speicher des Geräts geschrieben und können über eine USB-Schnittstelle ausgelesen werden. Für die Auswertungen der Szenarien wird die Wirkleistung (in W) und die Wirkenergie (in Wh) mit einer Genauigkeit von 0.1 Watt gemessen.

4.2.2.5 Gewährleistung des Bedienkomforts

Mit Hilfe des Werkzeug „sysstats“ [101] werden die im Kapitel 4.1.4 vorgestellten Daten ermittelt. Dieses Programm läuft parallel und unabhängig von dem Simulationsskript. Im Einzelnen werden folgende Daten zur Analyse herangezogen:

- Länge der Run-Queue der CPU (die Anzahl der Prozesse, die darauf warten, von dem Scheduler eine Zeitscheibe zugeteilt zu bekommen)
- CPU-Auslastung (User, Nice, System, I/O-Wait, Steal, Idle)
- Anzahl der Swap-Vorgänge pro Sekunde (Page-in (Disk → RAM) und Page-Out (RAM → Disk))
- Datendurchsatz der Festplatte

4.2.3 Implementierung

4.2.3.1 Kernel-Setup

Der Konfiguration des Kernels liegt das Minimalprinzip zu Grunde. Nicht benötigte Treiber werden entfernt und alles Weitere, soweit möglich, direkt in das Kernel-Image und nicht als Modul kompiliert. Außerdem sind alle Debug-Optionen deaktiviert. Die Tickless-Kernel-Option ist aktiviert und die Timer-Frequenz auf 100 Hz, den niedrigsten Wert, eingestellt. Die niedrige Frequenz ist für Systeme, die für die relativ hohe Latenzzeiten akzeptabel sind, vertretbar, so dass für die gewählten Szenarien 100 Hz ausreichend sind. Der Group-CPU Scheduler ist deaktiviert, um zu verhindern, dass bestimmte Gruppen von Prozessen bevorzugt behandelt werden. Außerdem wird der I/O-Scheduler CFQ (Completely Fair Queing) gewählt, der die Bandbreite gleichermaßen zwischen allen Prozessen verteilt und sich deshalb für Desktop-Systeme eignet, dessen Workload schwieriger vorauszusehen ist, als beispielsweise der Workload eines Stapelverarbeitungs-Systems mit festen, wiederkehrenden Aufgaben. Als Preemption Modell wird für die Light-Szenarien die Option „Server“ (PREEMT_NONE) verwendet. Dies kann dazu führen, dass die Abläufe im System für den Benutzer weniger gleichmäßig erscheinen, also teilweise mit Verzögerungen zu rechnen ist, da Prozesse mit niedrigerer Priorität stärker benachteiligt werden. Diese Einstellung eignet sich vor allem für Serversysteme, da die Performanz gesteigert wird. Für das Light-Profil ist diese Einstellung jedoch ebenso zulässig, da keine besonders hohen Ansprüche an die Reaktionszeit des System gelten bzw. zu erwarten ist, dass die Reaktionszeit auf Grund der insgesamt niedrigen Systemauslastung nicht sehr stark beeinflusst wird. Beim Profil des Normal-Users wird als Preemption-Modell „Desktop“ gewählt, da durch die höherer Systemauslastung die Verzögerungszeiten ansteigen und der Benutzer höhere Ansprüche an die Reaktivität des Systems hat.

Im Bereich des Power Management / ACPI werden alle Optionen, bis auf diejenigen, die nur zu Debugging-Zwecken geeignet sind, aktiviert. Insbesondere der P-States-Treiber und alle dazugehörigen Governor werden aktiviert.

Die komplette Konfiguration befindet sich auf der beiliegenden CD.

4.2.3.2 CPUFREQ

Das CPUFREQ-Subsystem beinhaltet standardmäßig vier verschiedene Governors, die auch in den Gentoo-Szenarien alle zum Einsatz kommen. Die Governor entscheiden, mit welcher Frequenz die CPU über einen bestimmten Zeitraum betrieben wird. Welche Frequenz-Stufen verfügbar sind wird durch die Hardware festgelegt. Die in dieser Arbeit verwendete CPU unterstützt die Taktraten 1.00 GHz, 1.80 GHz und 2.00 GHz. Die OpenSuse-Szenarien verwenden den „Ondemand“-Governor, da dieser bei der Installation des Betriebssystems bereits standardmäßig aktiviert wird. Im Folgenden wird die, der Linux Kernel Dokumentation entnommenen, Funktionsweise der vier Governor erläutert [86]:

Der „Powersave“- und der „Performance“-Governor funktionieren auf ähnliche Weise. Sie legen die CPU-Frequenz auf den niedrigsten bzw. höchsten Wert innerhalb des einstellbaren Bereichs zwischen 1.00 GHz und 2.00 GHz fest. Die Dokumentation des Kernels steht damit, wie später gezeigt wird, im Widerspruch zu den Messergebnissen, da trotz Powersave-Governor die CPU auch andere Frequenzen als 1.00 GHz nutze, wenn auch in geringerem Maße als die anderen Governor.

Mit dem „Ondemand“-Governor wird die CPU-Frequenz dynamisch, in Abhängigkeit von der CPU-Auslastung, angepasst. Der Parameter „sampling_rate“ gibt an, in welchen Abständen der Kernel die CPU-Auslastung abfragt. Typischerweise liegt der Wert bei 10 ms oder mehr. Die Gentoo-Szenarien verwenden den Standardwert von 200 ms, die OpenSuse-Szenarien benutzen den für OpenSuse üblichen Wert von 107 ms. Höhere Abtastraten verbessern hierbei nicht notwendigerweise die Energieeffizienz, da sich häufiges Polling wiederum negativ auf dass Power Management der CPU auswirkt. Des Weiteren kann ein Grenzwert bestimmt werden, der festlegt, ab welcher Auslastung die

CPU-Frequenz auf den höchsten Wert gesetzt wird. Die Testsysteme verwenden einen Wert von 95% Auslastung in den Gentoo-Szenarien und 80 % Auslastung in den OpenSuse-Szenarien.

Die Arbeitsweise des „Conservative“-Governor ähnelt dem „Ondemand“-Governor, nur ändert dieser die CPU-Frequenz weniger abrupt, sondern erhöht diese um einen festlegbaren Prozentsatz, der in allen Testsystemen bei dem Standardwert von 5% liegt. Zusätzlich kann ein Grenzwert für das Reduzieren der CPU-Frequenz definiert werden, der die CPU-Auslastung angibt und standardmäßig bei 20% liegt. Dieser Wert wird für alle Testsysteme ebenfalls unverändert übernommen.

4.2.4 Evaluation

4.2.4.1 Messergebnisse

An Hand des Gentoo- und des OpenSuse-Szenarios wurden sechs Testszenarien entworfen. Des sechste Szenario besteht aus dem OpenSuse-System, das allerdings den optimierten Kernel aus den Gentoo-Szenarien verwendet:

	1	**2**	**3**	**4**	**5**	**6**
Betriebssysteme	Gentoo	Gentoo	Gentoo	Gentoo	OpenSuse 11.1	OpenSuse 11.1
Kernel	2.6.29-r5	2.6.29-r5	2.6.29-r5	2.6.29-r5	2.6.27-r25	2.6.29-r5
P-States Governor	Ondemand	Powersave	Conservative	Performance	Ondemand	Ondemand

Tabelle 8: Mess-Szenarien

CPU

Im Folgenden wird die CPU-Statistik für P-States, C-States und dessen Auswirkungen auf den Tickless Kernel sowie die CPU-Auslastung dargestellt und analysiert.

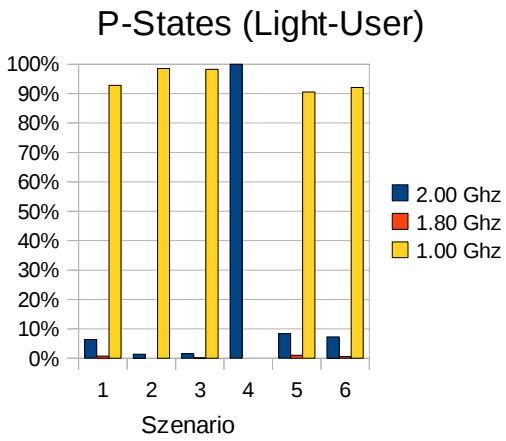

Abbildung 14: P-States (Light-User)

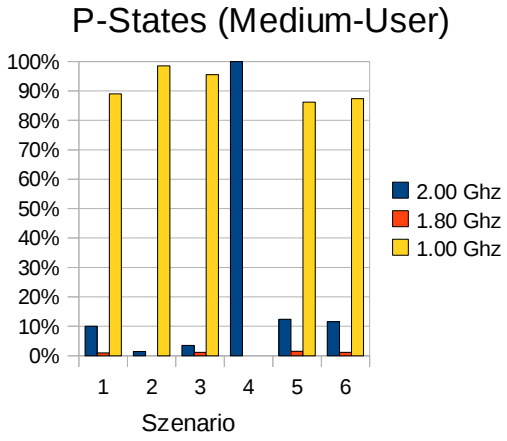

Abbildung 15: P-States (Medium-User)

Abbildung 14 und Abbildung 15 zeigen den prozentualen Anteil der Dauer, in welchem P-State sich die CPU während der Testläufe befunden hat. Das zweite Szenario, dass den „Powersave“-Governor verwendet, weist hierbei den günstigen Verlauf auf, dicht gefolgt von dem ähnlich funktionierenden „Conservative“-Governor verwendenden dritten Szenario. Die beiden Benutzerprofile weisen die

gleichen Charakteristiken auf, wobei die P-States im „Light-User“-Profil gegenüber dem „Medium-User“-Profil auf Grund der geringeren Systemauslastung besser ausgenutzt werden können.

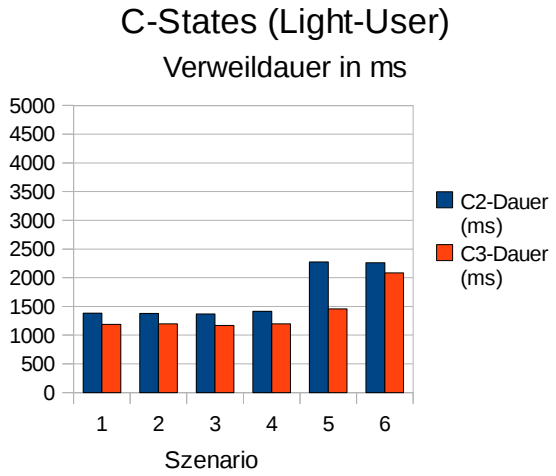

Abbildung 16: C-States (Light-User)

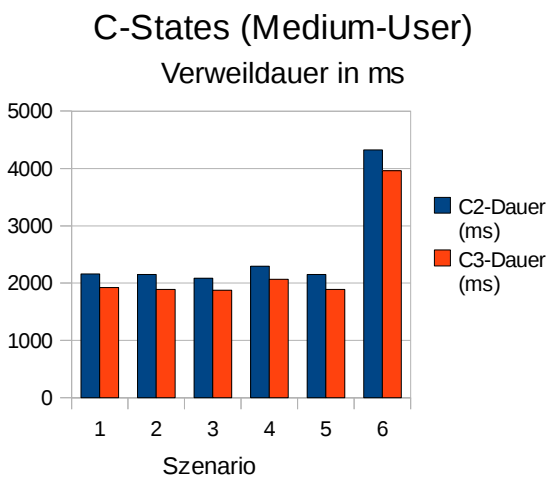

Abbildung 17: C-States (Medium-User)

Die Verweildauer in den C-States C2 und C3 wird in Abbildung 16 und Abbildung 17 dargestellt. Zwischen den Benutzerprofilen werden einige Unterschiede deutlich. Während sich der P-States-Governor bei dem „Light-User“-Profil kaum auf C-State-Nutzung auswirkt, werden bei höherer Systemauslastung stärkere Unterschiede zwischen den Policies deutlich. Mit dem „Powersave“-Governor wird die Effizienz der C-States verschlechtert, da dieser durch die Minimierung der CPU-Frequenz die Leerlauf-Zeit verringert, weil die zu erledigenden Aufgaben mehr Zeit in Anspruch nehmen. Der „Ondemand“-Governor weist dagegen bei viel besserer P-State-Auslastung eine fast genauso gute C-States-Auslastung wie der „Performance“-Governor auf, so dass anzunehmen ist, dass sich der „Ondemand“-Governor tatsächlich sehr gut an die Systemauslastung anpasst. Insgesamt ist zu beobachten, dass sich die C-State-Effizienz beim „Normal“-User, mit Ausnahme des „Powersave“-Governor im Gentoo-Szenario und dem optimiertem OpenSuse-Szenario, fast verdoppelt. Die beste C-State-Bilanz hat der „Ondemand“-Governor mit dem OpenSuse-Szenario.

Im Allgemeinen ist festzustellen, dass sich die CPU zwar je nach Szenario und Profil unterschiedlich lange in einem Schlafzustand befindet, die Dauer aber auch im besten Fall deutlich unter 1% des Gesamtdauer bleibt, während energiesparsame P-States für rund 90% der Laufzeit genutzt werden. Des Weiteren kann beobachtet werden, dass sich eine aggressive Nutzung der P-States zwar tendenziell negativ auf die C-State-Verwaltung auswirkt, die Auswirkungen aber nicht signifikant sind, so dass anzunehmen ist, dass eine effiziente P-States-Verwaltung mehr Einfluss auf den Energiebedarf der CPU hat als die Nutzung von C-States. Weitere Messungen ergaben, dass der Energieverbrauch des Gesamtsystems durch eine Reduzierung der CPU-Taktrate von 2.00 GHz auf 1.00 GHz, unabhängig von dem verwendeten Szenario, um ca. 20 Watt gesenkt wird. Ausgehend von der zuvor getroffene Annahme über den Energieverbrauch der CPU in den verschiedenen C-States mit Werten zwischen 0,5 und 100 Watt wird die These bestätigt, dass der Energiespareffekt maßgeblich durch die P-States beeinflusst wird, da die längere Verweildauer in den P-States den, im Vergleich zu den C-States, höheren Energiebedarf mehr als kompensiert.

