Energy-Efficient Applications

Seminar-Proceedings

edited by

Andreas Winter
Jan Jelschen
Carl von Ossietzky Universität Oldenburg,
Software Engineering

OLNSE Number 3/2012

April 2012

Oldenburg Lecture Notes in Software Engineering

Oldenburg Lecture Notes
on Software Engineering (OLNSE)
Carl von Ossietzky University Oldenburg
Department for Computer Science
Software Engineering
26111 Oldenburg, Germany

Introduction

Energy-efficiency has become an important issue in information and communication technology. According to a 2007 report of German research organization Fraunhofer [SNP+09], over ten percent of Germany's overall electrical energy consumption of 2007 was generated by the information and communication technology sector. By 2020, this figure is prediced to have risen to over 20 percent, which, with unchanged energy mix, would account for CO_2 emissions exceeding that of the entire German aviation sector [NHSS09].
Energy consumption in *Mobile Computing* poses a challenge. Due to the rapidly increasing adoption of smart phones, tablet PCs, and other mobile devices, there is also an increasing demand for higher battery capacities. However, advances in battery technology seem to move at a rather slow pace when compared to rapidly evolving mobile devices [BWV04].
Additionally, facing global warming and the reality of ever more apparent climate changes [GeS08],
The most direct approach to conserve energy in IT probably is trying to build more efficient hardware.
Lots of work has already been done in the area of energy efficiency in IT mostly driven by research in embedded systems and in wireless sensor networks.
Improving energy efficiency of software systems can be approached on different levels. A great amount of research focuses on low-level optimizations (e.g. machine code level) [RJ97].
Energy efficiency for higher software layers, in particular for the software application level, has not been addressed deeply, yet. In this paper, first ideas on saving energy on the end-user application and operating system level by applying *reengineering services* are presented. Analyzing both code as well as execution behavior and patterns of applications will show opportunities for optimization. With such information, code can be restructured to utilize hardware resources more efficiently, and the operating system is enabled to schedule application execution complying with their run-time needs and a possibly system wide energy efficient behavior.
These activities – analyzing software, then making improvements to it with the gained information and knowledge – are basically the same as in software reengineering. Reengineering aims at improving a program's quality in terms of maintainability, code quality, or similar goals [CC90]. This research agenda argues software reengineering tools and techniques, like *static and dynamic program analysis*, and *systematic code transformations like refactoring*, can be used to obtain more energy efficient applications.
To investigate this topic further a research seminar was held in winter semester 2011 / 2012. From this seminar three student papers emerged as well as an additional pro-seminar work. The first one is "Specialized Processors for Energy-Efficient Use of Applications" by Marion Gottschalk which focuses on recent processor development for increasing the energy-efficiency of applications. The second one is "Energy Aware Computing" by Cosmin Pitu which gives an overview about development in the field of energy aware computing. The last regular papar is "Towards an Energy Abstraction Layer" by Mirco Josefiok which aims at proposing an abstract energy measurement platform for mobile computing

devices. The pro-seminar work by Michel Falk focuses on evaluating options for energy efficient development on Android devices.
Preparation for the seminar resulted in an overview paper motivating research on efficient software applications. This paper sketched software-evolution methods and techniques to support software analysis to reengineer, i.e. improve the energy consumption of software systems. This paper [JGJ+12] was accepted for and presented at the ERA-Tack of the European Conference on Software Maintenance and Reengineering in Szeged, Hungary in March 2012

References

[BWV04] Lawrence S. Brakmo, Deborah A. Wallach, and Marc A. Viredaz. usleep: A technique for reducing energy consumption in handheld devices. In *IN PROC. INT. CONF. MOBILE SYSTEMS, APPLICATIONS, AND SERVICES*, pages 12–22, 2004.

[CC90] Elliot J. Chikofsky and James H Cross, II. Reverse engineering and design recovery: A taxonomy. *IEEE software*, 7(1):13–17, 1990.

[GeS08] GeSI. SMART 2020 Addendum Deutschland: Die IKT-Industrie als treibende Kraft auf dem Weg zu nachhaltigem Klimaschutz GeSI 2008, 2008.

[JGJ+12] Jan Jelschen, Marion Gottschalk, Mirco Josefiok, Cosmin Pitu, and Andreas Winter. Towards applying reengineering services to energy-efficient applications. In Rudolf Ferenc, Tom Mens, and Anthony Cleve, editors, *Proceedings of the 16th Conference on Software Maintenance and Reengineering*. IEEE, 2012.

[NHSS09] Wolfgang Nebel, Marko Hoyer, Kiril Schröder, and Daniel Schlitt. Untersuchung des Potentials von rechenzentrenübergreifendem Lastmanagement zur Reduzierung des Energieverbrauchs in der IKT. Studie für das bundesministerium für wirtschaft und technologie, OFFIS, Oldenburg, 2009.

[RJ97] Kaushik Roy and Mark C. Johnson. Software design for low power. In Wolfgang Nebel and Jean P. Mermet, editors, *Low power design in deep submicron electronics*, pages 433–460. Springer, Berlin, 1997.

[SNP+09] L. Stobbe, N.F. Nissen, M. Proske, A. Middendorf, B. Schlomann, M. Friedewald, P. Georgieff, and T. Leimbach. Abschätzung des Energiebedarfs der weiteren Entwicklung der Informationsgesellschaft. Abschlussbericht an das bundesministerium für wirtschaft und technologie, Fraunhofer ISI and IZM, Berlin, 2009.

Contents

Specialized Processors for Energy-Efficient Use of Applications

Marion Gottschalk
Carl von Ossietzky University Oldenburg, Germany
Department of Computer Science
marion.gottschalk@informatik.uni-oldenburg.de

***Abstract*—The performance of processors has increased dramatically since the first processor was developed in 1969. At the same time processors perform more complex functions than before. This is important for adoption of mobile devices to achieve requirements of users. However, performance has its price: high power consumption. Due to the basis of complex functions, which a processor should comprised, much power is used. This power comes from the battery of mobile devices. Due to increasing performance of processors, battery lifetime must be increased too, to supply processors with energy. Another approach is to optimize processors, so that the same functions could be perform as before, but with a decreased power consumption. The processor optimize should be the main theme in this work. Therefore, different processor types with main focus on specialized processors will be described.**

***Index Terms*—energy-efficient; special processors; ASIP; AFU; DSP; multi-core processors;**

I. Motivation

In the past mobile phones have been used only for calling and sending text messages. The development of mobile phones to smart-phones has increased dramatically the volume of functions, because current smart-phones are a combination of different mobile devices. Hence, the smart-phone, beside for calling, is also used for receiving and writing e-mails, for managing and synchronizing dates and for navigating. From these new applications the wish for more performance and longer battery lifetime by users originates [1]. One approach to accomplish this task, is to refine processors, because this component of the system has a high power consumption which is shown by Gelsinger [2].

For more than 40 years the industry has depended on *Moore's Law*, i.e. processors manufacture intent to double the number of transistors of a processor within 18 till 24 months, to increase performance. This aim has changed in the last 10 years. The significance for the energy-efficient of processors has raised in the awareness of users and producer. So the interest, in low power consumption has increased [1], [2]. The change from users desire for exclusively more performance toward longer battery lifetime is related to the before mentioned development of mobile phones to smart-phones, because users want to be as independent of energy sources as possible.

Due to the battery lifetime is not developed as quickly as processors and the applications, so it becomes increasingly difficult to improve battery lifetime [3]. According to [4], the majority power consumption can be partitioned on components *GSM module* and *display*. Hence, a power management system for smart-phones would be useful to minimized power consumption [4]. However, processors are also significant components, which must be considered because its always used and its known that more power is used as scheduled [2]. This problem is an interesting theme and it is described in the next sections.

A new solution, to reduce power consumption of processors, is the *3D-Transistors* also called *Tri-Gate-Transistor* [5]. This technique is characterized by the gate, which surrounds the power channel on three sides instead of one side. So the current flow into the transistor can produce and isolate faster. Thus working voltage and adverse effects, like leakage II-B, is saved and processors work more energy-efficient. *Energy-efficient* means that the energy is used as effectively as possible to be transformed into performance [1]. The new solution of *3D-Transistors* is described in Kaminski [5].

Another idea to create an energy-efficient processor, is the specialization of processors. Although combination of processor, like in [6], and modification of instruction set of processors, like in [7], can be differentiated. Here, specialized processors are created to solve a special task of an application. It is also possible that applications can be adapted to specialized processors [8].

These different approaches as well as a number of others are described in this work. First of all, the energy problematic of processors is described in II. This section represents the reason for approaches of modification of processors. In Section III different types of processors are characterized. After this the different approaches of specialized processors are described in IV. In V the options to reduced power consumption of applications by specialized processors are contained. There also a comparison to previously described problems of energy consumption, to clarified effectiveness of approaches. In addition in VI some other techniques of processors development such as specialized processors are described. A result and outlook are given in VII.

II. Problematics of energy consumption of a processor

This work focuses on approaches, to reduce power consumption of mobile devices. The component *processor* was choosen, because processors have a high power consumption

e.g. by leakage [2]. First, the design of processors and their *case* are considered, in order to recognize where problems may arise due to high power consumption. The energy problematic will be described after this.

A. Design of a processor

The processor is a component, which is built up of many small circuits, called transistors. These transistors are combined to different gates, which implement simple logical functions. With a variety of gates, which can contain about 3 billion transistors [9], it is possible to implement complex functions, as required in mobile devices required.

The Figure 1 shows a processor, schematically case and components regulate the temperature of processors.

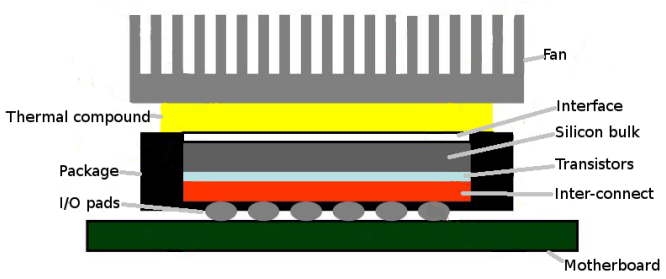

Fig. 1. Design of the processor
Extracted from Lecture LESD 2011

The proper functional area in figure 1 is the blue area. Here, the transistors are placed, which run applications. This area contains the transistors, cache, register-set, control unit and system interface [10]. In the red area, below the blue, inter-connector is integrated, which provides connection with the I/O-Pads, the gray ellipses, to the motherboard, the green area. By using this interface, processors receive tasks, to be done to run an application, and return results due to this interface. The gray area above the transistors shows the silicon-bulk, which is the channel for electrons. The black area around the silicon-bulk, the transistors and the inter-connector is the case of the processor, which protected the processor and dissipated the heat outside. The white area on the case shows the material interface, on which a fan is mounted. Between fan and material interface thermal compound is located, yellow area, for a better transfer of heat outside of the processor.

B. Problems of energy consumption

With current technology, power consumption of processors increases with use of multiple transistors per processor. While performance increases linearly, power consumption increases exponentially [2]. However, this depends not only on the number of circuits in processors, which required more power, but e.g. on the *leakage*. *Leakage* denotes e.g. power consumption of one transistor, if its switched-off [11]. Although no activity emanates from transistor, power flows and power consumption of processors is rising faster, if more and more transistors are integrated. In [11] more different leakage are describes, which occur during current time or when switching off the transistor. But to illustrate problems, it should suffice to know in this paper, that while a processor is idle and so many transistors not working, power still flows. A solution to this problem or improvement are above-mentioned *3D-Transistors* [5].

Another problem is heat generation in processors, described in [12]. The heat generated by converting energy [1], through the circuits of transistors, and must be removed. This is about different cooling systems and the *case* of the processor attempts [1]. The most widespread cooling system is the fan, because of this the case of processors are connected to this system, see figure 1. The removal of heat is very important, because of the silicon-bulk aging or can even be destroyed [1], [12]. The removal of heat requires power, e.g. to operate the fan. This factor must also be included in the calculation of power consumption of processors, which casts the development of processors regarding the increase in the number of transistors looks inefficient.

The above described problems are resulted partly from the absence of information when the processor can be switched-off. This problem can be solved on software side through an energy abstraction layer, so it will minimize problems on hardware side. However, to solve this problems software and hardware must be optimized together [13]. In the next sections options on hardware side will be considered.

III. Type of processors

According to [7] processors are differentiated in three types. These types are shown in figure 2. On the left side *dedicate processor* and on the right side *standard processor*. Between these both types the *special processor* is shown. In this section these three types are described briefly, to classified the specialized processors in next sections.

A. Dedicated processor

The dedicate processor is a processor, which is specialized for only one application and therefore has no room for additional functions. This type of processors is optimal for one application, because the speed of applications is maximal and power consumption is lowest. However, dedicate processors can not run other applications, and is rarely used due to its lack of flexibility [7]. The dedicated processors is most used as co-processor. For example in [14] a dedicated co-processor executes an encoding algorithm, which can reduces power consumption up to 40 % for an application compared to a standard processor. Decreasing switching activity of transistors is one reason for this.

B. Specialized Processor

The *specialized processors* are distinguished in different types. First, processors can be changed in their instruction set, so this processor has an advantage for certain applications. This is achieved e.g. by *Application-Specific Functional Units (AFUs)*. The AFUs allow more flexibility compared to standard processors, increased speed of processors and reduced

[1]Lecture *Low Energy Systems Design* WS 2011/12 slide 21

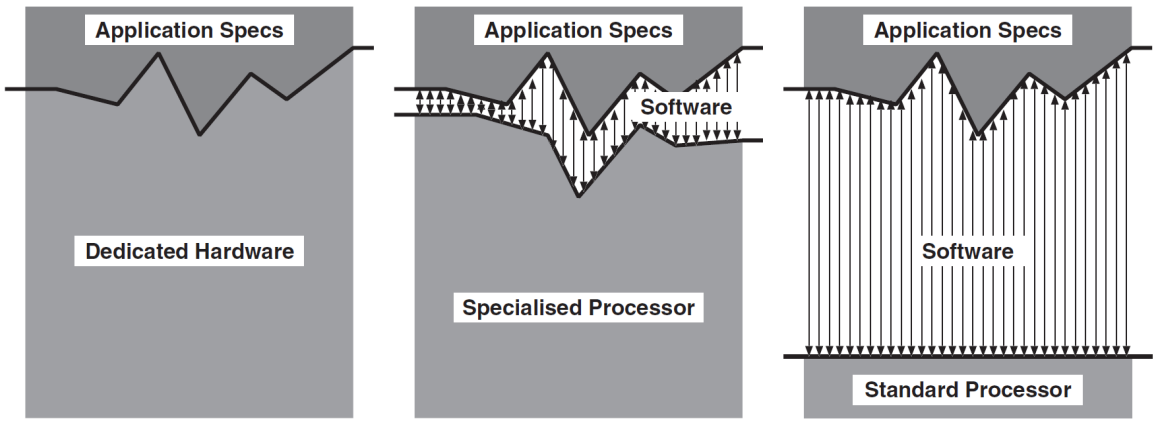

Fig. 2. Types of processors
Extracted from [7]

power consumption [7]. The middle visualization in Figure 2 shows an AFU. Secondly, in [1] same or different cores are integrated in one processor. This complete tasks can be performed on different cores to increase speed of execution of applications and thus reduce power consumption. So a specialized processor can be variable in its composition. In the following section IV more details and examples of specialized processors are given.

C. Standard processor

The standard processor consists of a default hardware, so that every application can be run on this processor. This processor is very flexible. That is shown in Figure 2 by the area *Software*. This means that much software is needed to implement applications on standard processors, and that it is flexible for different applications. However, standard processors are the slowest processors with highest power consumption compared to others [7]. A standard processor is an CPU (Central Processing Unit) which is used in a desktop-pc [15].

IV. Special processors

Here, specialized processors are divided into different types. In this paper specialized processor will be classified in three types. Thereby, *Modification of the instruction set*, *Modification of the design* and *Multi-core processors* are differentiated. By describing these types always an appeal to the power consumption is assembled. The presentation of specialized processors are certainly not complete, but should give a overview of their possibilities.

A. Modification of the instruction set

Different approaches exist to change the instruction set of processors and to reduce power consumption. In this section some specialized processors are described, in which not all specialized processors to be integrated in mobile devices, but in desktop-pc and server. Hence, this list is given an overview of general instances of specialized processors and shows the width of research area.

DSP The acronym DSP stands for *Digital Signal Processor*. The DSP can handle multiple tasks simultaneously due to its architecture (Harvard architecture). The standard processor would have to run several cycles to perform same tasks [16]. These tasks are mathematical calculations that can be performed in real time [17]. The parallelization is often beneficial in repetitive tasks such as decoding of MP3 files, therefore the DSP is used in MP3 players. The elements of DSP are sometimes used in a number of standard processors, as more and more multimedia content on smart-phones such as HDTV and 3D games need to be processed [18]. For these type of applications, the DSP is preferable due to its more energy-efficient operation than a standard processor [16].

GPU The GPU (Graphics Processing Unit) is a processor for visualization of videos and 3D-games. On GPU by parallelization of multiple tasks data- and calculation-intensive application are executed, like on DSP [19]. In [19] Tesla-GPUs of nVIDIA are used in combination with standard processors for supercomputer. This combination enables the more energy-efficient use of supercomputers, because applications will be executed according to their function on corresponding processors.

SPU The acronym SPU stands for *Sound Processing Unit*. The SPU is like GPU an additional processor for standard processors, to parallelize different tasks of applications [20]. On SPU audio signals are processed, i.e. decode MP3 files or processing of recording a microphone [20].

IPU The acronym IPU stands for *Image Processing Unit*. The IPU is responsible for the image processing and display management, e.g. color space conversion and to aggregate graphics and videos. This processor also can be combined with standard processors, because its equipped with powerful control and synchronization resources to perform tasks with minimal power [21], but only tasks for image processing.

According to description of specialized processors with a modified instruction set several instances different categories

of processors will be presented, which use different instance of processors. This list also is not complete but gives an overview of some specialized processors.

ASIP The acronym ASIP stands for *Application-Specific Instruction-set Processors*. It describes a processor, whose architecture has been changed [22]. In this case transistors on processor can be adjusted the requirement of applications [23]. This field will be partly automatically created for a special application, according to predefined metrics, e.g. runtime [22], and can be used for a special application [7]. In [23] is described a tool which can be analyzed applications and help the programmer to decide how the digital circuits of ASIP be programmed. The tool detects e.g. which data can be parallelized in execution but programmer must be decided.