Die P-State-Policies haben ebenfalls keine signifikanten Auswirkungen auf die Länge des Timer-Sleeps, also der Zeit, in der der Kernel-Tick ausgesetzt wird. Die Messungen ergaben, dass der Kernel-Tick zu ca. 90% inaktiv ist, abhängig vom verwendeten Profil und Szenario, was für eine hohe Effizienz des Tickless-Kernels spricht.

Um sicherzustellen, dass durch die Optimierungen nicht der Benutzerkomfort in inakzeptabler Weise eingeschränkt wird, wird während der Tests die CPU-Auslastung in Form der durchschnittlichen Länge der Run-Queue beobachtet und durch Abbildung 18 und Abbildung 19 sowie Tabelle 9

dargestellt. Zunächst wird deutlich, dass je mehr das System auf geringen Energieverbrauch ausgelegt ist, desto höher die Beanspruchung der CPU ist. Dies hängt damit zusammen, dass die CPU-Frequenz an den Workload angepasst wird, so dass Tasks und Prozesse entsprechend mehr CPU-Zeit brauchen, da deren Bearbeitung mehr Zeit in Anspruch nimmt. Das verwendete Benutzerprofil wirkt sich auf die meisten Governor recht ähnlich aus. Lediglich bei dem „Powersave“-Governor sowie dem „Conservative“-Governor gibt es kleine Unterschiede. Eine entscheidende Beobachtung ist, dass der optimierte Kernel in der OpenSuse-Umgebung, wie gezeigt, bessere Eigenschaften hinsichtlich des Energieverbrauchs aufweist und dennoch die CPU des Testsystem weniger beansprucht. Insgesamt führen die Optimierungen zu keiner kritischen Verschlechterung der Systemperformanz.

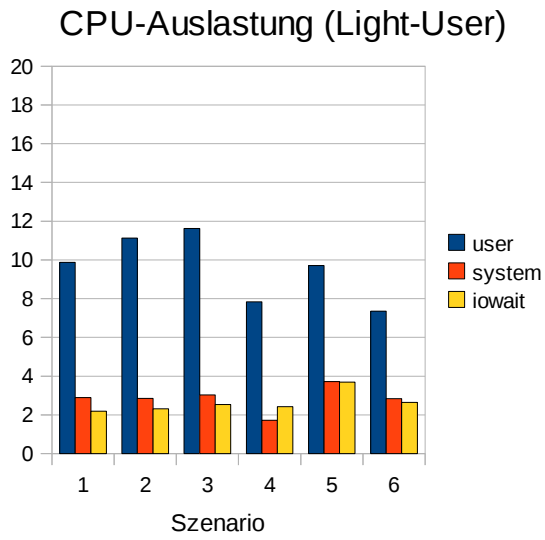

Abbildung 18: CPU-Auslastung (Light-User)

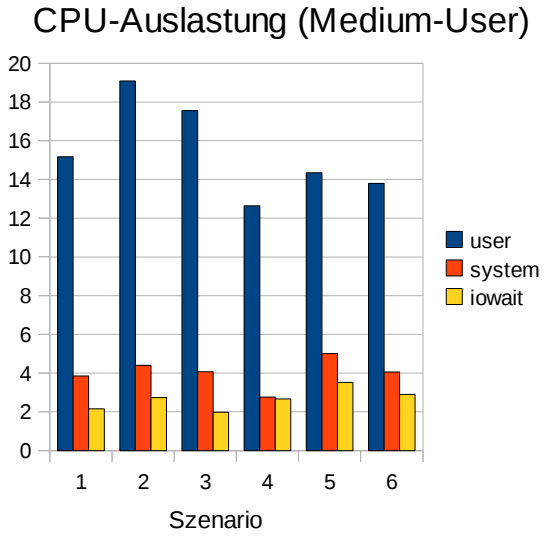

Abbildung 19: CPU-Auslastung (Normal-User)

Profil	Szenario 1	Szenario 2	Szenario 3	Szenario 4	Szenario 5	Szenario 6
Light	0,20	0,25	0,23	0,19	0,28	0,26
Medium	0,33	0,43	0,35	0,29	0,60	0,49

Tabelle 9: CPU-Load

Des Weiteren wurde untersucht, ob die Optimierungen zu Flaschenhälsen führen, die der Benutzer beispielsweise in Form eines nicht oder nur sehr träge reagierenden Systems wahrnimmt. Dazu wurde analysiert, über welchen Zeitraum die CPU in einem Intervall von einer Sekunde mehr als 90% bzw. 75% ausgelastet ist (Abbildung 20, Abbildung 21, Abbildung 22 und Abbildung 23).

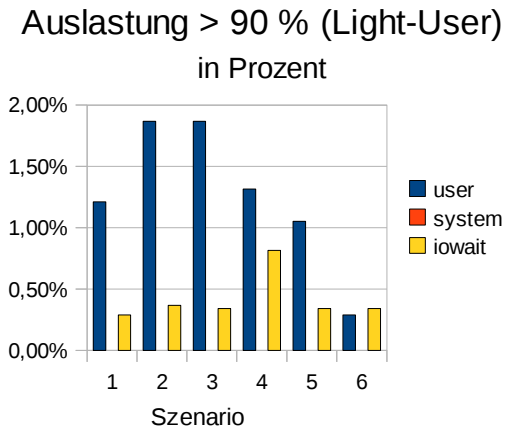

Abbildung 20: CPU-Auslastung > 90% (Light-User)

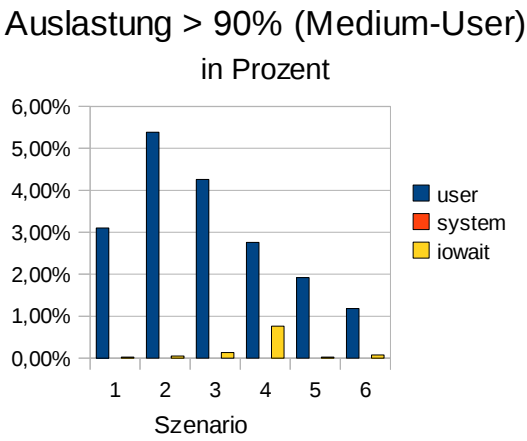

Abbildung 21: CPU-Auslastung > 90% (Normal-User)

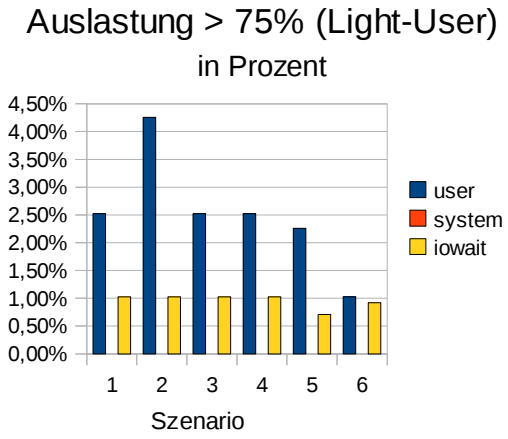

Abbildung 22: CPU-Auslastung > 75% (Light-User)

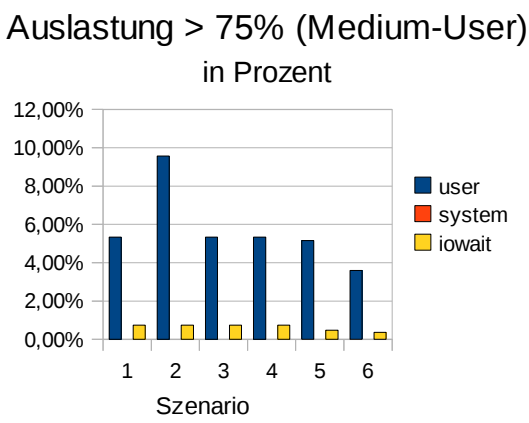

Abbildung 23: CPU-Auslastung > 75% (Medium-User)

Auch hier zeigt der „Powersave“-Governor in allen Szenarien die mit Abstand schlechtesten Werte, die sich allerdings alle in einem Bereich mit maximal gut 10% CPU-Auslastung befinden, was für die Reaktivität des Systems eigenen Beobachtungen zufolge als akzeptablen gilt. Durch die Steuerung der Testläufe mit Tastatur- und Mausmakros kann geprüft werden, ob die Aufgaben in einem erwarteten Zeitraum erledigt werden. Hohe Reaktionszeiten des Systems würden dazu führen, dass die Makros nicht fehlerfrei ausgeführt werden und die Makrosteuerung nicht mehr synchron läuft, womit der Testlauf insgesamt fehlgeschlagen wäre, was jedoch nicht aufgetreten ist.

Um die Auswirkungen des Cachings zu analysieren, wird jeder Testlauf zwei Mal hintereinander durchlaufen. Die These, dass sich die Verweildauer in den C-States bei erhöhter Systemauslastung verlängert, wird auch hier gestützt, da in nahezu allen Szenarien und Profilen die Schlafzeit beim zweiten Durchlauf zwischen 1% und 16% kürzer ausfällt. Auf den Timer-Sleep wirkt sich das Caching leicht positiv aus, so dass der Timer-Tick bis zu 5 % länger ausgesetzt werden kann (Tabelle 10 und Tabelle 11). Auffällig sind auch die starken Unterschiede zwischen den Benutzerprofilen.

Szenario	**1**	**2**	**3**	**4**	**5**	**6**
C2-Dauer	-11,55%	-13,70%	-9,91%	-16,03%	-1,40%	-2,78%
C3-Dauer	-9,57%	-12,28%	-8,55%	-15,36%	0,76%	-1,76%
Timer-Sleep	4,01%	3,54%	2,61%	3,41%	5,16%	5,29%

Tabelle 10: CPU-Statistik: Veränderung im 2. Durchlauf (Light-User)

Szenario	**1**	**2**	**3**	**4**	**5**	**6**
C2-Dauer	-7,19%	-15,26%	-6,20%	-6,51%	-15,26%	-1,31%
C3-Dauer	-6,04%	-10,99%	-4,44%	-5,52%	-10,99%	1,57%
Timer-Sleep	4,69%	4,45%	3,96%	2,92%	4,44%	4,04%

Tabelle 11: CPU-Statistik: Veränderung im 2. Durchlauf (Medium-User)

Festplatte

Die Auswirkungen der CPU-Policies auf das Power Management der Festplatte konnten nicht festgestellt werden, da die Messergebnisse zu starken Schwankungen unterlagen und andere Einflussfaktoren nicht ausgeschlossen werden konnten. Um aussagekräftige Ergebnisse zu erhalten, hätte der Energieverbrauch der Festplatte aufzeichnet werden müssen, was allerdings den Rahmen der Arbeit überschritten hätte. Die Ergebnisse befinden sich im Anhang C.3 und C.4 .

Abbildung 24 und Abbildung 25 zeigen drei ausgewählte Szenarien: Das sich bisher als am energieeffizientesten herausgestellte Szenario „Gentoo mit „Powersave“-Governor“ sowie die beiden

OpenSuse-Systeme, einmal mit Standardkernel und einmal mit optimierten Kernel. Hier wird ersichtlich, dass sich das minimale Gentoo-System nur bei den Schreibzugriffen positiv auswirkt und der optimierte Kernel im OpenSuse-System keine wesentlichen Auswirkungen hat. Die Anzahl der Lesezugriffe ist wieder zu stark schwankend, um eine genaue Aussage zu treffen.