AFU A processor is enhanced by AFU (Application-Specific Functional Unit), i.e. besides normal instruction set is a command instruction set, which can be adapted for a special application [7]. This additional instruction set can be determined after production of processors, to create customization and cost-efficiency, because it can be used in many areas. The AFU is a CPU with a ASIP unit. In [7] an algorithm is described to detect automatically separate data flows in an application, which can be parallelized on digital circuits of AFU. Tests have shown that algorithm of [7] has more performance as algorithm of Clubbing [24] and MaxMISO [25].

Microprocessor A microprocessor in general is a standard processor but also it can be a ASIP or AFU, whose components are all integrated on a microchip [1]. Two different architectures can be distinguished: CISC (Complex Instruction Set Computer) and RISC (Reduced Instruction Set Computer). RISC architecture has a small instruction set, but an advanced hardware compared to CISC architecture, because different from CISC, RISC has a hardware multiplier [10]. Due to reduced instruction set and therefore quicker switching of transistors, the microprocessor of smart-phones are designed in RISC architecture [26]. In addition IBM found that two thirds of machine code are only ten different instructions. Therefore, there is no need for the complex instructions as by CISC [10].

Mobile processor A mobile processor is a microprocessor, that is created specifically for mobile devices, such as smart-phones and tablet-PCs [27]. The most current mobile processor are combined processors which are described in Section VI-B, however some processors should be mentioned here. An example of a mobile processor is described in [28]. The mobile processor of nVIDIA consists of five cores, whereas four cores are needed for complex applications e.g. 3D-games. The fifth core is used for the standby mode, so that power consumption can be reduced at this time, because the other four cores are switch-off. Also Intel creates mobile processors, which is described in [29]. This mobile processor is a combination of a standard processor with a GPU. In [30] mobile processors of ARM are described. This processor combines three different processor CPU, GPU and DSP in which the CPU is a multi-core processor like in Section IV-C. A new approach of ARM processor is asynchrony work between CPUs. Hence, power consumption decreases between 25 % to 40 %. A demonstration of processors is shown in Figure 3. In this figure can be seen that DSP is used as WiFi module. In addition, it is evident that all cores use an united memory, to have access to reciprocal results. So there are possible different combinations in mobile processors.

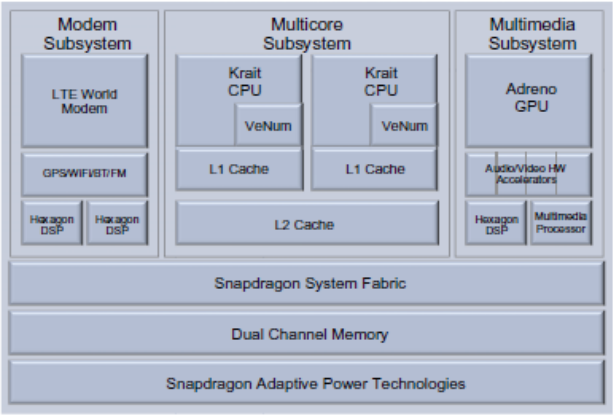

Fig. 3. Structure of a mobile Processor
Extracted from [30]

The most used instances of processors appear to be DSP and GPU. The considered mobile processors here, are mostly multi-core processor or combination of processors which use DSP and GPU [28]–[30]. The other specialized processors appear to be for devices like a MP3 player [20].

B. Modification of the design

Besides the above-mentioned specialized processors, there is the possibility of reducing the power consumption of a processor, by changing the design of the processor, e.g. to reduce the leakage [31]. For this purpose the both approaches *Multi-thresholding* [31] and *Material* [32] are described.

For multi-thresholding exist e.g. the possibility, that high or low thresholds voltage transistors are operated or individual transistors are built into a processor, which can be operated at different thresholds. The first possibility allows, to switch between two different threshold voltages. Depending on how many computing power at the time is required, threshold voltage can be set high to reduced the leakage, which causes the speed of the processor to decrease. If threshold is lowered, this increases computing power again, but also leakage and power consumption [31]. The other possibility uses various types of transistors, which are differentiate by different threshold voltages. Depending on tasks of applications, the effective area of processors will be choosen [1], [31]. Due to multi-thresholding on a processor leakage reduction from 75 % to

90 % can be achieved [33]. If considering that about 50 % of power consumption are leakages, this is a massive energy saving.

The second approach is changing materials of processors e.g. for silicon-bulk. New components in silicon-bulk can weaken leakage [1], [32]. A specialized processor for extreme application areas, which consists of silicon-carbide-bulk, can bear temperatures up to 600 degrees and survive with an additional inverter leakage at near zero [32]. Another approach is using molybdenite to produce transistors [34]. The material is less voluminous as silicon and transmits as well as silicon. Therefore, this material produces significantly smaller and energy-efficient processors because a lower voltage is needed, to switch transistors.

Not only the silicon-bulk can be changed, but also the construction of transistors [5]. As mentioned in Section I, the gate is isolated on three sides instead of one side. Due to better isolation the same performance with 0,8 V are provided instead of 1 V. Hence, energy-saving due to reduce leakage and lower voltage [35].

C. Multi-core processor

A *multi-core processor* is a composite of multiple cores in one processor. As further modifications of processor architectures could not contributed to improve performance, multi-core processors are designed, so that the rule of *Moore's Law* can continue to be complied. The tasks can be parallelized on a multi-core processor, allowing execution of such applications to be accelerated and clock rate to be decreased because different tasks run in one clock rate [1]. Due to reduced clock rate, heat generation and power consumption decline [9]. Precondition for parallelization of an application is to create a parallel machine code from code of applications [9]. One problem is, that code of existing applications has to be adapted, to exploit multi-core processor useful [36].

A specialized processor, which is a multi-core processor with 512 cores and used in desktop-PC's, is e.g. *nVIDIA Fermi-Grafic processor* [37]. Because of the cores on GPU use the same cache, the performance is increased and power saved by reduced communication between transistors [38].

Another specialized processor is MPSoC (Multiprocessor System-on-Chip), which is used in smart-phones. Here heterogeneous processors are combined [9]. In [39] the approach is described. MPSoC consists of several processors, which were each specialized for certain areas, e.g. for encrypting or encoding of videos (see IV-A). This allows the processor architecture to be adapted to the application, i.e. the area of processors can be switched-off, if its not needed for this application. This flexibility reduces power consumption of applications. In addition, this approach can be used for real-time calculation of applications. The problem with MPSoC is the high programming effort. Precise knowledge about processors are required, to know which areas of processors are addressed or can be switched-off, to reduced power consumption.

MAP (Multimedia applications processors) is an example for MPSoC. In [21] different heterogeneous processors are described, e.g. an IPU and a GPU for multimedia applications are contained on one micro-chip.

So far the described multi-core processors based on standard processors, but there are also multi-core processors with DSP, which is described in [17]. A multi-core processor is a homogeneous processor, if its only consists of DSP. For calculation of complex function in smart-phones this multi-core processor is advantageous, because the use of multimedia and the desire of real-time execution always increased [17]. Due to multiple processors, e.g. 3G, WiMAX and 3D-Games are executed by each DSP, which increases performance and reduced power consumption of a smart-phone.

V. Energy savings by special processors

After description of different specialized processors, here their options to reduced power consumption will be summarized. This section gives an overview and makes reference to II-B, to compare approaches with energy problems of processors.

So to reduced power consumption of processors, first it is useful to create a processor for a special task, so that only a few circuits between transistors occur and the way between transistors, which communicate, is minimized [39]. Opportunities for specialization of processors were named in IV-A. With this option the heat generation, which is one problem of processors, is minimized.

Since developing of specialized processors is advanced, it is possible to combine these specialized processors on one micro-chip. So processors which are required for the running of an application, can be used, while the other processors are switched-off. This also allows running of parallel tasks [17], [39], [40] and reduced power consumption by short communication distances.

The specialization and combination of processors leads to standardization of data from applications, e.g. multimedia and communication, to processors which processes this [39]. This standardization could minimizes production costs and increases budget to produce better specialized processors. However, specialized processors must provides enough flexibility to allow changes in applications [39].

Another option to reduce power consumption, is to minimized leakages. There is the possibility to create new materials for processors [1], [32] and multi-thresholding [1], [31], which are described in IV-B. Also new engine, like 3D-Transistors, minimized leakages and power consumption, described in [5].

This options are also realized for smart-phones, to increased battery lifetime of devices. In [28] and [29] this options have been described.

VI. Other ways to reduce energy consumption

In addition to development and modification of processors, there are other approaches to enhance performance of processors and to reduce their power consumption. For that purpose the *Working voltage and clock rate* and *Processor combination* will be described here.

A. Working voltage and clock rate

Another approach to reduce power consumption of processors is achieved for example by adjustments to working voltage and clock rate of processors, described in [1]. For this intent the *Advanced Control and Power Interface (ACPI)* was created. ACPI gives the system, the possibility to move components of system in an energy-saving state. Here, states of processors will be considered separately. In addition, to states (C0 to C3), performance stages (P0-PN) are considered. If processors change the state of C0 to C1, areas of processor are switched-off, so the working voltage gets reduced. Changes, however, only the performance state of processors from P0 to P1, clock rate are minimized and power consumption reduced. By the help of ACPI it is possible to control power consumption of processors, during ongoing operation [1]. Thus, ACPI can be used effectively, could first be determined with help of user profiles, when it makes the most sense to switch the processor state.

This method is named *Speedstep* [41] by Intel and named *Cool'n'Quiet* [42] and *PowerNow!* [43] by AMD. At *Speed-Step* the clock rate and working voltage of processors are adapted for ACPI [41]. This mechanism was created for Notebooks and can decide between *power mode* and *battery mode*, so that performance of processors decreases if the notebook is in *battery mode*. Meanwhile, more gradations do exist [44], as described in ACPI. At *Cool'n'Quiet* and *PowerNow!* the same mechanism are used as in *SpeedStep*. One distinction is made. The *Cool'n'Quiet* is created for desktop-pc and *PowerNow!* is created for notebooks [42], [43].

B. Processor combination

Another approach, to reduce the power consumption of applications is the processor combination whereas heterogeneous and homogeneous processors are differentiated. In heterogeneous different processors are combined and in homogeneous the same processors are combined. ARM shows in [6] that a powerful processor can be combined with a weak processor. The *Cortex-A7 Processor* involves two mobile processors with different performance. This has the advantage, that depending on complexity of applications, one processor can be selected while the other processor is switched-off. So powerful processor, which also requires more power, is only used when really needed. Tests have shown that weak processor approximately three times more energy-efficient than powerful processor. Therefore, the switching software from ARM uses weak processor as long as possible [6].

VII. Summary

Concluding, the results and expectations of processor development in respect to energy consumption will be summarized.

A. Result

For a long time only power consumption of processors was an important topic in processor development [1]. Thereby, different approaches on hardware and software site have followed, these are described in IV and VI.

First of all, it is to say that the effort is worth of specialized processor. However, best results for energy-efficient is given by adapt Hardware and Software e.g. multi-core processors and parallelized tasks in applications.

Options to reduce power consumption on software side, could be the record of users behavior and switch-off of processors or parts of processors [1]. Since the switch-off is often inefficient, different operating modes for processors are existing, to take processors back into service quickly [1]. In addition, clock rate can be reduced by applications but other applications at the same time must be considered. Hence, clock rate can not be reduced automatically [7]. Another approach on software side is, to parallelize the implementation of applications [9].

On hardware side were described following techniques. The specialized processors can be combined to heterogeneous and homogeneous multi-core processors [9]. There exists an approach to assimilate specialized processors automatically to an application [7]. Another approach exists where the design of processors will be modified or specialized. The modification of processor design could be for example the combination of new material for silicon-bulk [1], [32], [45], so leakage or working voltage would sink. The specialization of processor design is e.g. multi-thresholding [1], [31].

B. Outlook

With parallelization of mulit-core processors or DSP, performance of smart-phones and tablet-PCs increases and power consumption is saved [40]. It should be noted, that applications must support parallelization of processors. If applications do not do this the modification of processors show no effect [9], [39]. In [40] it is support, that in 2015 more than 128 cores will be integrated on one multi-core processor.

The combination of DSP and standard processors or other specialized processors to a multi-core processor, like in [29], minimized the height of processors and created a better communication between DSP and other processor. Due to DSP in multi-core processor performance increased and power consumption fall. Currently (2011) the design of processors were new approaches realized, to reduced leakage. This is described in [5] with insert of 3D-Transistors. One idea, which is pursued by android, is deployment of user behavior, which could be specialized to individual cores, so that different cores are in different operating modes.

References

[1] K. Wüst, *Microprozessortechnik*. Wiesbaden: Vieweg+Teubner Verlag, 2011, ch. 12, pp. 237–248.

[2] P. Gelsinger, "Moore's Law - The Genius Lives On," in *The Technical Impact of Moore's Law*, vol. 20, no. 3. IEEE, 2006, pp. 18–20.

[3] A. Seeger. (2011) Smartphone-Entwicklung: Der Akku ist das Limit. online. areamobile. Letzter Zugriff am 21.01.2012. [Online]. Available: http://www.areamobile.de/specials/20467-smartphone-entwicklung-der-akku-ist-das-limit

[4] A. Carroll and G. Heiser, "An Analysis of Power Consumption in a Smartphone," in *USENIXATC'10 Proceedings of the 2010 USENIX conference on USENIX annual technical conference*. USA: USENIX Association Berkeley, 2010, pp. 21–21.

[5] T. Kaminski. (2011, Mai) Intel erfindet den Transistor mit einer 3D-Struktur neu. online. intel. Letzter Zugriff am 08.12.2011. [Online]. Available: http://newsroom.intel.com/community/de_de/blog/2011/05/04/intel-erfindet-den-transistor-mit-einer-3d-struktur-neu

[6] P. Greenhalgh, "Big.LITTLE Processing with ARM Cortex™-A15 & Cortex-A7," ARM, White Paper, September 2011.

[7] K. Atasu, L. Pozzi, and P. Ienne, "Automatic Application-Specific Instruction-Set Extensions under Microarchitectural Constraints," *International Journal of Parallel Programming*, vol. 31, no. 6, pp. 411–428, December 2003.

[8] A. Bode. (2006, Oktober) Multicore-Architekturen. online. Gesellschaft für Informatik e.V. München. Letzter Zugriff am 01.01.2012. [Online]. Available: http://www.gi-ev.org/no_cache/service/informatiklexikon/informatiklexikon-detailansicht/meldung/multicore-architekturen-145.html

[9] U. Brinkschulte and T. Ungerer, *Mikrocontroller und Mikroprozessoren*. Berlin: Springer, 2010, ch. 9, pp. 395–415.

[10] K. Wüst, *Mikroprozessortechnik*. Springer, 2011, ch. 7, pp. 87–110.

[11] T.-C. Chen, "Where CMOS is Going: Trendy Hype vs. Real Technology," in *The Technical Impact of Moore's Law*, vol. 20, no. 3. IEEE, 2006, pp. 5–9.

[12] R. Fischer, "Methodik für die Verlustleistungsabschätzung von Prozessoren mit Pipeline-Strukturen," Ph.D. dissertation, Universität München, Mai 2010.

[13] J. Kreuzer, R. Diemer, and T. Huber, "Energy requirements of mobile phones and sensor technologies in mobile health applications," in *4TH EUROPEAN CONFERENCE OF THE INTERNATIONAL FEDERATION FOR MEDICAL AND BIOLOGICAL ENGINEERING IFMBE Proceedings*, vol. 22, 2009, pp. 1042–1045.

[14] L. Benini, G. De Micheli, A. Macii, E. Macii, and M. Poncino, "Reducing Power Consumption of Dedicated Processors Through Instruction Set Encoding," in *8th GLS*, 1998.

[15] CPU - Central Processing Unit. online. Webopedia. Letzter Zugriff am 21.01.2012. [Online]. Available: http://www.webopedia.com/TERM/C/CPU.html

[16] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, 2001, ch. 28, pp. 503–534. [Online]. Available: http://www.dspguide.com/ch28/1.htm

[17] L. J. Karam, I. AlKamal, A. Gatherer, G. A. Frantz, D. V. Anderson, and B. L. Evans, "Trends in Multicore DSP Platforms," in *Signal Processing*, vol. 26, no. 6. IEEE, November 2009, pp. 38–49.

[18] P. Carlston, "Optimizing Digital Signal and Image Processing on Intel Architecture Processors," intel, White Paper, January 2009.

[19] (2011, November) Tesla-GPUs von NVIDIA treiben weltweit energieeffizientesten Petaflop-Supercomputer an. online. nVIDIA. Santa Clara, Kalifornien. Letzter Zugriff am 05.01.2012. [Online]. Available: http://www.nvidia.de/object/tesla-gpus-power-greenest-supercomputer-20111123-de.html

[20] "NVIDIA nForce MCP Audio Processing Unit," nVIDIA, Technical Brief, 2001.

[21] "Multimedia Applications Processors," AMD, Tech. Rep., September 2010.

[22] Q. Dinh, D. Chen, and M. D. F. Wong, "Efficient ASIP Design for Configurable Processors with Fine-Grained Resource Sharing," in *FPGA '08*. Monterey, California, USA: ACM, February 2008, pp. 99–106.

[23] W. Geurts, G. Groosen, D. Lanneer, and J. V. Praet, "Design of Application-Specific Instruction-Set Processors for Multi-Media, using a Retargetable Compilation Flow," in *Proceedings GSPx 2005* , 2005, pp. 1–6.

[24] C. Alippi, W. Fornaciari, L. Pozzi, and M. Sami, "A DAG Based Design Approach for Reconfigurable VLIW Processors," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, March 1999, pp. 778–779.

[25] M. Baleani, F. Gennari, Y. Jiang, Y. Patel, R. K. Brayton, and A. Sangiovanni-Vincentelli, "HW/SW Partitioning and Code Generation of Embedded Control Applications on a Reconfigurable Architecture Platform," in *Proceedings of the 10th International Workshop on Hardware/Software Codesign*, Estes Park, Colorado, May 2002, pp. 151–156.

[26] B. Leitenberger. CISC und RISC - Die Gegensätze der Rechnerarchitekturen. online. Letzter Zugriff am 06.01.2012. [Online]. Available: http://www.bernd-leitenberger.de/cisc-risc.shtml

[27] Processors. Support. intel. Letzter Zugriff am 06.01.2012. [Online]. Available: http://www.intel.com/support/processors/sb/CS-031505.htm#24

[28] B. Molen. (2011, September) NVIDIA releases Kal-El white papers, announces a fifth 'Companion' core for less demanding tasks. online. engadget. Letzter Zugriff am 05.01.2012. [Online]. Available: http://www.engadget.com/2011/09/20/nvidia-releases-kal-el-white-papers-announces-a-fifth-companio/

[29] J. Stern. (2011) online. [Online]. Available: http://www.engadget.com/2011/01/03/intels-2nd-generation-core-processor-family-announced-includes/

[30] "Snapdragon S4 Processors: System on Chip Solutions for a New Mobile Age," ARM, White Paper, October 2011.