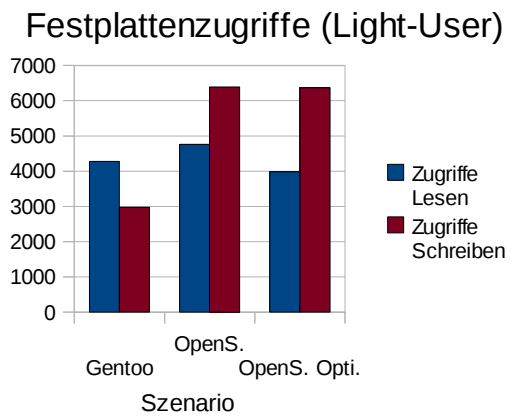

Abbildung 24: Festplattenzugriffe (Light-User)

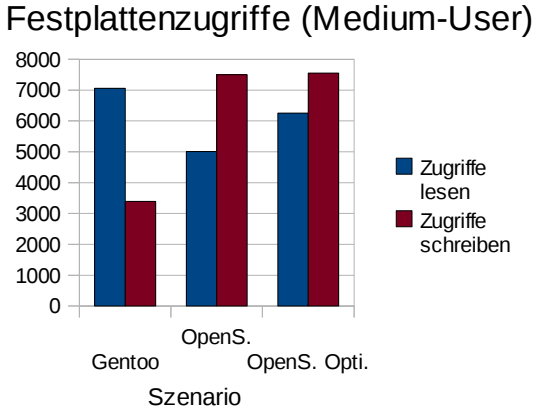

Abbildung 25: Festplattenzugriffe (Medium-User)

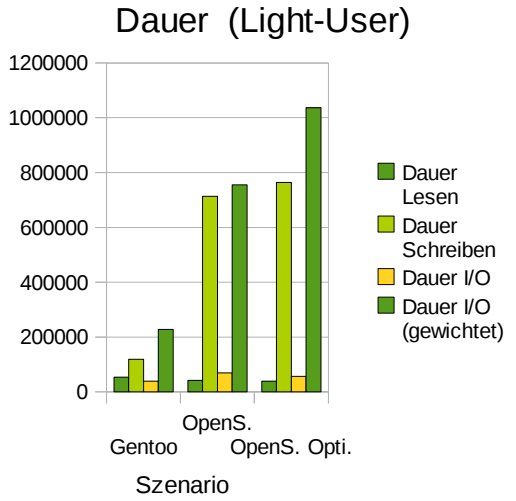

Abbildung 26: Gesamtdauer der Festplattenoperationen (Light-User)

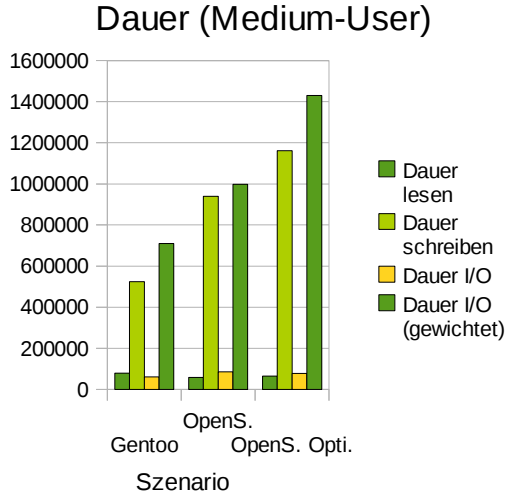

Abbildung 27: Gesamtdauer der Festplattenoperationen (Medium-User)

Hinsichtlich der Gesamtdauer der Festplattenoperationen verhält sich das Gentoo-System günstiger gegenüber dem OpenSuse-System. Allerdings ist zu beobachten, dass der optimierte Kernel mit OpenSuse ungünstigere Werte liefert (siehe Abbildung 26 und Abbildung 27).

Das Caching wirkt sich im Allgemeinen positiv auf die Anzahl und Dauer der Lesezugriffe aus, so dass im zweiten Testdurchlauf die Messergebnisse der lesenden Festplattenzugriffe um ca. 90% - 100% geringer ausfielen. Zwischen den Szenarien konnten keine eindeutigen Unterschiede festgestellt werden. Die Anzahl der Schreibzugriffe ist in allen Szenarien im zweiten Durchlauf höher als im ersten, wobei die Gesamtdauer der Schreiboperationen in der Regel geringer ist. Eine Ausnahme bildetet dabei das vierte Szenario (Gentoo, „Performance“-Governor“) mit Medium-Benutzerprofil. Die Daten schwanken hier allerdings stark zwischen den ersten beiden Gentoo-Szenarien mit Werten um zwischen 65% und 80% und dem fünften Szenario (OpenSuse, nicht optimierter Kernel) mit 4%. Zusätzlich ist zu beobachten, dass sich die Cache-Effizienz bei dem OpenSuse-System durch die Verwendung des optimierten Kernels tendenziell verbessert, wobei jedoch die Werte zu stark schwanken, um eindeutige Aussagen treffen zu können. Die Ergebnisse befinden sich in tabellarischer Form im Anhang C.3 und C.4 .

Um die Benutzerakzeptanz bezüglich der Reaktivität des System zu gewährleisten, wird die Länge der Run-Queue des I/O-Schedulers und die durch I/O-Anfragen verursachte CPU-Zeit analysiert. Dabei ist festzustellen, dass das optimierte Gentoo sowohl in der Zeit, in der die Länge Run-Queue größer als eins ist, als auch in der benötigten CPU-Zeit in allen Szenarien günstiger ist als das nicht optimierte OpenSuse-System. Dies ist darauf zurückzuführen, dass im Zuge der Optimierung hinsichtlich der Energieeffizienz auch die Anzahl der I/O-Zugriffe reduziert wird. Die detaillierten Messdaten befinden sich im Anhang C.9 und C.10 .

Interrupts

Ein Optimierungsziel ist die Reduzierung von unnötigen Interrupts. In der Gegenüberstellung der insgesamt ausgelösten Interrupts (Abbildung 29 und Abbildung 28) zeigen beide Benutzerprofile die gleichen Charakteristika auf. Während sich der P-State-Governor erwartungsgemäß nur unwesentlich auswirkt, wird deutlich, dass die minimierten Gentoo-Szenarien deutlich weniger Interrupts verarbeiten müssen. Ebenso wirkt sich der optimierte Kernel positiv auf das OpenSuse-System aus. Dies ist unter Anderem darauf zurückzuführen, dass im optimierten Kernel alle nicht benötigten Hardware-Komponenten deaktiviert sind. Eine detaillierte Auflistung der Messergebnisse, die auch den Typ des Interrupts berücksichtigt, befindet sich in Kapitel C.5 und C.6 . Hinsichtlich der Reaktivität des System wurde festgestellt, dass keine der Optimierungen dazu führt, dass Interrupts nicht oder nicht rechtzeitig verarbeitet werden konnten.

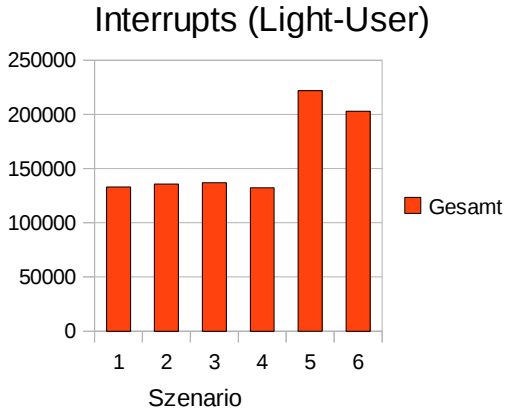

Abbildung 28: Interrupts (Light-User)

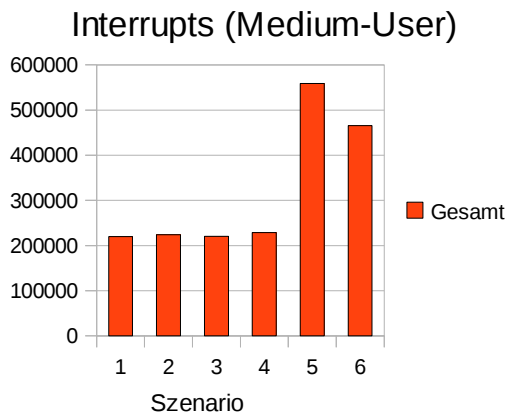

Abbildung 29: Interrupts (Medium-User)

Energieverbrauch

Um die Auswirkungen der Optimierungen auf den tatsächlichen Energieverbrauch des Testrechners zu untersuchen, wird für bestimmte Szenarien der Energieverbrauch für jeweils beide Benutzerprofile, „light“ und „medium“, gemessen. Als Szenarien wurde das auf theoretischer Basis energieeffizienteste Szenario (Gentoo mit „Powersave“-Governor), das am wenigsten energieeffiziente Szenario (OpenSuse mit Standard-Kernel) sowie das OpenSuse-Szenario mit optimiertem Kernel ausgewählt.

Szenarien	Gentoo light	OpenSuse light		Gentoo medium	OpenSuse medium	
	opt.	normal	opt.	opt.	normal	opt.
Verbrauch je Szenario (W)	22,1	23,3	23,1	23,7	25,9	25,1
Tagesverbrauch (W) [5]	501,56	528,79	524,26	528,71	577,78	559,94
Jahresverbrauch (kW) [6]	100,31	105,76	104,85	105,74	115,56	111,99

Tabelle 12: Energie-Messergebnisse

Tabelle 12 zeigt die Ergebnisse der Messungen des Energieverbrauchs des gesamten PC-Systems sowie eine Hochrechnung des Energieverbrauchs auf einen Arbeitstag und ein Arbeitsjahr. Das optimierte System liefert einen Vorteil von 4,33 % bei dem Light-User-Profil und 8,49 % bei dem Medium-User-Profil gegenüber dem nicht optimierten OpenSuse-System. Ein Teil der Energieeinsparungen ist auf den optimierten Kernel zurückzuführen, was die Differenz bei den Werten der OpenSuse-Szenarien zeigt. Außerdem geht aus den Werten hervor, dass der Verwendungszweck des Computersystems starke Auswirkungen auf den Energieverbrauch hat und die Optimierungen bei höherer Auslastung des Systems einen größeren Effekt haben.

[5] Bei einem Arbeitstag von 8 Stunden
[6] Bei 200 Arbeitstagen pro Jahr

5. Schlussbetrachtung

Dieser Kapitel bietet eine kritische Reflexion der Arbeit und liefert einen Ausblick auf mögliche Anknüpfungspunkte.

5.1 Fazit

Diese Arbeit sollte zeigen, ob anhand von bestimmten Optimierungen die Energieeffizienz von typischen PC-Systemen mit Linux Betriebssystem, wie sie beispielsweise in Büros eingesetzt werden, gesteigert werden kann. Dabei sollte der Energieverbrauch gesenkt werden, ohne die Systemleistung in einem Maß zu beeinflussen, die den Benutzer bei der Erledigung alltäglicher Aufgaben stört. Dabei wurde nur die Zeit, in der das System für den Benutzer bedienbar ist, also ohne die Zeit während des Hoch- und Herunterfahrens und ohne systemweite Schlaf- und Ruhezustände berücksichtigt.

Je nach Verwendungszweck des Computers konnte dabei der Energieverbrauch mit einem optimierten System zwischen ca. 4 % bei geringer und ca. 8 % bei höherer Systemauslastung im Vergleich zu einem nicht modifizierte OpenSuse-System reduziert werden. Die Optimierungen zielten dabei insbesondere auf die Anpassung der CPU-Frequenz, die Reduzierung der Festplattenzugriffe und die Vermeidung unnötiger Interrupts sowie die Minimierung des gesamten Systems bezüglich dessen installierten Software, des verwendeten Kernels und der verwendeten Module. Die Optimierungen führten zu einer höheren Beanspruchung der CPU, die allerdings in einem vertretbaren Rahmen blieb und zu keinen merkbaren Verschlechterung der Reaktivität des Systems führte. Im Gegensatz dazu hatte die Reduzierung der Festplattenzugriffe auch positive Auswirkungen auf die I/O-Leistung.

Einige der Optimierung, wie die des CPU-Schedulers, wurden unter Berücksichtigung des verwendeten Szenarios getroffen, so dass das Benutzerverhalten a priori bekannt war und somit in die Optimierungsentscheidungen mit einfließen konnte, was in der Realität normalerweise nicht der Fall ist. Mit einem universeller ausgerichtetem System wären die Optimierungen weniger effizient gewesen. Ebenso wurden keine Szenarien mit sehr starker Systemauslastung, beispielsweise Szenarien für Softwareentwicklung, verwendet, weil diese mit den zur Verfügung stehenden Mitteln nicht realisierbar waren.

Die Seiteneffekte fanden in dieser Arbeit wenig Beachtung, weil es dazu notwendig gewesen wäre, den genauen Energieverbrauch einzelner Hardware-Komponenten zu messen. Weiterführende Optimierungen müssten Seiteneffekte und den Verbrauch einzelner Hardware-Komponenten im Detail berücksichtigen, um den Energieverbrauch weiter zu senken.

5.2 Ausblick

Für den Einsatz in der Praxis ist es von Vorteil, wenn das Benutzerverhalten und der Verwendungszweck des Systems nicht a priori bekannt sein muss. Dazu könnte beispielsweise ein Linux-Daemon Daten über bestimmte Charakteristika des Benutzerverhaltens zur Laufzeit bestimmen und die Optimierung auf Basis dieser Daten dynamisch vornehmen.

Eine weiterer Anknüpfungspunkt ist die Berücksichtigung von systemweiten Schlaf- und Ruhezuständen, sowie die Energieeffizienz beim Starten und Herunterfahrens des PCs. Mit zusätzlichen Server-Profilen könnte somit eine Architektur, wie die in der Praxis häufig verwendeten Client-Server-Architektur, ganzheitlich analysiert werden.

In eine andere Richtung geht die detaillierte Analyse einzelner Hardware-Komponenten und deren Seiteneffekte im Zusammenhang mit anderer Hardware, da die Messung des Gesamtenergieverbrauchs in dieser Richtung nur begrenzte Möglichkeiten zulässt.

Ein weiterer Themenbereich für anknüpfende Arbeiten ist die Berücksichtigung der Wärmeentwicklung von PCs, da die Thermik in unmittelbaren Zusammenhang mit der Auslastung des Systems steht und eine entsprechende Kühlung sowohl Einfluss auf den Energiebedarf, als auch, wegen der Lärmentwicklung, auf die Benutzerakzeptanz hat.