[31] J. Kao, S. Narendra, and A. Chandrakasan, "Subthreshold Leakage Modeling and Reduction Techniques," in *ICCAD '02 Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*. New York: IEEE, 2002, pp. 141–148.

[32] P. G. Neudeck, G. M. Beheim, and C. S. Salupo, "600 C Logic Gates Using Silicon Carbide JFET's," in *Government Microcircuit Applications Conference Technical Digest*, Anaheim, CA, March 2000, pp. 421–424.

[33] S. Augsburger and B. Nikolic, "Combining Dual-Supply, Dual-Threshold and Transistor Sizing for Power Reduction," in *Proceedings of the 2002 IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'02)*. IEEE, 2002.

[34] S. Perrin. (2011, January) A material to revolutionize electronics. online. News Mediacom. Letzter Zugriff am 22.01.2012. [Online]. Available: http://actu.epfl.ch/news/a-material-to-revolutionize-electronics/

[35] V. Rißka. (2011, May) Intel kündigt revolutionären 3D-Transistor an. online. Computer Base. Letzter Zugriff am 27.01.2012. [Online]. Available: http://www.computerbase.de/news/2011-05/intel-kuendigt-revolutionaeren-3d-transistor-an/

[36] S. Roloff. (2009, September) Multicore-Architekturen. online. Ferienakademie 2009. Letzter Zugriff am 01.01.2012. [Online]. Available: www2.in.tum.de/hp/file?fid=311

[37] "TESLA M2090 DUAL-SLOT Computing Processor Module," nVIDIA, Tech. Rep. $BD-05766-001v02$, June 2011, Board Specification.

[38] (2011) Die nächste Generation der CUDA Architektur – Codename: FERMI Ein echter Supercomputer in einem Grafikprozessor. online. nVIDIA. Letzter Zugriff am 30.12.2011. [Online]. Available: http://www.nvidia.de/object/fermi_architecture_de.html

[39] A. Jerraya, H. Tenhunen, and W. Wolf, "Multiprocessor System-on-Chips," in *Computer*, vol. 38. IEEE, 2005, pp. 36–40.

[40] A. Bode, "Multicore-Architekturen," in *Informatik Spektrum*. Springer, May 2006.

[41] "Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor," intel, Tech. Rep. 301170-001, 2004.

[42] (2008) Energy-efficient AMD Desktop Processors. online. AMD. Letzter Zugriff am 31.12.2011. [Online]. Available: http://www.amd.com/us/products/technologies/cool-n-quiet/Pages/cool-n-quiet.aspx

[43] AMD PowerNow!™ Technology. online. AMD. Letzter Zugriff am 30.12.2011. [Online]. Available: http://www.amd.com/de/products/technologies/amd-powernow-technology/Pages/amd-powernow-technology.aspx

[44] M. Larabel. (2006, February) Intel EIST SpeedStep. online. phoronix. Letzter Zugriff am 31.12.2011. [Online]. Available: http://www.phoronix.com/scan.php?page=article&item=397&num=2

[45] T. Masubara and M. Hirose, "Overview of CMOS Technology Development in the MIRAI Project," in *The Technical Impact of Moore's Law*, vol. 20, no. 3. IEEE, September 2006.

Seminar Energieeffiziente Applikationen Energy Aware Computing

Cosmin Pitu
Carl von Ossietzky Universität
Department für Informatik
Oldenburg, Deutschland
cosmin.pitu@informatik.uni-oldenburg.de

***Abstract*—**
***Index Terms*—**

I. Einführung

Der Klimawandel ist eine der zentralen ökologischen, wirtschaftlichen und sozialen Herausforderungen der Zukunft. Dies wird auch in [1] diskutiert; in diesem Zusammenhang beweist die benannte Studie die Wichtigkeit der Informations- und Kommunikationstechnik (IKT). In der Tat gibt es aufgrund der Schnelllebigkeit der Technik- und Marktentwicklung kaum noch klare Grenzen zwischen den Begriffen: So nimmt einerseits das Gerätespektrum zu und vermischt sich andererseits aufgrund einer wachsenden Konvergenz von Funktionalitäten und Diensten, wie es auch in der genannten Studie [2] bestätigt wurde.

Trotz der Synergie von Funktionalitäten und die dadurch entstehende 'All-in-one' Geräte, verbleibt das verbindende Merkmal des Gerätespektrums immer noch dasselbe: der Bedarf an Energie der einzelnen Geräte. Deswegen sollte das prägnanteste Ergebnis der Studie [2] nicht mehr erstaunen: *"IKT-Strombedarf liegt in Deutschland bei über zehn Prozent"*. Weiterhin werden Informations- und Kommunikations-Technologien als globale Infrastrukturen selbst zu Großverbrauchern elektrischer Energie, wie im [1] bekanntgegeben. Schon im Jahr 2001 lag die Energiemenge, die durch die (IKT) in Deutschland verbraucht wurde, mit rund 38 TWh[1] bei 7,1 Prozent des gesamten Elektroenergieverbrauchs. Im Jahr 2007 betrug der IKT-bedingte Stromverbrauch mit 55,4 TWh bereits 10,5 Prozent des gesamten Stromverbrauchs in Deutschland. Dieser Stromverbrauch wird ohne Gegenmaßnahmen bis zum Jahr 2020 voraussichtlich um mehr als 20 Prozent auf rund 66,7 TWh deutlich steigen. Der resultierende CO2-Ausstoße, bei gleichbleibendem Energiemix, übersteigt den der gesamten deutschen Luftfahrt. Auf diesen Umstand wird unter Anderem in der Studie [3] eingegangen.

Global wird angestrebt, das Energiesparpotenzial der IKT-Industrie auszuschöpfen und dies durch globale Initiativen wie GeSI[2] oder *The Climate Group*[3] zu unterstützen. Diese Initiativen zeigen Richtlinien und Fallstudien auf, ebenso verschiedenen Kooperationen mit Unternehmen, um die Durchführung von energiesparenden Maßnahmen zu ermöglichen. Bundesweit wird dies durch die Umsetzung des "*Aktionsplan Green IT*" geleistet, der auf dem IT-Gipfel 2008 in Darmstadt von Wirtschaft, Wissenschaft und Bundesregierung verabschiedet wurde. Im Zuge dessen wird in Deutschland z.B. das Technologieprogramm des Bundesministeriums für Wirtschaft und Technologie namens *"IT2Green: Energieeffiziente IKT für Mittelstand, Verwaltung und Wohnen"* gegründet. Damit wird deutlich, dass auch Deutschland über ein beachtliches Klimaschutz-potenzial verfügt. Weiterhin entwickeln die größten Konzernen aus der IKT-Welt seit Jahren neue Programme zur Energieeffizienz: darunter zählt zum Beispiel die allgemeine 'Green' Initiative von Google[4] oder das 'Big Green' Projekt von IBM, in dem seit 2007 jährliche Investitionen in Höhe von 1 Milliard Dollar gemacht werden[5].

Folglich arbeiten diese Gremien auf einer globalen Skala: zum Beispiel legt die globale Studie 2008 namens *"SMART 2020: Enabling the low carbon economy in the information age"* (und dessen deutschen Addendum, [1]) den Fokus auf den größten Subsektoren der IKT: Rechenzentren und dem PC-Bereich. Darin werden auch einige möglichen, grundlegenden Mechanismen vorgestellt, mit denen eine erhöhte Energieeffizienz von Applikationen erreicht werden kann.

II. Motivation

Der Ziel der "Energy-Aware Computing" Domäne ist das Bewusstsein über den Energieverbrauch sowohl im Hardware- als auch im Softwarebereich zu fördern. Eine ähnliche Sichtweise wird in [4] vorgestellt: der Zweck wäre nicht nur die schnellstmöglichste Ausführung bestimmter Algorithmen/Berechnungen zu erreichen, sondern auch die damit gekoppelten Energiebedürfnisse minimal zu halten. Die zugrunde liegende Überlegung wäre die folgende: Energie-

[1]Terawattstunde (TWh): 10^{12} Wattstunden, diese Abkürzung wird zur Angabe großer Energiemengen verwendet

[2]Global e-Sustainability Initiative, eine internationale Partnerschaft von IKT-Unternehmen und Industrieverbänden, www.gesi.org

[3]Eine Non-Profit-Organisation mit dem Ziel, Lösungen gegen den Klimawandel und hin zu einer "Low Carbon"-Wirtschaft voranzutreiben

[4]Quelle: http://www.google.com/green

[5]Quelle: http://www-03.ibm.com/press/us/en/pressrelease/21524.wss

verbrauch oder Energie allgemein muss genau wie andere beschränkten Ressourcen eines Rechners (wie z.B. Speicherplatz oder Prozessorleistung) angesehen werden. Aus Nutzerstudien kommt die grundlegende Erkennung: längere Akkulaufzeiten sind Benutzer wichtiger als ein verbreiterter Funktionsumfang, dieser Wunsch ist sogar mehr von Bedeutung als die Entwicklung von besseren, "immersiveren" Anwendungen [5].

Weiterhin müssen die Auswirkungen dieses immer erhöhenden Energieverbrauchs (in der Welt der Supercomputers auch als "Moore's Law for Power Consumption" bekannt [6]) zu einer Zielgruppe von Anwendern zugewiesen werden. Während die direkten Einflüsse des Energieverbrauchs auf der Performanz eines Rechners vielleicht für einen alltäglichen Benutzer nicht unbedingt auffällig sind, dedizierten Anwendern ist dieses Thema von Wert. IKT-Unternehmen sind in erster Linie als Zielgruppe betroffen: die Verringerung von Emissionshöhen und Kostensenkung passen gut zusammen – eine verbesserte Energieeffizienz der einzelnen IT-Systeme spiegelt sich automatisch in senkenden Kosten [7].

Der aktuelle Stand der Forschung in dieser Domäne spannt den ganzen Spektrum des Computings: von der Senkung des Energieverbrauchs eines einzelnen Prozessorchips bis hin zur Kühlungproblematik eines großen Rechenzentrums. Wie erwähnt, der Domäne ist hauptsächlich für IKT-Unternehmen von Relevanz, insbesondere wenn es aus mehrerer Studien deutlich geworden ist, dass die mit der Beschaffung von Computing Hardware verbunden Kosten wesentlich von den Betriebskosten überschritten werden [7]; Betriebskosten beinhalten im Grunde genommen immer erhöhende Stromkosten als Einflussfaktoren. Konkrete Vorhersagen kommen aus der Industrie – Microsoft stellt die Perspektive dar, dass die Betriebskosten für Servers von der Beschaffungskosten bis 2015 übertreffen werden. Außerdem sind zum Thema "Betrieb" auch die dadurch entstehenden CO2-Ausstöße einzuschätzen.

Die nächste Frage wäre dann, welche Probleme werden in diesem Domäne adressiert. Die grundlegende Betrachtungswinkel muss systemisch sein: man versucht die von einer bestimmten auszuführenden Rechenaufgabe erforderliche Energie zu minimieren. Dies erfolgt indem die verfügbaren Ressourcen (z.B. Prozessor, Speicher, Netzwerk) in einer effizienten Art und Weise gesteuert werden, durch Maßnahmen wie z.B. das System möglichst nicht auslasten bzw. im Leerlauf behalten.

Diese Herangehensweise stellt das ursprüngliche Modell des Computerwesens gegenüber: Computersysteme sind ursprünglich mit den Gedanken entworfen worden, immer die beste Rechenleistung zur Verfügung stellen zu können, ohne bestimmter Rücksicht auf die Art der zu lösenden Aufgabe. Dieses Modell wird in einem energieeffizienten System immer noch berücksichtigt, da bestimmte Situationen vorkommen, in denen die Rechenleistung ausgenutzt muss; Im Zuge solcher Fälle muss wie erwähnt das System ebenso den Energieverbrauch auf ein Minimum senken.

Diese Perspektive wird auch in [4] erläutert, indem eine Herausforderung der Domäne hervorgerufen wird: es ergibt sich eine inverse Beziehung zwischen die für die Ausführung einer bestimmten Aufgabe benötigte Energie und die erforderliche Ausführungszeit – damit wird ein Kompromiss zwischen Energieverbrauch und Ausführungszeit verlangt. Folglich kommen oft Aufgaben vor, die mit einem geringen Energieaufwand gelöst werden kommen; in anderen Fällen, in denen die Geschwindigkeit der Lösung von Bedeutung ist, muss die Ausführungszeit mit den Zielen des Energieverbrauchs ausgeglichen werden.

Weiterhin hat die herausgestellte Problematik – wie wird der Energieverbrauch eines Rechnersystems bis zu einem Minimum limitiert – in den vergangenen Jahren eine stetig zunehmende Bedeutung gewonnen, wie in [8] erläutert: die betroffenen Teilgebieten sind die eingebetteten sowie die hochleistungsfähigen Systemen. Deswegen müssen unbedingt Überlegungen zu dem erwünschten Verhältnis des Energieverbrauchs in der Systemplanung enthalten sein. Während die höheren Abstraktionsebenen hinsichtlich der Energieeffizienz optimiert werden können (durch z.B. Compilers, Programmiersprachen oder auf der Ebene des Betriebssystems), stoßen alle diese Optimierungen an die Grenzen der Physik bzw. die thermodynamischen Feinheiten des Energieverbrauchs.

Das heißt, dass selbst ein systemweites Anstreben einer totalen Energieeffizienz nicht erreicht werden kann. In anderen Wörter ist es unmöglich, eine reine Verhältnismäßigkeit der Energie[6] zu erhalten, wie in [4] hingewiesen.

In [7] wird begründet, dass einer Weg zur Erreichung einer systemischen Ansicht des Energieverbrauchs der folgende wäre: jede Komponente des Systems sollte Informationen bezüglich der aktuellen Energiesituation mitteilen, sodass geeignete, energiesparende Maßnahmen ergreifen werden können. Die zentrale Stelle verwaltet die einzelnen Mitteilungen der Komponenten; enthalten in eine Mitteilung ist sowohl der Stand des Energieverbrauchs als auch eine Schnittstelle zu der verfügbaren Energie-bezogenen Steuermechanismen – diese Vision könnte mit einem Energy Abstraction Layer verbunden werden.

Um von der globalen Perspektive eine 'lokale', grundlegende Perspektive zu schaffen, müssen die wesentlichen Bestandteile eines Rechners betrachtet werden – die Welten der Hardware und anschließend die Software. Näher betrachtet werden die wesentlichen Konzepten zur Realisierung von energieeffizienten Hardware-Komponenten und zur Entwicklung energieeffizienter Software.

Die vereinfachte Ansicht der Software-Hardware Schnittstelle ist die einer Schichten-Hierarchie, in der die Hardware im Mittelpunkt steht. Darauf bauen die Schichten der Systemsoftware und anschließend die der Anwendungssoftware auf. Dies wird im Kursbuch von Patterson und Hennessy [9] beschrieben. Damit ist die Hauptsatz der Interaktion zwischen Software und Hardware

[6]Ein oder mehrere Rechnersystemen befinden sich in einem Zustand der Verhältnismäßigkeit der Energie (engl. "energy proportional") wenn sich der Energieverbrauch eines Rechnersystems im Verhältnis mit der durchgeführten Rechenarbeit befindet.

vorgestellt und zwar: Software weist die Hardware ein, wie die benötigten Funktionen auszuführen sind. Damit lassen sich Optimierungen der Energieeffizienz eines Rechensystems hauptsächlich auch durch Software-Techniken beeinflussen – dieses Thema wird im Laufe dieser Seminararbeit in den Mittelpunkt stehen.

Mit dieser Abgrenzung ergibt sich die Frage, warum sich ein Softwareentwickler mit Konzepten auseinander setzen muss, die anscheinend stark in der Hardwarewelt verankert sind. In [10] wird bekanntgegeben, dass die Mehrheit von Softwareentwickler keine Aufgaben in der täglichen Projektentwicklung hinsichtlich der Optimierung des Energieverbrauchs berücksichtigen. Dies fällt insbesondere in der Welt der mobilen Geräten auf, wo der Akzent häufig auf die Verbesserung der Akkulaufzeit gesetzt wird.

Außerdem wird in [11] die folgende Motivation gegeben: Trotz der transparenten Handhabung der meisten energiesparenden Mechanismen der Ziel-Plattform, können Anwendungen nichts Wirkliches im Raum des Energiesparens direkt bewirken. Was aber Anwendungen tun können, wäre ihr eigenes Benehmen während der Laufzeit zu kontrollieren.

In der Tat könnte durch einen rücksichtslosen Ablauf eines Rechenvorgangs zu der Situation kommen, indem die Anwendung die eigentlichen energiesparenden Maßnahmen des Plattforms hindert! Folglich spielt die Art und Weise, in dem ein Software implementiert worden ist, eine große Rolle in Themen wie Akkulaufzeit oder Energiekosten. Dadurch wird es deutlich, dass Techniken am Niveau der Software benötigt sind, die während ihrer Ausführung mögliche Energieersparnisse berücksichtigen. Ein Teil dieser Techniken werden im folgenden Kapitel vorgestellt.

III. Ausgangssituation

Nach der einführenden Aspekten wird die Diskussion über Energieverbrauch und Energieeffizienz in eine Ausgangsrichtung geführt. Die Untersuchung des Energieverbrauchs eines Rechners muss mit der Beschreibung des eigentlichen Ist-Zustands anfangen – nämlich welche Prozesse zu jedem Zeitpunkt in einem Rechner laufen. Die Benutzung eines Rechners beinhaltet unvermeidbar Energieverbrauch; damit befindet sich der Rechner in einem von zwei möglichen Zustände: entweder im Leerlauf (engl. idle state) oder bei der Ausführung einer Rechenaufgabe – in diesem Fall fließen Software Befehle sowie Daten durch die elektronischen Pfade; Dadurch wird für beide Zustände Energie verbraucht [12]. Software (oder auch Applikationen, Kurzform *Apps*) kann wie erwähnt der Energieverbrauch eines Systems deutlich beeinflussen, ein Thema das in der kommenden Kapitel näher eingegangen wird.

In [11] wird anhand der zwei beschriebenen Zustände zwischen *aktive* und *inaktive* (engl. "idle") Software unterschieden.

Unter aktive Software werden diejenige Applikationen untergliedert, die ihres Ziel (anders gesagt, ihrer Haupt-Anwendungsfall) erfüllen. Ziele könnten z.B. die Ausführung einer Tabellenkalkulation, das Abspielen von Musik oder Filme oder das Surfen ins Internet. In allen dieser Fälle wird eine Arbeitsbelastung (einzelnen Arbeitspakete oder eine Rechenaufgabe) verwaltet, präziser auf dem Prozessor (engl. CPU) oder ggf. (auch) auf der Videokarte (engl. GPU).