A Hardwareausstattung im Detail

Ausgabe des Programms „lshw“ ([102]) unter Gentoo:

```
ba-test
    description: Desktop Computer
    product: To Be Filled By O.E.M.
    vendor: To Be Filled By O.E.M.
    version: To Be Filled By O.E.M.
    serial: To Be Filled By O.E.M.
    width: 64 bits
    capabilities: vsyscall64
    configuration: boot=normal chassis=desktop uuid=00020003-0004-0005-0006-000700080009
  *-core
       description: Motherboard
       product: K8VB
       vendor: ASUSTeK Computer Inc.
       physical id: 0
       version: Rev 1.xx
       serial: MB-1234567890
       slot: USB2
     *-firmware
          description: BIOS
          vendor: American Megatrends Inc.
          physical id: 0
          version: 1007.001 (06/17/2005)
          size: 64KiB
          capacity: 448KiB
          capabilities: isa pci pnp apm upgrade shadowing escd cdboot bootselect socketedrom edd int13floppy1200 int13floppy720 int13floppy2880 int5printscreen int9keyboard int14serial int17printer int10video acpi usb agp ls120boot zipboot biosbootspecification
     *-cpu
          description: CPU
          product: AMD Athlon(tm) 64 Processor 3000+
          vendor: Advanced Micro Devices [AMD]
          physical id: 4
          bus info: cpu@0
          version: AMD Athlon(tm) 64 Processor 3000+
          serial: To Be Filled By O.E.M.
```

```
      slot: Socket 754
      size: 1GHz
      capacity: 2400MHz
      width: 64 bits
      clock: 200MHz
      capabilities: fpu fpu_exception wp vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 syscall nx mmxext x86-64 3dnowext 3dnow up rep_good cpufreq
    *-cache:0
         description: L1 cache
         physical id: 5
         slot: L1-Cache
         size: 64KiB
         capacity: 64KiB
         capabilities: pipeline-burst internal varies data
    *-cache:1
         description: L2 cache
         physical id: 6
         slot: L2-Cache
         size: 512KiB
         capacity: 512KiB
         capabilities: pipeline-burst internal varies unified
  *-memory
       description: System Memory
       physical id: 37
       slot: System board or motherboard
       size: 1GiB
    *-bank:0
         description: DIMM DDR Synchronous 400 MHz (2.5 ns)
         product: PartNum0
         vendor: Manufacturer0
         physical id: 0
         serial: SerNum0
         slot: DIMM0
         size: 512MiB
         width: 64 bits
         clock: 400MHz (2.5ns)
    *-bank:1
```

```
       description: DIMM DDR Synchronous 400 MHz (2.5 ns)
       product: PartNum1
       vendor: Manufacturer1
       physical id: 1
       serial: SerNum1
       slot: DIMM1
       size: 512MiB
       width: 64 bits
       clock: 400MHz (2.5ns)
     *-bank:2
       description: DIMM [empty]
       product: PartNum2
       vendor: Manufacturer2
       physical id: 2
       serial: SerNum2
       slot: DIMM2
*-pci:0
     description: Host bridge
     product: VT8385 [K8T800 AGP] Host Bridge
     vendor: VIA Technologies, Inc.
     physical id: 100
     bus info: pci@0000:00:00.0
     version: 01
     width: 32 bits
     clock: 66MHz
     configuration: driver=agpgart-amd64 latency=64
   *-pci
       description: PCI bridge
       product: VT8237 PCI bridge [K8T800/K8T890 South]
       vendor: VIA Technologies, Inc.
       physical id: 1
       bus info: pci@0000:00:01.0
       version: 00
       width: 32 bits
       clock: 66MHz
       capabilities: pci pm normal_decode bus_master cap_list
     *-display UNCLAIMED
```

```
        description: VGA compatible controller
        product: Radeon RV100 QY [Radeon 7000/VE]
        vendor: ATI Technologies Inc
        physical id: 0
        bus info: pci@0000:01:00.0
        version: 00
        width: 32 bits
        clock: 66MHz
        capabilities: agp agp-2.0 pm vga_controller bus_master cap_list
        configuration: latency=64 mingnt=8
*-storage:0 UNCLAIMED
     description: RAID bus controller
     product: PDC20376 (FastTrak 376)
     vendor: Promise Technology, Inc.
     physical id: 8
     bus info: pci@0000:00:08.0
     version: 02
     width: 32 bits
     clock: 66MHz
     capabilities: storage pm bus_master cap_list
     configuration: latency=240 maxlatency=18 mingnt=4
*-network:0 UNCLAIMED
     description: Ethernet controller
     product: 88E8001 Gigabit Ethernet Controller
     vendor: Marvell Technology Group Ltd.
     physical id: a
     bus info: pci@0000:00:0a.0
     version: 13
     width: 32 bits
     clock: 66MHz
     capabilities: pm vpd bus_master cap_list
     configuration: latency=64 maxlatency=31 mingnt=23
*-network:1
     description: Ethernet interface
     product: 3c905C-TX/TX-M [Tornado]
     vendor: 3Com Corporation
     physical id: e
```

bus info: pci@0000:00:0e.0

logical name: eth0

version: 78

serial: 00:04:75:d3:dd:b7

size: 100MB/s

capacity: 100MB/s

width: 32 bits

clock: 33MHz

capabilities: pm bus_master cap_list ethernet physical tp mii 10bt 10bt-fd 100bt 100bt-fd autonegotiation

configuration: autonegotiation=on broadcast=yes driver=3c59x duplex=full ip=10.10.1.123 latency=64 link=yes maxlatency=10 mingnt=10 multicast=yes port=MII speed=100MB/s

*-storage:1 UNCLAIMED

description: RAID bus controller

product: VIA VT6420 SATA RAID Controller

vendor: VIA Technologies, Inc.

physical id: f

bus info: pci@0000:00:0f.0

version: 80

width: 32 bits

clock: 33MHz

capabilities: storage pm bus_master cap_list

configuration: latency=64

*-ide

description: IDE interface

product: VT82C586A/B/VT82C686/A/B/VT823x/A/C PIPC Bus Master IDE

vendor: VIA Technologies, Inc.

physical id: f.1

bus info: pci@0000:00:0f.1

version: 06

width: 32 bits

clock: 33MHz

capabilities: ide pm bus_master cap_list

configuration: driver=VIA_IDE latency=32

*-ide:0

description: IDE Channel 0

physical id: 0

bus info: ide@0

```
          logical name: ide0
          clock: 33MHz
        *-disk
             description: ATA Disk
             product: ST340014A
             vendor: Seagate
             physical id: 0
             bus info: ide@0.0
             logical name: /dev/hda
             version: 8.01
             serial: 3JX8PTAB
             size: 37GiB (40GB)
             capacity: 37GiB (40GB)
             capabilities: ata dma lba iordy smart security pm partitioned partitioned:dos
             configuration: mode=udma5 signature=e83a5a1b smart=on
           *-volume:0
                description: EXT3 volume
                vendor: Linux
                physical id: 1
                bus info: ide@0.0,1
                logical name: /dev/hda1
                version: 1.0
                serial: 970af90c-f8f2-4c01-b69d-55c0c8970daf
                size: 133MiB
                capacity: 133MiB
                  capabilities: primary bootable journaled extended_attributes large_files huge_files recover
ext3 ext2 initialized
                        configuration: created=2009-07-20 15:51:49 filesystem=ext3 modified=2009-09-09
11:17:58 mounted=2009-09-09 11:15:07 state=clean
           *-volume:1
                description: Linux swap volume
                physical id: 2
                bus info: ide@0.0,2
                logical name: /dev/hda2
                version: 1
                serial: 79473e03-9ff6-486b-8d7f-e2fa318d25a7
                size: 1961MiB
                capacity: 1961MiB
```

```
              capabilities: primary swap initialized
              configuration: filesystem=swap pagesize=4096
           *-volume:2
              description: EXT3 volume
              vendor: Linux
              physical id: 3
              bus info: ide@0.0,3
              logical name: /dev/hda3
              logical name: /
              version: 1.0
              serial: 10f6b70f-a873-4637-861a-2a62ddbb5796
              size: 35GiB
              capacity: 35GiB
               capabilities: primary journaled extended_attributes large_files huge_files recover ext3 ext2
initialized
                        configuration: created=2009-07-20 16:00:51 filesystem=ext3 modified=2009-09-09
16:57:39 mount.fstype=ext3 mount.options=rw,noatime,errors=continue,data=ordered mounted=2009-
09-09 16:57:39 state=mounted
        *-ide:1
           description: IDE Channel 1
           physical id: 1
           bus info: ide@1
           logical name: ide1
           clock: 33MHz
         *-cdrom
            description: DVD writer
            product: _NEC DVD_RW ND-1300A
            physical id: 1
            bus info: ide@1.1
            logical name: /dev/hdd
            version: 1.0C
             capabilities: packet atapi cdrom removable nonmagnetic dma lba iordy audio cd-r cd-rw dvd
dvd-r
            configuration: mode=udma2 status=nodisc
     *-usb:0
        description: USB Controller
        product: VT82xxxxx UHCI USB 1.1 Controller
        vendor: VIA Technologies, Inc.
```

```
     physical id: 10
     bus info: pci@0000:00:10.0
     version: 81
     width: 32 bits
     clock: 33MHz
     capabilities: pm uhci bus_master cap_list
     configuration: driver=uhci_hcd latency=64 module=uhci_hcd
   *-usbhost
        product: UHCI Host Controller
        vendor: Linux 2.6.29-gentoo-r5 uhci_hcd
        physical id: 1
        bus info: usb@2
        logical name: usb2
        version: 2.06
        capabilities: usb-1.10
        configuration: driver=hub slots=2 speed=12.0MB/s
*-usb:1
     description: USB Controller
     product: VT82xxxxx UHCI USB 1.1 Controller
     vendor: VIA Technologies, Inc.
     physical id: 10.1
     bus info: pci@0000:00:10.1
     version: 81
     width: 32 bits
     clock: 33MHz
     capabilities: pm uhci bus_master cap_list
     configuration: driver=uhci_hcd latency=64 module=uhci_hcd
   *-usbhost
        product: UHCI Host Controller
        vendor: Linux 2.6.29-gentoo-r5 uhci_hcd
        physical id: 1
        bus info: usb@3
        logical name: usb3
        version: 2.06
        capabilities: usb-1.10
        configuration: driver=hub slots=2 speed=12.0MB/s
      *-usb
```

```
          description: Mouse
          product: Dell USB Mouse
          vendor: Dell
          physical id: 1
          bus info: usb@3:1
          version: 29.10
          capabilities: usb-1.10
          configuration: driver=usbhid maxpower=50mA speed=1.5MB/s
*-usb:2
     description: USB Controller
     product: VT82xxxxx UHCI USB 1.1 Controller
     vendor: VIA Technologies, Inc.
     physical id: 10.2
     bus info: pci@0000:00:10.2
     version: 81
     width: 32 bits
     clock: 33MHz
     capabilities: pm uhci bus_master cap_list
     configuration: driver=uhci_hcd latency=64 module=uhci_hcd
   *-usbhost
        product: UHCI Host Controller
        vendor: Linux 2.6.29-gentoo-r5 uhci_hcd
        physical id: 1
        bus info: usb@4
        logical name: usb4
        version: 2.06
        capabilities: usb-1.10
        configuration: driver=hub slots=2 speed=12.0MB/s
*-usb:3
     description: USB Controller
     product: VT82xxxxx UHCI USB 1.1 Controller
     vendor: VIA Technologies, Inc.
     physical id: 10.3
     bus info: pci@0000:00:10.3
     version: 81
     width: 32 bits
     clock: 33MHz
```

```
      capabilities: pm uhci bus_master cap_list
      configuration: driver=uhci_hcd latency=64 module=uhci_hcd
    *-usbhost
         product: UHCI Host Controller
         vendor: Linux 2.6.29-gentoo-r5 uhci_hcd
         physical id: 1
         bus info: usb@5
         logical name: usb5
         version: 2.06
         capabilities: usb-1.10
         configuration: driver=hub slots=2 speed=12.0MB/s
  *-usb:4
      description: USB Controller
      product: USB 2.0
      vendor: VIA Technologies, Inc.
      physical id: 10.4
      bus info: pci@0000:00:10.4
      version: 86
      width: 32 bits
      clock: 33MHz
      capabilities: pm ehci bus_master cap_list
      configuration: driver=ehci_hcd latency=64 module=ehci_hcd
    *-usbhost
         product: EHCI Host Controller
         vendor: Linux 2.6.29-gentoo-r5 ehci_hcd
         physical id: 1
         bus info: usb@1
         logical name: usb1
         version: 2.06
         capabilities: usb-2.00
         configuration: driver=hub slots=8 speed=480.0MB/s
  *-isa
      description: ISA bridge
      product: VT8237 ISA bridge [KT600/K8T800/K8T890 South]
      vendor: VIA Technologies, Inc.
      physical id: 11
      bus info: pci@0000:00:11.0
```

```
       version: 00
       width: 32 bits
       clock: 33MHz
       capabilities: isa pm bus_master cap_list
       configuration: latency=0
  *-multimedia
       description: Multimedia audio controller
       product: VT8233/A/8235/8237 AC97 Audio Controller
       vendor: VIA Technologies, Inc.
       physical id: 11.5
       bus info: pci@0000:00:11.5
       version: 60
       width: 32 bits
       clock: 33MHz
       capabilities: pm cap_list
       configuration: driver=VIA 82xx Audio latency=0 module=snd_via82xx
  *-communication UNCLAIMED
       description: Communication controller
       product: AC'97 Modem Controller
       vendor: VIA Technologies, Inc.
       physical id: 11.6
       bus info: pci@0000:00:11.6
       version: 80
       width: 32 bits
       clock: 33MHz
       capabilities: pm cap_list
       configuration: latency=0
*-pci:1
     description: Host bridge
     product: K8 [Athlon64/Opteron] HyperTransport Technology Configuration
     vendor: Advanced Micro Devices [AMD]
     physical id: 101
     bus info: pci@0000:00:18.0
     version: 00
     width: 32 bits
     clock: 33MHz
*-pci:2
```

```
        description: Host bridge
        product: K8 [Athlon64/Opteron] Address Map
        vendor: Advanced Micro Devices [AMD]
        physical id: 102
        bus info: pci@0000:00:18.1
        version: 00
        width: 32 bits
        clock: 33MHz
      *-pci:3
        description: Host bridge
        product: K8 [Athlon64/Opteron] DRAM Controller
        vendor: Advanced Micro Devices [AMD]
        physical id: 103
        bus info: pci@0000:00:18.2
        version: 00
        width: 32 bits
        clock: 33MHz
      *-pci:4
        description: Host bridge
        product: K8 [Athlon64/Opteron] Miscellaneous Control
        vendor: Advanced Micro Devices [AMD]
        physical id: 104
        bus info: pci@0000:00:18.3
        version: 00
        width: 32 bits
        clock: 33MHz
        configuration: driver=k8temp
```