Auf der anderen Seite wird im Fall von inaktiven Software nur die Software betrachtet, die im Wesentlichen im Leerlauf ausgeführt wird, das heißt es werden keine konkrete Rechenaufgaben durchgeführt, es wird nur auf dem Signal zur Aktivierung gewartet. Beispiele aus dieser Kategorie wären: ein geöffnetes Textverarbeitungsprogramm, das aber im Hintergrund oder minimiert läuft, sowie ein laufendes Chatprogramm, das Mitteilungen weder verschickt noch empfängt.

Beide Typen von Software können durch Softwaretechniken hinsichtlich der effizienteren Energieverbrauch optimiert werden. Es werden als nächstes mögliche Herangehensweisen zur Optimierung des Energieverbrauchs vorgestellt, zuerst wird die systemische Perspektive erläutert.

IV. Energy-Aware Computing – Systemweit

In der wichtigen Arbeit [5] von Ranganathan wird die Entwicklung des Gebiets zusammengefasst: Strom und Energie sind zentrale Merkmale, die bei der Entwurf eines Rechensystems (von Spitzenrechner und Rechenzentren bis zum mobilen Geräten) immer zu berücksichtigen sind.

Während vorherige Arbeiten den Motto "Verschwendung von Energie vermeiden" deutlich hervorgehoben haben, bezieht sich die aktuelle Entwicklungsrichtung auf der Herausforderung, die Ursachen der Verschwendung zu identifizieren.Es wird damit versucht, neue Methoden zu entwickeln, die solchen Ursachen vermeiden könnten.

A. Ursachen der Energieverschwendung

Arbeiten wie [13] haben gezeigt, dass die meisten Aufgaben, die im Fall von mobilen Geräten durchgeführt werden (wie z.B. Musik hören, telefonieren, sowie surfen oder Emails verschicken) sehr unterschiedliche Ergebnisse hinsichtlich des Energieverbrauchs liefern. Damit stellt sich die Frage, warum manche Designs eines Systems energieineffizienter als andere für dieselbe Aufgabe sind.

Eine erste Berücksichtigung wäre die, dass in der Regel Systemen für den meist auftretenden, allgemeinen Anwendungsfall entworfen worden sind. Damit können kurzlebige Fällen eintreffen, denen aber schnell fast alle verfügbaren Ressourcen zur Verfügung gestellt werden. Ein anderer mit diesem Benehmen des Systems verknüpften Grund wäre die benötigte Fehlerredundanz bei der Ausführung: mehrere Ressourcen gleicher Art müssen bereitgestellt werden, um die Verfügbarkeit oder Korrektheit der Ausführung sicherzustellen.

Weiterhin werden nicht alle Phasen einer Rechenaufgabe in Benchmarks [7] berücksichtigt – normalerweise unterstreichen diese Methoden Höhepunkten, die mit der vollen Rechenleistung behandelt werden müssen. Dadurch geht die realistische Art und Weise, in der das System verwendet wird, verloren, da sich die Ergebnisse zu viel auf Einzelfälle konzentrieren. Vom Datenzentren-Bereich bis hin zur mobilen Systemen zeigt sich

[7]Die grundlegende, datenbasierte Basis für Systemdesign, in dem Merkmale wie z.B. die Geschwindigkeit der benötigten Berechnungen erfasst werden

die Belastung des Prozessors meistens gering (etwa 10-30%), also weit entfernt von einer völligen Auslastung.

Ein weiteres Faktor der Energieverschwindung wäre auch die Modularität des Systems: individuelle Komponenten werden nicht genügend miteinander integriert, was ihre Interaktionen angeht: die einzelnen Bauteile können von verschiedenen Teams oder sogar Hersteller gefertigt werden, teilweise ohne das gesamte Zusammenspiel der Bauteilen oder Komponente in voller Breite in einem Systemkontext sich vorzustellen.

Damit werden weitere Leistungsschwäche ins System eingeführt. Die Kommunikation zwischen diesen Schichten ist manchmal auch sub-optimal: auch wenn individuell energiesparende Maßnahmen ergriffen werden, kann diese Tatsache im globalen System weitere Ineffizienzen verursachen, falls die einzelnen Schichten den aktuellen Zustand einer anderen Schichte nicht rechtzeitig erfahren können.

Als Konsequenz werden die nicht genügend über den aktuellen Zustand informierten Schichten voraussichtlich von einem Grenzfall ausgehen, führend zu zusammengesetzten Energieineffizienz. Trotzdem können sich auch positive Situationen ergeben, wie z.B. die Kommunikation zwischen die Schichten eines drahtlosen Protokoll: die physikalische Schicht könnte energieeffizienter agieren, falls sie die Situation auf der Anwendungsschicht kennen würde.

Nachdem mögliche Ursachen für die Energieineffizienz vorgestellt worden sind, werden mögliche "Rezepte" *dagegen* eingeführt

B. "Rezepte" für Energieeffizienz

Energiegewinne können aber erzielt werden, in der genannten Arbeit [13] werden auch allgemeine Rezepte für Effizienz gegeben, die im Folgenden präsentiert werden.

1) Verhältnismäßigkeit der Energie erzielen: Eine erste Strategie wäre die Verkleinerung von zugewiesener Energie für nicht benutzte Ressourcen. Der Konzept wird auch als "Verhältnismäßigkeit der Energie" (engl. energy proportionality) bekannt und impliziert die Aufteilung von Ressourcen anhand der aktuellen Last. Um diese Optimierung automatisch auszuführen, werden spezifischen Algorithmen benötigt, die die Konsequenzen einer systemweiten Verringerung von unbenutzten Ressourcen mitberücksichtigen, indem z.B. auch die Kosten einer eventuellen, erneuten Hochfahrt eine Rolle spielen.

Die Techniken zur Erhaltung der Verhältnismäßigkeit stammen aus der Hardware-Welt und wurden z.B. in [14] zusammengefasst: im Fall von mobilen Rechner werden mehreren Voltzahl-Ebenen benutzt, oder, auf einer niedrigeren Ebene, ein energieeffizienterer Aufbau von Schaltungen sowie eine verbessertes Auftasten mittels Torschaltung (engl. clock gating) eingeführt. Eine weitere benutzte Technologie wäre die dynamische Skalierung der Frequenz der benutzten Voltzahl (engl. voltage-frequency scaling).

Das Skala einer Verringerung kann verschieden sein: es könnte eine bestimmte Komponente heruntergeschraubt werden, oder, falls eine gewisse Komponente die benötigten Maßnahmen nicht unterstützt, könnte die Verringerung systemweit stattfinden.

Beispiele einer systemweiten Optimierung im Fall von Rechnernetzen wäre die Veränderung der Leitweglenkung sodass unbenutzte Routers ausgeschaltet werden. Die Konsolidierung verschiedener Aufgaben verwendend virtuellen Maschinen repräsentiert ein anderes Beispiel solcher Optimierungen, die eine Verhältnismäßigkeit der Energie erzielt.

2) Konsolidierung von Aufgaben: Weiterhin wird von der Konsolidierung 'teuerer' Aufgaben (aus der Perspektive des Energieverbrauchs) diskutiert. Solche Aufgaben sind z.B. die bekannten Input/Output Aufgaben, die die Festplatte auslesen – mit der Konsolidierung von mehreren Auslesen-Operationen auf einem einzelnen Festplattedreh (engl. disk spin) könnte die gesamte benötigte Energie verringert werden. Eine ähnliche Strategie ist das "Vor-Herbeiholen" von Daten (engl. prefetching), die nur im Fall von *a priori* bekannten Datenströme oder die Anwendung eines mit mehreren Prozessen geteilten Zwischenspeichers anwendbar ist.

Dieser vorgestellte Leitfaden lässt sich für weitere höhere Funktionalitäten verwenden, indem die einzelnen Aufgaben eines Rechensystems konsolidiert oder zusätzlich aufgeladen werden – z.B. eine systemweit benutzte Operation oder Teilaufgabe könnte einmalig durchgeführt werden und dessen Ergebnis mittels dedizierten Techniken zu den einzelnen Bestandteilen bzw. zu den größerer Aufgaben kommuniziert werden.

3) Energieeffizienz in seltenen Fälle nicht betrachten: Diese Strategie beschreibt eine ziemlich unkonventionelle Betrachtungsweise, die gezielt und gewünscht die Energieeffizienz des Systems nur in den meist auftretenden Fälle bzw. beim häufigen Niveau der Arbeitsbelastung betrachtet. Das heißt, dass die globale Effizienz verbessert wird durch die explizite Verschlechterung der energiesparenden Maßnahmen bei Grenzfälle und durch die entsprechende Verbesserung dessen bei häufigeren Arbeitsabläufe.

Zum Beispiel könnte das Netzteil eines Servers bei normalen oder häufigsten Arbeitsbelastungen mit der höchsten Effizienz optimiert werden; damit werden gezielt die Grenzfälle nicht so energieeffizient behandelt werden.

4) Strom einer Dritte aufwenden: Im Raum dieses Konzeptes wird die lokale Perspektive der Energieeffizienz betrachtet: in einem Gerät (meistens wird ein mobiles Gerät als Beispiel genannt) kann z.B. die Akkulaufzeit dadurch verbessert werden, indem eine komplexere Berechnung von einem anderen Rechensystem einer Dritte gelöst wird. Dieses Paradigma entspricht dem größeren Thema von "Cloud Computing".