B Benutzerprofil-Skripte

B.1 Light-User Shell-Skript

```
#!/bin/sh
LOG_FILE=$1
echo "start User Simulation" >> $LOG_FILE

#1. Internet surfen (Firefox wikipedia)
cnee --replay --f /home/markus/conf/light_macro/firefox.xns -rwp --time 25
```

```
xdotool key control+alt+F7
echo 1 >> $LOG_FILE

#2. E-Mail schreiben (Mozilla Thunderbird)
cnee --replay --f /home/markus/conf/light_macro/thunderbird.xns -rwp --time 2
xdotool key control+alt+F7
echo 2 >> $LOG_FILE

#3. Text schreiben mit (OpenOffice Writer)
cnee --replay --f /home/markus/conf/light_macro/writer.xns -rwp --time 5
xdotool key control+alt+F7
echo 3 >> $LOG_FILE

#3.1. Vorhin geschriebenen Text geringfügig editieren und nachher loeschen (OpenOffice Writer)
cnee --replay --f /home/markus/conf/light_macro/writer_2.xns -rwp --time 3
rm /home/markus/derpanther.odt
xdotool key control+alt+F7
echo 3.1 >> $LOG_FILE

#4. Weiteren Text editieren (OpenOffice Writer)
cnee --replay --f /home/markus/conf/light_macro/writer_edit.xns -rwp --time 3
xdotool key control+alt+F7
echo 4 >> $LOG_FILE

#5. PDF-Datei lesen (Evince)
cnee --replay --f /home/markus/conf/light_macro/evince.xns -rwp --time 2
xdotool key control+alt+F7
echo 5 >> $LOG_FILE

#6. Webbrowser-Cache leeren (Firefox)
cnee --replay --f /home/markus/conf/light_macro/firefox_clear.xns -rwp --time 2
xdotool key control+alt+F7
echo 6 >> $LOG_FILE
```

B.2 Medium-User Shell-Skript

```
#!/bin/sh
LOG_FILE=$1
```

```
echo "start User Simulation (NORMAL)" >> $LOG_FILE

#01.1 Mp3 hören (VLC)
echo "Start 01:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE
vlc /home/markus/scripts/doc/1.mp3 &
xdotool key control+alt+F7
echo "Stop 01:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE

#02. E-Mail schreiben (Mozilla Thunderbird)
echo "Start 02:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE
cnee --replay --f /home/markus/conf/light_macro/thunderbird.xns -rwp --time 2
xdotool key control+alt+F7
echo "Stop 02:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE

#03 Internet surfen (Firefox wikipedia) (Anwendung geöffnet lassen)
echo "Start 03:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE
cnee --replay --f /home/markus/conf/normal_macro/firefox.xns -rwp --time 10
xdotool key control+alt+F7
echo "Stop 03:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE

#04. E-Mail schreiben (Mozilla Thunderbird) (Anwendung geöffnet lassen)
echo "Start 04:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE
cnee --replay --f /home/markus/conf/normal_macro/thunderbird.xns -rwp --time 5
xdotool key control+alt+F7
echo "Stop 04:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE

#4. Hintergrundprozess: Kompilieren
sh /home/markus/scripts/normal_macro/gcc_scripts.sh $LOG_FILE &

#5. Text schreiben mit (OpenOffice Writer) (wei beim light-macro)
echo "Start 05:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE
cnee --replay --f /home/markus/conf/light_macro/writer.xns -rwp --time 1
xdotool key control+alt+F7
echo "Stop 05:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE

#6. Vorhin geschriebenen Text geringfügig editieren und nachher loeschen (OpenOffice Writer)
echo "Start 06:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE
```

```
cnee --replay --f /home/markus/conf/light_macro/writer_2.xns -rwp --time 3
rm /home/markus/derpanther.odt
xdotool key control+alt+F7
echo "Stop 06 "  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE

#07. Weiteren Text editieren (OpenOffice Writer)
echo "Start 07 "  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE
cnee --replay --f /home/markus/conf/light_macro/writer_edit.xns -rwp --time 3
xdotool key control+alt+F7
echo "Stop 07:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE

#7. Hintergrundprozess: Komprimieren
sh /home/markus/scripts/normal_macro/tar_scripts.sh $LOG_FILE &

#1.2. MP3-Player schließen (VLC)
sudo killall vlc &

#8. Hintergrundprozess: Kompilieren (GCC)
sh /home/markus/scripts/normal_macro/gcc_script_2.sh $LOG_FILE &

#08. PDF-Datei lesen (Evince) (wie beim light-macro)
echo "Start 08:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE
cnee --replay --f /home/markus/conf/light_macro/evince.xns -rwp --time 1
xdotool key control+alt+F7
echo "Stop 08:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE

#09. Thunderbird schließen
echo "Start 09 "  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE
cnee --replay --f /home/markus/conf/normal_macro/thunderbird_close.xns -rwp --time 1
xdotool key control+alt+F7
echo "Stop 09:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE

#10. Firefox schließen & Cache löschen
echo "Start 10:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE
cnee --replay --f /home/markus/conf/normal_macro/firefox_close.xns -rwp --time 1
xdotool key control+alt+F7
echo "Stop 10:"  `date '+%y-%m-%d-%H%M%S - %N' `  >> $LOG_FILE
```

#12. Archiv wieder löschen

rm /home/markus/doc.tar.gz

C Messungen

C.1 CPU-Statistik (Light-User)

Szenario		**1**	**2**	**3**	**4**	**5**	**6**
P-States	**2.00 Ghz**	6,35%	1,42%	1,61%	100,00%	8,41%	7,28%
	1.80 Ghz	0,74%	0,00%	0,10%	0,00%	1,02%	0,60%
	1.00 Ghz	92,86%	98,58%	98,29%	0,00%	90,58%	92,12%
C2-Dauer (ms)	**1. Durchlauf**	728	733	716	759	1145	1146
	2. Durchlauf	652	645	651	654	1129	1115
	Gesamt	1380	1378	1367	1414	2274	2262
C3-Dauer (ms)	**1. Durchlauf**	622	632	609	640	726	1051
	2. Durchlauf	568	563	561	555	731	1033
	Gesamt	1189	1194	1170	1195	1457	2083
Timer-Sleep (s)	**1. Durchlauf**	541,79	533,85	533,66	562,4	530,37	504,68
	2. Durchlauf	564,44	553,43	547,97	582,25	559,22	532,85
	Gesamt	1106,23	1087,28	1081,63	1144,65	1089,59	1037,53

Tabelle 13: CPU-Statistik (Light-User)

C.2 CPU-Statistik (Medium-User)

Szenario		**1**	**2**	**3**	**4**	**5**	**6**
P-States	**2.00 Ghz**	10,04%	1,42%	3,50%	100,00%	12,35%	11,53%
	1.80 Ghz	0,91%	0,00%	1,08%	0,00%	1,44%	1,08%
	1.00 Ghz	89,04%	98,58%	95,53%	0,00%	86,21%	87,39%
C2-Dauer (ms)	**1. Durchlauf**	1116	1151	1074	1183	1151	2176
	2. Durchlauf	1042	998	1011	1110	998	2148
	Gesamt	2158	2149	2085	2293	2149	4325
C3-Dauer (ms)	**1. Durchlauf**	991	995	959	1062	995	1966
	2. Durchlauf	934	897	918	1007	897	1997
	Gesamt	1925	1892	1877	2069	1892	3963
Timer-Sleep (s)	**1. Durchlauf**	509,22	480,77	493,99	535,21	494,95	555,91
	2. Durchlauf	534,25	503,15	514,33	551,29	517,95	579,32
	Gesamt	1043,47	983,93	1008,32	1086,5	1012,89	1135,23

Tabelle 14: CPU-Statistik (Medium-User)

C.3 Festplatten-Statistik (Light-User)

1. Durchgang							
	Szenario	**1**	**2**	**3**	**4**	**5**	**6**
Lesen	**Zugriffe**	4010	4000	3788	4412	4401	3602
	Zusammengefügt	45	47	47	46	1874	44
	# Sektoren	391231	390376	388900	394445	490366	443337
	Dauer	50800	52213	52170	57873	39804	36073
Schreiben	**Zugriffe**	1120	1136	1152	1446	2772	2521
	Zusammengefügt	2048	1948	1849	2315	6972	6510
	# Sektoren	29235	28725	27539	32589	86749	78864
	Dauer	71467	88513	41287	124640	362512	399133
I/O	**Dauer I/O**	30960	31103	29643	34067	45925	35457
	Dauer I/O (gewichtet)	122263	140727	93457	182507	402312	435200

Tabelle 15: Festplatten-Statistik 1. Durchgang (Light-User)

2. Durchgang							
	Szenario	**1**	**2**	**3**	**4**	**5**	**6**
Lesen	**Zugriffe**	1	277	1375	32	360	379
	Zusammengefügt	0	0	2	0	0	1
	# Sektoren	8	3032	14224	307	3072	4501
	Dauer	0	1227	7110	187	1893	2667
Schreiben	**Zugriffe**	1678	1837	1715	1717	3620	3848
	Zusammengefügt	1837	1921	3367	1884	6501	6517
	# Sektoren	28131	30072	40672	28816	82573	82936
	Dauer	19517	30072	34231	41162	350605	364773
I/O	**Dauer I/O**	6350	8140	12953	6543	22897	20600
	Dauer I/O (gewichtet)	46200	86577	52450	49700	352499	601040

Tabelle 16: Festplatten-Statistik 2. Durchgang (Light-User)

Gesamt							
	Szenario	**1**	**2**	**3**	**4**	**5**	**6**
Lesen	**Zugriffe Lesen**	4011	4278	5163	4444	4761	3981
	Zusammengefügt	45	47	49	46	1874	45
	# Sektoren	391239	393408	403124	394751	493438	447839
	Dauer Lesen	50800	53440	59280	58060	41697	38740
Schreiben	**Zugriffe Schreiben**	2799	2973	2868	3164	6392	6369
	Zusammen-gefügt	3885	3869	5216	4199	13473	13027
	# Sektoren	57365	58797	68211	61405	169323	161800
	Dauer Schreiben	90984	118585	75518	165802	713117	763907
	Dauer I/O	37310	39243	42597	40610	68823	56057
	Dauer I/O (gewichtet)	168463	227303	145907	232207	754811	1036240
Caching	**Zugriffe lesen**	99,98%	93,07%	63,70%	99,28%	91,81%	89,47%
	Zusammen-gefügt	100,00%	99,29%	95,04%	100,00%	100,00%	98,50%
	# Sektoren	100,00%	99,22%	96,34%	99,92%	99,37%	98,98%
	Dauer lesen	100,00%	97,65%	86,37%	99,68%	95,24%	92,61%
	Zugriffe schreiben	-49,81%	-61,74%	-48,86%	-18,74%	-30,58%	-52,66%
	Zusammen-gefügt	10,28%	1,40%	-82,10%	18,60%	6,76%	-0,12%
	# Sektoren	3,78%	-4,69%	-47,69%	11,58%	4,81%	-5,16%
	Dauer schreiben	72,69%	66,03%	17,09%	66,98%	3,28%	8,61%
Caching I/O	**Dauer I/O**	79,49%	73,83%	56,30%	80,79%	50,14%	41,90%
	Dauer I/O (gewichtet)	62,21%	38,48%	43,88%	72,77%	12,38%	-38,11%

Tabelle 17: Festplatten-Statistik Gesamt (Light-User)

C.4 Festplatten-Statistik (Medium-User)

1. Durchgang							
		1	**2**	**3**	**4**	**5**	**6**
Lesen	**Zugriffe**	5082	6563	4752	5328	4499	5918
	Zusammengefügt	61	68	63	72	2263	87
	# Sektoren	458549	473755	388900	461685	548387	554280
	Dauer	69477	76727	63590	70383	54233	62120
Schreiben	**Zugriffe**	1302	1256	1260	1540	3261	3224
	Zusammengefügt	6529	11147	6349	6703	13874	15734
	# Sektoren	67472	103373	65192	68411	148258	160899
	Dauer	153420	456060	97777	187873	481779	685090
I/O	**Dauer I/O**	40743	49937	38113	42667	56055	54080
	Dauer I/O (gewichtet)	222890	532780	161360	258250	536007	747203

Tabelle 18: Festplatten-Statistik 1. Durchgang (Medium-User)

2. Durchgang							
		1	**2**	**3**	**4**	**5**	**6**
Lesen	**Zugriffe**	40	493	425	480	513	336
	Zusammengefügt	0	1	0	1	29	3
	# Sektoren	325	5912	4328	4576	8336	6123
	Dauer	150	2403	2203	2453	3444	2340
Schreiben	**Zugriffe**	1866	2137	2088	2389	4242	4326
	Zusammengefügt	5942	6407	6144	6242	13676	14579
	# Sektoren	62485	68400	65899	69083	145197	151277
	Dauer	43706	68400	79595	206741	458083	477220
I/O	**Dauer I/O**	8183	11403	10997	11930	29468	23200
	Dauer I/O (gewichtet)	94010	176677	161430	254213	461527	683553

Tabelle 19: Festplatten-Statistik 2. Durchgang (Medium-User)

Gesamt							
	Szenario	**1**	**2**	**3**	**4**	**5**	**6**
Lesen	**Zugriffe Lesen**	4011	4278	5163	4444	4761	3981
	Zusammengefügt	45	47	49	46	1874	45
	# Sektoren	391239	393408	403124	394751	493438	447839
	Dauer Lesen	50800	53440	59280	58060	41697	38740
Schreiben	**Zugriffe Schreiben**	2799	2973	2868	3164	6392	6369
	Zusammen-gefügt	3885	3869	5216	4199	13473	13027
	# Sektoren	57365	58797	68211	61405	169323	161800
	Dauer Schreiben	90984	118585	75518	165802	713117	763907
	Dauer I/O	37310	39243	42597	40610	68823	56057
	Dauer I/O (gewichtet)	168463	227303	145907	232207	754811	1036240
Caching	**Zugriffe lesen**	99,98%	93,07%	63,70%	99,28%	91,81%	89,47%
	Zusammen-gefügt	100,00%	99,29%	95,04%	100,00%	100,00%	98,50%
	# Sektoren	100,00%	99,22%	96,34%	99,92%	99,37%	98,98%
	Dauer lesen	100,00%	97,65%	86,37%	99,68%	95,24%	92,61%
	Zugriffe schreiben	-49,81%	-61,74%	-48,86%	-18,74%	-30,58%	-52,66%
	Zusammen-gefügt	10,28%	1,40%	-82,10%	18,60%	6,76%	-0,12%
	# Sektoren	3,78%	-4,69%	-47,69%	11,58%	4,81%	-5,16%
	Dauer schreiben	72,69%	66,03%	17,09%	66,98%	3,28%	8,61%
Caching I/O	**Dauer I/O**	79,49%	73,83%	56,30%	80,79%	50,14%	41,90%
	Dauer I/O (gewichtet)	62,21%	38,48%	43,88%	72,77%	12,38%	-38,11%

Tabelle 20: Festplatten-Statistik Gesamt (Medium-User)

C.5 Interrupt-Statistik (Light-User)

1. Durchgang						
Szenario	**1**	**2**	**3**	**4**	**5**	**6**
timer	31033	31435	30244	32680	41882	45655
ide0	5708	5712	5423	6341	8311	6763
ide1	0	0	0	0	6580	5644
eth0	2912	3209	3424	3235	2988	2318
LOC	29157	30358	29373	27278	60295	46962
Gesamt	68810	70714	68464	69535	120056	107342

Tabelle 21: Interrupt-Statistik 1. Durchgang (Light-User)

2. Durchgang						
Szenario	**1**	**2**	**3**	**4**	**5**	**6**
timer	31553	30708	30105	31969	42789	45710
ide0	1681	2111	4493	1738	4433	4820
ide1	0	0	0	0	6783	5814
eth0	2765	2759	3749	3085	3201	3809
LOC	28223	29466	30225	25913	58057	46751
Gesamt	64223	65045	68572	62705	115263	106903

Tabelle 22: Interrupt-Statistik 2. Durchgang (Light-User)

Gesamt						
Szenario	**1**	**2**	**3**	**4**	**5**	**6**
timer	62586	62143	60349	64649	84671	91364
ide0	7389	7823	9915	8079	12745	11583
ide1	0	0	0	0	13363	11458
eth0	5678	5968	7173	6320	6189	6127
LOC	57380	59825	59599	53191	118352	93714
Gesamt	133033	135759	137036	132240	235319	214246

Tabelle 23: Interrupt-Statistik Gesamt (Light-User)

C.6 Interrupt-Statistik (Medium-User)

1. Durchgang						
Szenario	**1**	**2**	**3**	**4**	**5**	**6**
timer	44738	43083	43127	48782	110935	93842
ide0	7063	9968	6547	7426	9315	10492
ide1	0	0	0	0	6671	5730
eth0	3488	3931	3569	2420	3588	2944
LOC	57728	61644	59248	56387	158067	126584
Gesamt	113016	118627	112491	115014	288575	239592

Tabelle 24: Interrupt-Statistik 1. Durchgang (Medium-User)

2. Durchgang						
Szenario	**1**	**2**	**3**	**4**	**5**	**6**
timer	44263	41069	43044	48456	110597	96991
ide0	1985	1895	1918	3024	4803	4385
ide1	0	0	0	0	6902	5904
eth0	3579	3867	3565	3399	3542	3852
LOC	57193	58694	59554	58688	157949	126150
Gesamt	107020	105525	108081	113566	283793	237282

Tabelle 25: Interrupt-Statistik 2. Durchgang (Medium-User)

Gesamt						
Szenario	**1**	**2**	**3**	**4**	**5**	**6**
timer	89001	84152	86171	97237	221532	190833
ide0	9047	11863	8465	10450	14118	14877
ide1	0	0	0	0	13573	11634
eth0	7067	7798	7134	5819	7130	6796
LOC	114921	120339	118802	115074	316015	252734
Gesamt	220036	224152	220572	228580	572368	476874

Tabelle 26: Interrupt-Statistik Gesamt (Medium-User)

C.7 CPU-Belastung (Light-User)

Szenario		1	2	3	4	5	6
Mittelwert	**user**	9,87	11,13	11,63	7,84	9,71	7,35
	system	2,89	2,86	3,03	1,72	3,72	2,84
	iowait	2,19	2,31	2,53	2,43	3,68	2,64
	idle	85,06	83,7	82,81	88,01	82,6	86,79
Maximum	**user**	99,33	98,67	99	99,33	99,33	97
	system	23,33	20,33	24,33	20	31,59	29,67
	iowait	96,33	96,67	96,67	98,67	95,31	97,67
	idle	99	99	98,67	100	98	100
> 90 %	**user**	1,21%	1,87%	1,87%	1,31%	1,05%	0,29%
	system	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
	iowait	0,29%	0,37%	0,34%	0,81%	0,34%	0,34%
> 75 %	**user**	2,52%	4,26%	2,52%	2,10%	2,26%	1,03%
	system	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
	iowait	1,03%	1,03%	1,03%	1,50%	0,71%	0,92%
Load-Average		0,19698	0,25397	0,23810	0,19238	0,27762	0,25889

Tabelle 27: CPU-Belastung (Light-User)

C.8 CPU-Belastung (Medium-User)

Szenario		1	2	3	4	5	6
Mittelwert	**user**	15,18	19,09	17,56	12,64	14,35	13,8
	system	3,86	4,4	4,07	2,77	5	4,05
	iowait	2,16	2,74	1,98	2,66	3,52	2,91
	idle	78,8	73,77	76,4	81,93	75,59	77,86
Maximum	**user**	99,67	99,33	99	99,67	98,67	99
	system	49	43,67	51	48,33	41,53	47,33
	iowait	90,33	91,33	92,33	96,67	89,72	91
	idle	99,33	98,67	99,33	100	97,99	99,33
> 90 %	**user**	3,10%	5,39%	4,26%	2,76%	1,92%	1,18%
	system	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
	iowait	0,03%	0,05%	0,13%	0,76%	0,03%	0,08%
> 75 %	**user**	5,34%	9,57%	5,34%	5,34%	5,15%	3,60%
	system	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
	iowait	0,74%	0,74%	0,74%	0,74%	0,47%	0,37%
Load-Average		0,33413	0,42841	0,34810	0,29127	0,59698	0,48540

Tabelle 28: CPU-Belastung (Medium-User)

C.9 Festplatten-Belastung (Light-User)

		1	2	3	4	5	6
Mittelwert	**Transports / s**	5,79	6,15	6,69	6,38	9,32	8,58
	Lesen/s	278	280	287	280	335	300
	Schreiben/s	45,09	46,38	53,79	48,89	132,18	126,49
	Durchschnittsgröße	7,96	7,48	7,8	8,13	13,05	11,81
	Queuelänge	0,13	0,18	0,11	0,18	0,58	0,82
	Durchschnittszeit 1	4,99	4,72	5,19	5,09	12	14,62
	Durchschnittszeit 2	1,52	1,29	1,45	1,46	4,36	1,83
	CPU-Zeit	2,92	3,08	3,34	3,19	5,36	4,39
Maximum	**Transports / s**	219	258,33	197,33	341,67	377,21	346,67
	Lesen/s	54144	53669	53352	52421	68302	64016
	Schreiben/s	2112	2405	3165	3077	4211	3963
	Durchschnittsgröße	459	441	471	491	642	613
	Queuelänge	37,01	76,77	12,77	56,63	79,27	91,53
	Durchschnittszeit 1	197,31	549,13	192,62	228,93	693,69	887,68
	Durchschnittszeit 2	23,33	18,33	20	16,83	152	27,5
	CPU-Zeit	100	100	100	100	99,73	100
Queue > 1		2,84%	2,94%	3,23%	3,42%	4,14%	4,69%
CPU-Zeit > 50		1,68%	1,97%	2,02%	1,97%	2,67%	3,21%

Tabelle 29: Festplatten-Belastung (Light-User)

C.10 Festplatten-Belastung (Medium-User)

		1	2	3	4	5	6
Mittelwert	**Transports / s**	6,87	9,75	6,95	7,93	10,46	11,68
	Lesen/s	328	346	330	335	375	386
	Schreiben/s	100,67	133,59	101,45	106,65	225,56	241,78
	Durchschnittsgröße	13,69	13,94	14,19	12,7	17,7	17,68
	Queuelänge	0,25	0,56	0,25	0,4	0,78	1,12
	Durchschnittszeit 1	6,97	6,89	6,58	7,56	14,85	19,15
	Durchschnittszeit 2	1,87	1,9	1,91	1,77	3,9	2,35
	CPU-Zeit	3,78	4,77	3,79	4,23	6,54	5,98
Maximum	**Transports / s**	226,33	425,33	257,7	405,33	228	275,67
	Lesen/s	51592	52267	52675	54701	64016	58763
	Schreiben/s	30056	26741	25683	29899	31815	22981
	Durchschnittsgröße	472	530	443	452	696	584
	Queuelänge	50,37	101,79	63,78	126,56	74,96	83,26
	Durchschnittszeit 1	458,74	491,21	480,33	609,23	758,23	841,23
	Durchschnittszeit 2	30	30	20	26,67	136	26,67
	CPU-Zeit	100	100	100	100	99,6	100
Queue > 1		4,07%	4,61%	3,91%	4,58%	5,12%	8,24%
CPU-Zeit > 50		2,51%	3,50%	2,38%	3,06%	3,98%	3,92%

Tabelle 30: Festplatten-Belastung (Medium-User)

C.11 Inhalt der CD-Beilage

Die CD enthält neben dieser Arbeit im PDF-Format folgende Verzeichnisse:

- Auswertungen:

  Dieses Verzeichnis enthält die ausgewerteten Messdaten für jedes Szenario, sowie eine Übersicht der Daten, unterteilt nach Light- und Medium-Benutzerprofil (Auswertung_light.ods bzw. Auswertung_medium.ods).

  Die Notation s1p1c1_001 ist wie folgt zu verstehen:

  - Szenario 1 (1 = Gentoo, 2 = OpenSuse)
  - Profil 1 (1 = Light-User, 2 = Medium-User, auch Normal-User genannt)
  - Configuration 1 (1 = Ondemand-Governor, 2 = Powersave-Governor, 3 = Conservative-Governor, 4 = Performance-Governor)

  - Suffix _001: Die Auswertungen fassen alle drei Testläufe zusammen, weshalb an dieser Stelle der Suffix immer 001 ist. Im Verzeichnis „Data“ wird durch den Suffix Testlaufnummer angegeben (1. Testlauf (002 = 2. Testlauf, ...)

- Conf

  Hier befinden sich die Xnee-Makros-Dateien.

- Data

  Dieses Verzeichnis enthält Daten & Logfiles im Rohformat. Notation siehe Verzeichnis „Auswertung“.

- Kernel

  In diesem Verzeichnis befinden sich die Konfigurationsdateien der verwendeten Kernel

- Scripts:

  Die Skripte zur Steuerung der Xnee-Makros und zum Sammeln der Analysewerte befinden sich hier.

- Tools:

  Dieses Verzeichnis enthält Ruby-Skripte, in erster Linie Parser, die der Erleichterung der Logfile-Analyse dienen.

Glossar

32-Bit Protected-Mode: Der 32-Bit Protected Mode ist ein spezieller Betriebsmodus bei x86-Architekturen, bestehend aus vier Schutzebenen, auch Ringe genannt, die es erlauben, den ausführenden Codesegmenten unterschiedliche Privilegien zuzuteilen. Die Ringe erlauben die Unterteilung in Benutzer- und Kernel-Modus, wie er in modernen Betriebssystemen umgesetzt wird (siehe Kernel- und Userspace). Das Gegenstück zum Protected-Mode ist der Real-Mode.

32-Bit Real-Mode: Im Gegensatz zum Protected-Mode kann im Real-Mode jedes Programm auf den gesamten Hauptspeicher und die Hardware ohne weiteres zugreifen, was ein massives Sicherheitsproblem darstellt und dieser Modes deshalb für moderne Multitaskting-Betriebssysteme ungeeignet ist.

Advanced Configuraton and Power Interface (ACPI): ACPI ist der Nachfolger von APM und spezifiziert einen offenen Industriestandard für die Energievervaltung und Konfiguration von Desktop-PCs, Notebooks und Servern.

Advanced Power Management (APM): Vorgänger von ACPI, bei dem das Power Management größtenteils in der Firmware implementiert ist und die Einflüssmöglichkeiten für das Betriebssystem somit sehr gering sind.

Basic Input Output System (BIOS): BIOS wird die Firmware der x86-PCs genannt, die unmittelbar nach dessen Einschalten ausgeführt wird.

Cache-Miss: Ein Cache-Miss bezeichnet den fehlgeschlagenen Versuch, Daten von einem schnellen Zwischenspeicher zu lesen. Das Lesen schlägt fehl, wenn die angeforderten Daten nicht vorhanden oder nicht (mehr) gültig sind.

CMOS: Complementary Metal Oxide Semiconductor, ein Halbleiter-Bauelement, das in der Regel in integrierten Schaltkreisen verwendet wird.

C-States: C-States sind ACPI-Zustände der CPU, wobei C0 der Arbeitszustand ist und C1 bis Cx die Schlafzustände kennzeichnen. Je höher der Schlafzustand, desto weniger CPU-Kontext wird gesichert und desto länger dauert es, bis die CPU wieder in den Arbeitszustand versetzt werden kann. Mit der Tiefe des Schlafzustands sinkt allerdings auch der Energiebedarf.

Differentiated Definition Block: Der DDB stellt dem Betriebssystem Informationen über die Konfiguration und Ansteuerung von ACPI-fähigen Komponenten bereit und bildet damit die Grundlage das ACPI-Systems.

Dynamic Voltage Scaling (DVS): DVS ist eine Technik, die es erlaubt, die Versorgungsspannung von Hardware zur Laufzeit zu verändern.

Dynamic Power Management (DPM): DPM ist eine Technik, die es erlaubt, die Energieverwaltung zur Laufzeit an veränderte Bedingungen anzupassen.

Firmware: Firmware ist eine in elektronische Geräte fest eingebettete Software, die in der Regel nicht oder nur unter bestimmten Bedingungen dazu bestimmt ist, vom Endanwender geändert zu werden.

G-States: Globale Power-States sind Zustände, die das gesamte Computersystem umfassen. Es wird zwischen G0 (Working State), G1 (Sleeping State), G2 (Soft-Off) und G3 (Mechanical Off) unterschieden. Je höher der G-State, desto länger dauert es, das System wieder in den Arbeitszustand zu versetzten. Im Zustand G2 und G3 wird weder vom Betriebssystem noch von den laufenden Anwendungsprogrammen der Kontext gesichert, so dass diese auch nicht wieder hergestellt werden können.

Kernel: Der Kernel (dt. Betriebssystemkern) ist der zentrale Bestandteil des Betriebssystems. Durch die monolithe Eigenschaft des Linux-Kernel dient dieser als Schnittstelle zur Hardware und zur Verwaltung des Speichers, der Prozesse und des Dateisystems.

Kernelspace: Der Kernelspace ist ein privilegierter Bereich, in dem aus Sicherheitsgründen nur der Kernel arbeiten kann. Dessen Gegenstück ist der Userspace.

Operating System Power Management (OSPM): OSPM ist durch das Betriebssystem gesteuerte Energieverwaltung und wird daher abgegrenzt vom Power Management, dass in der Hardware eingebettet ist und auf das das Betriebssystem keinen Einfluss nehmen kann.

Plattform: Die Plattform bezeichnung die Hardware und das Betriebssystem eines PCs eine eine zusammengefasste Komponente, in Abgrenzung zur Anwendungssoftware.

Policy: Eine Policy ist ein Algorithmus für das Power Management, der entscheidet, zu welchem Zeitpunkt und zu welchen Bedingungen die Hardware angewiesen wird, ihren Zustand zu verändern.

Prädiktive Strategien: Prädiktive Strategien machen Annahmen über das zukünftige Verhalten eines Systems, um davon ausgehend Entscheidungen bezüglich das Power Managements zu treffen; beispielsweise das Festlegen des Zeitpunkts, wann eine Hardwarekomponente in einen Schlafzustand versetzt wird.

P-States: Moderne Prozessoren können mit verschiedenen Frequenzen betrieben werden, die sogenannten Performance-States oder P-States. Eine niedrigere Frequenz hat einen geringeren Energiebedarf zur Folge, reduziert aber auch die Leistung der CPU.

S-States: S-States oder System-States geben Zustand des gesamten Systems an und reichen von S0 (Working) bis S5 (Tiefschlaf, ohne Sicherung irgendeines Kontextes wie Arbeitsspeicher oder CPU-Register). Je tiefer der Schlafzustand, desto größer der Energiespareffekt, aber desto länger dauert es, bis sich das System wieder im betriebsbereiten Zustand S0 befindet und desto weniger Systemkontext wird gesichert.

Timeout-Strategien: Timer-Strategien sind simple Power Management-Strategien, die nach Ablauf eines definierten, festen Zeitintervalls eine Hardware-Komponente in einen energiesparenderen Zustand versetzen.

Timer Tick: Der Timer Tick ist ein Kernel-interner Timer-Interrupt, der typischerweise zwischen 100 und 1000 mal pro Sekunde ausgelöst wird und der für einer Vielzahl von Kernel-Funktionen, beispielsweise für periodische Aufgaben, genutzt wird.

Throttling: Throttling bedeutet das Auslassen von CPU-Takten und wird in der Regel angewandt, um die CPU-Temperatur zu reduzieren. Der Energieeinsparung ist geringer als bei dem Herabsetzen der Betriebsspannung.

T-States: Throttling States stellen unterschiedliche Throttling-Raten der CPU dar. Diese werden in Prozent angegeben, so dass zum Beispiel ein Throttling von 50% bedeutet, dass jeder zweite Takt ausgelassen wird.

Userspace: Der Userspace ist der Bereich in einem Linux-System, in dem Anwendungsprogramme aus Sicherheitsgründen mit begrenzten Rechten ausgeführt werden, in Abgrenzung zum Kernelspace.

X-Client: Der X-Client ist ein Anwendungsprogramm, dass als Teil des X Window System die grafischen Ein- und Ausgabe-Dienste des X-Servers nutzt.

X-Server: Der X-Server steuert die Ein- und Ausgabegeräte eines PCs und stellt den X-Clients grafische Dienste zur Verfügung.

X Window System: Das X Window System ist ein als Client-Server-System entworfenes Netzwerkprotokoll und eine Standardbaukasten zum Erstellen von grafischen Benutzeroberflächen. Es wurde auf allen gebräuchlichen Betriebssystemen implementiert.

Abkürzungsverzeichnis

ACPI: Advanced Configuration and Power Interface.....
ACPICA: ACPI Component Architecture.....
AML: ACPI Machine Language.....
APM: Advanced Power Management.....
ASL: ACPI Source Language.....
ATO: Adaptive Timeout Policy.....
CFQ: Completely Fair Queing.....
CMOS: Complementary Metal-Oxide Semiconductor.....
DDB: Differentiated Definition Block.....
DPM: Dynamic Power Management.....
DVS: Dynamic Voltage Scaling.....
EA: Exponential Average.....
EC: Embedded Controller.....
ENIAC: Electronic Numerical Integrator and Computer.....
IC: Integrated Circuit.....
OSPM: Operating System Power Management.....
PACE: Processor Acceleration to Conserve Energy.....
PCAP: Program-Counter Access Predictor.....
PM: Power Manager.....
PMC: Power Manageable Component.....
SP: Service Provider.....
SQ: Service Queue.....
SR: Service Requester.....

Literaturverzeichnis / Quellenangaben

[1] C. Martin, “ENIAC: press conference that shook the world,” *Technology and Society Magazine, IEEE*, vol. 14, 1995, pp. 3-10.

[2] A.S. Tanenbaum, *Moderne Betriebssysteme*, Pearson Studium, 2002.

[3] P. Kurp, “Green computing,” *Commun. ACM*, vol. 51, 2008, pp. 11-13.

[4] “ElektroG - Gesetz über das Inverkehrbringen, die Rücknahme und die umweltverträgliche Entsorgung von Elektro- und Elektronikgeräten.”

[5] G. Hubner, *Stochastik: Eine anwendungsorientierte Einführung für Informatiker, Ingenieure und Mathematiker*, Vieweg+Teubner, 2009.

[6] “Markow-Kette – Wikipedia,” http://de.wikipedia.org/wiki/Markow-Kette, 9. 9. 2009.

[7] R.J. Baker, *CMOS: Circuit Design, Layout, and Simulation*, Wiley & Sons, 2007.

[8] M. Srivastava, A. Chandrakasan, and R. Brodersen, “Predictive system shutdown and other architectural techniques for energy efficient programmable computation,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 4, 1996, pp. 42-55.

[9] F. Douglis, P. Krishnan, and B. Marsh, “Thwarting the power-hungry disk,” *Proceedings of the USENIX Winter 1994 Technical Conference on USENIX Winter 1994 Technical Conference*, San Francisco, California: USENIX Association, 1994, pp. 23-23.

[10] K. Li, R. Kumpf, P. Horton, and T. Anderson, “A Quantitative analysis of disk drive power management in portable computers,” *Proceedings of the USENIX Winter 1994 Technical Conference on USENIX Winter 1994 Technical Conference*, San Francisco, California: USENIX Association, 1994, pp. 22-22.

[11] B. Marsh, F. Douglis, and P. Krishnan, “Flash memory file caching for mobile computers,” *System Sciences, 1994. Vol. I: Architecture, Proceedings of the Twenty-Seventh Hawaii Internation Conference on*, 1994, pp. 451-460.

[12] J. Lorch and A. Smith, “Software strategies for portable computer energy management,” *Personal Communications, IEEE*, vol. 5, 1998, pp. 60-73.

[13] Y. Lu, L. Benini, and G.D. Micheli, “Operating-system directed power reduction,” *Proceedings of the 2000 international symposium on Low power electronics and design*, Rapallo, Italy: ACM, 2000, pp. 37-42.

[14] V.L. Benini and G.D. Micheli, *Dynamic Power Management*.

[15] L. Benini, A. Bogliolo, S. Cavallucci, and B. Ricco, “Monitoring system activity for OS-directed dynamic power management,” *Low Power Electronics and Design, 1998. Proceedings. 1998 International Symposium on*, 1998, pp. 185-190.

[16] Eui-Young Chung, L. Benini, and G. De Micheli, “Dynamic power management using adaptive learning tree,” *Computer-Aided Design, 1999. Digest of Technical Papers. 1999 IEEE/ACM International Conference on*, 1999, pp. 274-279.

[17] L. Benini, A. Bogliolo, and G. De Micheli, “A survey of design techniques for system-level dynamic powermanagement,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, 2000, pp. 299-316.

[18] Eui-Young Chung, L. Benini, A. Bogliolo, Yung-Hsiang Lu, and G. De Micheli, "Dynamic power management for nonstationary service requests," *Computers, IEEE Transactions on*, vol. 51, 2002, pp. 1345-1361.

[19] R. Golding, P. Bosch, C. Staelin, T. Sullivan, and J. Wilkes, "Idleness is not sloth," *Proceedings of the USENIX 1995 Technical Conference Proceedings on USENIX 1995 Technical Conference Proceedings*, New Orleans, Louisiana: USENIX Association, 1995, pp. 17-17.

[20] F. Douglis, P. Krishnan, and B.N. Bershad, "Adaptive Disk Spin-down Policies for Mobile Computers," *Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing*, USENIX Association, 1995, pp. 121-137.

[21] Yung-Hsiang Lu and G. De Micheli, "Adaptive hard disk power management on personal computers," *VLSI, 1999. Proceedings. Ninth Great Lakes Symposium on*, 1999, pp. 50-53.

[22] Yung-Hsiang Lu and G. De Micheli, "Comparing system level power management policies," *Design & Test of Computers, IEEE*, vol. 18, 2001, pp. 10-19.

[23] Chi-Hong Hwang and A. Wu, "A predictive system shutdown method for energy saving of event-driven computation," *Computer-Aided Design, 1997. Digest of Technical Papers., 1997 IEEE/ACM International Conference on*, 1997, pp. 28-32.

[24] C. Gniady, Y. Hu, and Y. Lu, "Program counter based techniques for dynamic power management," *High Performance Computer Architecture, 2004. HPCA-10. Proceedings. 10th International Symposium on*, 2004, pp. 24-35.

[25] L. Benini, A. Bogliolo, G. Paleologo, and G. De Micheli, "Policy optimization for dynamic power management," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 18, 1999, pp. 813-833.

[26] Q. Qiu and M. Pedram, "Dynamic power management based on continuous-time Markov decision processes," *Proceedings of the 36th ACM/IEEE conference on Design automation*, New Orleans, Louisiana, United States: ACM, 1999, pp. 555-561.

[27] T. Simunic, L. Benini, P. Glynn, and G.D. Micheli, "Dynamic power management for portable systems," *Proceedings of the 6th annual international conference on Mobile computing and networking*, Boston, Massachusetts, United States: ACM, 2000, pp. 11-19.

[28] T. Simunic, L. Benini, P. Glynn, and G. De Micheli, "Event-driven power management," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 20, 2001, pp. 840-857.

[29] Yung-Hsiang Lu, Eui-Young Chung, T. Simunic, T. Benini, and G. De Micheli, "Quantitative comparison of power management algorithms," *Design, Automation and Test in Europe Conference and Exhibition 2000. Proceedings*, 2000, pp. 20-26.

[30] D. Ramanathan, S. Irani, and R. Gupta, "An analysis of system level power management algorithms and their effects on latency," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 21, 2002, pp. 291-305.

[31] D. Ramanathan, S. Irani, and R. Gupta, "Latency effects of system level power management algorithms," *Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, San Jose, California: IEEE Press, 2000, pp. 350-356.

[32] M. Samadi and A. Afzali-Kusha, "Power management with fuzzy decision support system," *ASIC, 2007. ASICON '07. 7th International Conference on*, 2007, pp. 74-77.

[33] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, “A dynamic voltage scaled microprocessor system,” *Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE International*, 2000, pp. 294-295, 466.

[34] A. Sinha and A. Chandrakasan, “Dynamic voltage scheduling using adaptive filtering of workload traces,” *VLSI Design, 2001. Fourteenth International Conference on*, 2001, pp. 221-226.

[35] J.R. Lorch and A.J. Smith, “Improving dynamic voltage scaling algorithms with <italic>PACE</italic>,” *SIGMETRICS Perform. Eval. Rev.*, vol. 29, 2001, pp. 50-61.

[36] T. Pering, T. Burd, and R. Brodersen, “Voltage scheduling in the IpARM microprocessor system,” *Low Power Electronics and Design, 2000. ISLPED '00. Proceedings of the 2000 International Symposium on*, 2000, pp. 96-101.

[37] V. Gutnik and A. Chandrakasan, “Embedded power supply for low-power DSP,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 5, 1997, pp. 425-435.

[38] T. Burd and R. Brodersen, “Energy efficient CMOS microprocessor design,” *System Sciences, 1995. Proceedings of the Twenty-Eighth Hawaii International Conference on*, 1995, pp. 288-297 vol.1.

[39] F. Xie, M. Martonosi, and S. Malik, “Compile-time dynamic voltage scaling settings: opportunities and limits,” *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, San Diego, California, USA: ACM, 2003, pp. 49-62.

[40] F. Xie, M. Martonosi, and S. Malik, “Bounds on power savings using runtime dynamic voltage scaling: an exact algorithm and a linear-time heuristic approximation,” *Proceedings of the 2005 international symposium on Low power electronics and design*, San Diego, CA, USA: ACM, 2005, pp. 287-292.

[41] K. Flautner, S. Reinhardt, and T. Mudge, “Automatic Performance Setting for Dynamic Voltage Scaling,” *Wireless Networks*, vol. 8, 2002, pp. 507-520.

[42] K. Govil, E. Chan, and H. Wasserman, “Comparing algorithm for dynamic speed-setting of a low-power CPU,” *Proceedings of the 1st annual international conference on Mobile computing and networking*, Berkeley, California, United States: ACM, 1995, pp. 13-25.

[43] K. Flautner and T. Mudge, “Vertigo: automatic performance-setting for Linux,” *SIGOPS Oper. Syst. Rev.*, vol. 36, 2002, pp. 105-116.

[44] H. Aydi, P. Mejia-Alvarez, D. Mosse, and R. Melhem, “Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems,” *Proceedings of the 22nd IEEE Real-Time Systems Symposium*, IEEE Computer Society, 2001, p. 95.

[45] P. Pillai and K.G. Shin, “Real-time dynamic voltage scaling for low-power embedded operating systems.”

[46] C. Krishna and Y. Lee, “Voltage-clock-scaling adaptive scheduling techniques for low power in hard real-time systems,” *Real-Time Technology and Applications Symposium, 2000. RTAS 2000. Proceedings. Sixth IEEE*, 2000, pp. 156-165.

[47] T. Okuma, T. Ishihara, and H. Yasuura, “Real-time task scheduling for a variable voltage processor,” *System Synthesis, 1999. Proceedings. 12th International Symposium on*, 1999, pp. 24-29.

[48] M. Weiser, B. Welch, A. Demers, and S. Shenker, “Scheduling for reduced CPU energy,” *Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation*, Monterey, California: USENIX Association, 1994, p. 2.

[49] T. Pering, T. Burd, and R. Brodersen, “The simulation and evaluation of dynamic voltage scaling algorithms,” *Proceedings of the 1998 international symposium on Low power electronics and design*, Monterey, California, United States: ACM, 1998, pp. 76-81.

[50] D. Grunwald, I.I.I. Charles B. Morrey, P. Levis, M. Neufeld, and K.I. Farkas, “Policies for dynamic clock scheduling,” *Proceedings of the 4th conference on Symposium on Operating System Design \& Implementation - Volume 4*, San Diego, California: USENIX Association, 2000, pp. 6-6.

[51] J.R. Lorch and A.J. Smith, “Scheduling techniques for reducing processor energy use in MacOS,” *Wirel. Netw.*, vol. 3, 1997, pp. 311-324.

[52] T. Simunic, L. Benini, A. Acquaviva, P. Glynn, and G.D. Micheli, “Dynamic voltage scaling and power management for portable systems,” *Proceedings of the 38th conference on Design automation*, Las Vegas, Nevada, United States: ACM, 2001, pp. 524-529.

[53] V. Delaluz, M. Kandemir, N. Vijaykrishnan, A. Sivasubramaniam, and M.J. Irwin, “DRAM Energy Management Using Sof ware and Hardware Directed Power Mode Control,” *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, IEEE Computer Society, 2001, p. 159.

[54] A.S. Tanenbaum and J.R. Goodman, *Structured Computer Organization.*, Prentice Hall International, 2001.

[55] L. Benini and G.D. Micheli, “System-level power optimization: techniques and tools,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 5, 2000, pp. 115-192.

[56] F. Bellosa, “The benefits of event: driven energy accounting in power-sensitive systems,” *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, Kolding, Denmark: ACM, 2000, pp. 37-42.

[57] X. Fan, C. Ellis, and A. Lebeck, “Memory controller policies for DRAM power management,” *Proceedings of the 2001 international symposium on Low power electronics and design*, Huntington Beach, California, United States: ACM, 2001, pp. 129-134.

[58] S. Wuytack, F. Catthoor, and H. De Man, “Transforming set data types to power optimal data structures,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 15, 1996, pp. 619-629.

[59] J. Julio L. da Silva, F. Catthoor, D. Verkest, and H.D. Man, “Power exploration for dynamic data types through virtual memory management refinement,” *Proceedings of the 1998 international symposium on Low power electronics and design*, Monterey, California, United States: ACM, 1998, pp. 311-316.

[60] A.R. Lebeck, X. Fan, H. Zeng, and C. Ellis, “Power aware page allocation,” *Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, Cambridge, Massachusetts, United States: ACM, 2000, pp. 105-116.

[61] Ching-Long Su, Chi-Ying Tsui, and A. Despain, “Low power architecture design and compilation techniques for high-performance processors,” *Compcon Spring '94, Digest of Papers.*, 1994, pp. 489-498.

[62] H. Zhang and J. Rabaey, “Low-swing interconnect interface circuits,” *Proceedings of the 1998 international symposium on Low power electronics and design*, Monterey, California, United States: ACM, 1998, pp. 161-166.

[63] C.H. Gebotys, “Low energy memory and register allocation using network flow,” *Proceedings of the 34th annual conference on Design automation*, Anaheim, California, United States: ACM, 1997, pp. 435-440.

[64] P.R. Panda and N.D. Dutt, “Reducing Address Bus Transitions for Low Power Memory Mapping,” *Proceedings of the 1996 European conference on Design and Test*, IEEE Computer Society, 1996, p. 63.

[65] Mike Tien-Chien Lee, V. Tiwari, S. Malik, and M. Fujita, “Power analysis and minimization techniques for embedded DSP software,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 5, 1997, pp. 123-135.

[66] Z. Hu, S. Kaxiras, and M. Martonosi, “Let caches decay: reducing leakage energy via exploitation of cache generational behavior,” *ACM Trans. Comput. Syst.*, vol. 20, 2002, pp. 161-190.

[67] J. Flinn and M. SATYANARAYANAN, “Energy-aware adaptation for mobile applications,” *Proceedings of the seventeenth ACM symposium on Operating systems principles*, Charleston, South Carolina, United States: ACM, 1999, pp. 48-63.

[68] D.P. Helmbold, D.D.E. Long, and B. Sherrod, “A dynamic disk spin-down technique for mobile computing,” *Proceedings of the 2nd annual international conference on Mobile computing and networking*, Rye, New York, United States: ACM, 1996, pp. 130-142.

[69] A.E. Papathanasiou and M.L. Scott, *Increasing Disk Burstiness for Energy Efficiency*, University of Rochester, 2002.

[70] T. Heath, E. Pinheiro, J. Hom, U. Kremer, and R. Bianchini, “Application transformations for energy and performance-aware device management,” *Parallel Architectures and Compilation Techniques, 2002. Proceedings. 2002 International Conference on*, 2002, pp. 121-130.

[71] Dell Computer Corp., “Dell System 320SLi User Guide,” 1992.

[72] P. Krishnan, P.M. Long, and J.S. Vitter, “Adaptive Disk Spindown via Optimal Rent-to-Buy in Probabilistic Environments,” *Algorithmica*, vol. 23, Jan. 1999, pp. 31-56.

[73] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke, “DRPM: dynamic speed control for power management in server class disks,” *SIGARCH Comput. Archit. News*, vol. 31, 2003, pp. 169-181.

[74] Q. Zhu, F.M. David, C.F. Devaraj, Z. Li, Y. Zhou, and P. Cao, “Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management,” *Proceedings of the 10th International Symposium on High Performance Computer Architecture*, IEEE Computer Society, 2004, p. 118.

[75] M. SATYANARAYANAN, J.J. Kistler, L.B. Mummert, M.R. Ebling, P. Kumar, and Q. Lu, “Experience with disconnected operation in a mobile computing environment,” *Mobile \& Location-Independent Computing Symposium on Mobile \& Location-Independent Computing Symposium*, Cambridge, Massachusetts: USENIX Association, 1993, pp. 2-2.

[76] A. Papathanasiou and M. Scott, “Energy efficiency through burstiness,” *Mobile Computing Systems and Applications, 2003. Proceedings. Fifth IEEE Workshop on*, 2003, pp. 44-53.

[77] S. Singh, M. Woo, and C.S. Raghavendra, “Power-aware routing in mobile ad hoc networks,” *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, Dallas, Texas, United States: ACM, 1998, pp. 181-190.

[78] A. Vahdat, A. Lebeck, and C.S. Ellis, “Every joule is precious: the case for revisiting operating system design for energy efficiency,” *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, Kolding, Denmark: ACM, 2000, pp. 31-36.

[79] R. Kravets and P. Krishnan, “Power management techniques for mobile communication,” *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, Dallas, Texas, United States: ACM, 1998, pp. 157-168.

[80] A. Rudenko, P. Reiher, G.J. Popek, and G.H. Kuenning, “The remote processing framework for portable computer power saving,” *Proceedings of the 1999 ACM symposium on Applied computing*, San Antonio, Texas, United States: ACM, 1999, pp. 365-372.

[81] S. Chandra, C.S. Ellis, and A. Vahdat, “Managing the storage and battery resources in an image capture device (digital camera) using dynamic transcoding,” *Proceedings of the 3rd ACM international workshop on Wireless mobile multimedia*, Boston, Massachusetts, United States: ACM, 2000, pp. 73-82.

[82] “ACPI - Advanced Configuration and Power Interface.”

[83] S. Balakrishnan and J. Ramanan, “Power-aware operating system using acpi.”

[84] A. Grover, “Modern System Power Management,” *Queue*, vol. 1, 2003, pp. 66-72.

[85] “ACPICA - ACPI Component Architecture,” http://www.acpica.org/, 5. 5. 2009.

[86] “Documentation,” http://www.kernel.org/doc/Documentation/, 9. 9. 2009.

[87] “acpid - the ACPI daemon,” http://acpid.sourceforge.net/, 9. 9. 2009.

[88] “acpid(8) - Linux man page,” http://linux.die.net/man/8/acpid, 9. 9. 2009.

[89] “hdparm | Get hdparm at SourceForge.net,” http://sourceforge.net/projects/hdparm/, 9. 9. 2009.

[90] “Laptop Mode Tools,” http://samwel.tk/laptop_mode/, 7. 7. 2009.

[91] “GNOME: The Free Software Desktop Project,” http://www.gnome.org/, 7. 7. 2009.

[92] “Gentoo Linux -- Gentoo Linux News,” http://www.gentoo.org/, 7. 7. 2009.

[93] “Gentoo Linux – Wikipedia,” http://de.wikipedia.org/wiki/Gentoo_Linux, 9. 9. 2009.

[94] “Gentoo Linux -- About Gentoo,” http://www.gentoo.org/main/en/about.xml, 9. 9. 2009.

[95] “Gentoo-Portage - News,” http://gentoo-portage.com/, 9. 9. 2009.

[96] “YaST - openSUSE,” http://de.opensuse.org/YaST#WWW-Verkn.C3.BCpfungen, 9. 9. 2009.

[97] “SourceForge.net: - Project Web Hosting - Open Source Software,” http://hdparm.sourceforge.net/, 7. 7. 2009.

[98] “RFC 5424 - The Syslog Protocol,” http://tools.ietf.org/html/rfc5424, 9. 9. 2009.

[99] “GNU Xnee - GNU Project,” http://www.gnu.org/software/xnee/, 7. 7. 2009.

[100] “OProfile - A System Profiler for Linux (News),” http://oprofile.sourceforge.net/news/, 9. 9. 2009.

[101] “SYSSTAT,” http://pagesperso-orange.fr/sebastien.godard/, 7. 7. 2009.

[102] “HardwareLiSter - ezIX,” http://ezix.org/project/wiki/HardwareLiSter, 9. 9. 2009.

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 18. September 2009 _____

Ort, Datum Unterschrift