5) Strom verwenden, um Strom zu sparen: Ein weiteres Konzept zur besseren Energieeffizienz wäre folgender: es können Anwendungen gestartet werden, die Strom sparen können, obwohl es für die Ausführung solcher Hilfsanwendungen eventuell *mehr* Strom gebraucht wird.

Beispiele dieser Kategorie von Anwendungen wären globalen "Garbage collectors", welche periodisch das Profil (der Umfang, die Menge) des benötigten Arbeitsspeichers verringern, sodass einzelnen Speicherbanken in einem stroms-

parenden Modus (z.B. zu einer niedrigeren Frequenz) wechseln können. Außerdem kommen Algorithmen zur Datenkompression infrage, die geringere Energie für die Kommunikation, Austausch sowie Speichern von Daten brauchen.

C. Zusammenfassung

Zusammenfassend lässt sich die Energieeffizienz aktueller Rechensystemen zumindest mit einer Größenordnung verbessern. Ein erster Schritt bis dahin wäre ein Umdenken des unterliegenden Designs: aus der Sicht des Hardware-Designs ist die Unterstützung des Systems bei allen Schichten gefragt, sei es auf der Hardware-, Software- oder Anwendungsschicht. Die Kommunikation zwischen diesen einzelnen Schichten sollte essentielle Informationen enthalten, die die Verwaltung, Überwachung sowie Analyse des aktuellen systemweiten Energieverbrauchs von der Energiespareinrichtung ermöglichen. Dadurch können spezifischen Maßnahmen systemweit ergriffen werden.

Außerdem muss eine systematische Untersuchung der ”angeborenen” Ineffizienzen im Rechensystem stattfinden: es können Anwendungen im System laufen, die diese Maßnahmen sogar verhindern, mit der potentiell gravierenden Konsequenz einer suboptimalen Energieeffizienz.

Diese Erkenntnisse wurden auch im [15] hervorgerufen: die Rolle, die Software spielt beim Energieverbrauch ist vorrangig, da die von dem Prozessor durchgeführte Operationen vom Software gesteuert wird – dadurch werden alle einzelnen höheren Schichten in ihrer Energieverbrauch beeinflusst; weitere Beispiele von ineffiziente Software hinsichtlich des Energieverbrauchs sind in [16] zu finden.

Der Fokus wird im nächsten Kapitel auf der Softwareebene gelegt, wo Optimierungen auch erzielt werden können.

V. Best Practices für Energieffizienz im Software

Die Strebungen der Industrie zwecks Erhöhung der Energieeffizienz eines Rechensystems haben sich bis jetzt meistens auf der Hardware Ebene konzentriert [15]. Während es schon wichtig ist, die für grundlegende Berechnungen benötigten Energie auf der Hardware-Schicht zu verringern, verhältnismäßige wenige Recherche wurden für die Optimierung der Anzahl oder allgemeiner Qualität dieser Berechnungen investiert. Ein Zeichen dafür ist, dass Methodologien zum Lebenszyklus der Software-Entwicklung und das damit verbundene Prozessmanagement keine Parameter für Energieeffizienz enthalten. Weiterhin gehört Energieeffizienz nicht zu dem über 50 Parameter des ISO-Standards zu Softwarequalität (vgl. ISO 9126:2003). Das heißt, ein Umdenken der Qualität von Applikationen hinsichtlich ihrer Energieverbrauch muss infrage kommen.

Die historische Entwicklung der Energieeffizienz auf der Hardware Ebene wird in [17] auch dargestellt: seit Jahren haben Plattform-Anbieter versucht, die Akkulaufzeit mobiler Plattformen zu verbessern, indem Akku-Technologien ständig herausgebildet wurden. Andere Komponenten wurden auch weiterentwickelt, z.B. im Fall von Prozessoren wurden mehrere Zustände (P-States) mit geringerem Energieverbrauch hinzugefügt; Außerdem haben Komponenten mit gefüllt übermäßigen Energieverbrauch wie Displays ihre Energieeffizienz auch verbessert.

Dazu aber bleibt es immer noch Raum für Weiterentwicklung – auf dieser Stelle kommen die Software-Komponenten ins Spiel, aus dem Grund, dass ’gut benehmende’ Software erheblich zur Energieeinsparung beitragen kann.

Der Ausgangspunkt für möglichen Verbesserungen ist ein Rechensystem, dessen Plattform-Hardware mit aggressiven Funktionalitäten zur Energieeinsparung entworfen worden ist (z.B. ACPI, C- und P-States, PCIE ASPM, SATA LPM). Software-Komponenten, bestehend aus dem Betriebssystem, Firmware und Drivers der Hardware-Ebene spielen eine wichtige Rolle im Fall der Energieeffizienz der gesamten Plattform, wie in [18] motiviert.

A. Rechnerische Effizienz

Wie im Kapitel III erläutert wurde, wird in der Literatur [12] zwischen aktiven und inaktiven Software unterschieden; dazu wird ”hochgerechnet” bis zur Ebene der von einem Rechnensystems verbrauchten Energie. Dadurch untergliedert sich die Energie in der *aktiven* bzw. *inaktiven* (engl. idle) Kategorien. Ein eingeschaltetes Rechensystem befindet sich in einem *immer aktiven* Zustand, seine Energieeffizienz wird aber wesentlich durch die verwendete Art von Energie (aktiv oder inaktiv) beeinflusst. Die Schlussfolgerung wäre: je schneller die Aufgabe gelöst werden kann (mit Verbrauch von aktiver Energie), desto schneller kann das System in einen inaktiven, energieeinsparenden Zustand wieder wechseln.

Dadurch wird den wesentlichen Bestandteil der rechnerischen Effizienz angedeutet: die Zeit, in der sich das System in einem ’höheren’ Zustand (relativ zum Energieverbrauch) befindet, muss minimiert werden. Dieser Konzept wird als ”Rennen zum Leerlauf” (engl. race to idle) gekennzeichnet [12]. Davon abgesehen, dass fast alle Anwendungszenarien von Client-Systemen (also außer Server-Systemen in Datenzentren z.B.) den Leerlauf beinhalten, ist dieses wichtige Ziel [19] berechtigt (engl. ’respect system idle’).

1) Algorithmen: Eine erste Maßnahme zur Erhaltung einer erhöhten rechnerischen Effizienz ist die Auswahl von leistungsfähigen bzw. energieeffizienteren Algorithmen. Es muss einen Kompromiss zwischen Performanz und Energie getroffen werden, mittels einer Analyse der Ziele des Systems, dessen Architektur und die zu lösende Aufgabe. Datenstrukturen wie z.B. ein Stapel könnte energieeffizienter agieren als eine Warteschlange, oder ein B-Baum könnte für eine gewisse Aufgabe besser als ein binärer Baum sein.

Ein effizienterer Algorithmus kann bis zu zweimal so viel Energie ersparen (vgl. [12]); effizienter könnte in diesem Fall auch ”nicht so komplex” oder ”leistungsfähig” bedeuten – z.B. während das System auf Akku läuft, könnte ein nicht so performantes Video-Decoder benutzt werden [20].

2) Multithreading: Ein weiterer Konzept ist die Benutzung von mehreren Prozessen oder Threads (engl. multi-threading) zur Lösung einer Aufgabe. Damit wird eine erhöhte Leistung sowie eine verbesserte Energieeffizienz erhalten [11]: beispielsweise lösen acht Prozessen (Threads) eine Aufgabe viermal

schneller als ein einzelner Thread.

Wichtig in diesem Fall ist die Benutzung von sogennannten "ausgewogenen" Threads. Das heißt die acht genannten Threads sollen sich vorab so gleichmäßig wie möglich die einzelnen Teilaufgaben verteilen. Falls ein bestimmter Thread überwiegend mehr Zeit für eine Berechnung braucht im Vergleich zu den anderen Threads, dann wird dies als 'unausgewogenen' Threading bezeichnet, eine Situation die vermieden werden soll.

Unterschiede bei der gebrauchten Prozessor-Zeit (engl. CPU time) können auch zu unnötiger Energieverschwendung führen [17]: z.B. während ein Thread der Prozessor zu 100% belastet, können die anderen Threads nur einer Belastung von 10-20% entsprechen.

Außerdem wird auch in [20] empfohlen, die Planung oder Zuweisung (engl. scheduling) der einzelnen Threads zu dem einen oder anderen CPU-Kern vom Betriebssystem behandeln zu lassen, sodass verschiedene Optimierungen am Ebene des Ausführungsplattforms eingesetzt werden können.

3) Vectorization und Performanz-optimierte Bibliotheken: Diese Strategie versucht, die Anzahl von aktiven Zyklen auf dem Prozessor (engl. CPU cycles) zu verringern. Der Quelltext, der dieselbe Operation auf mehreren unabhängigen Datensätze ausführt, lässt sich gut *vektorisieren* (engl. Vectorization), sodass die Ausführung bei anderen Datensätze weniger Zeit in Anspruch nimmt.

Moderne Prozessoren von AMD und Intel besitzen fortgeschrittene Anweisungen wie z.B. SIMD (Single-Instruction, Multiple Data), die dieselbe Berechnungen (z.B. Gleitkomma-Operationen) auf mehreren Datensätze durchführen können.

Im Fall von aktuellsten Prozessor auf die Intel *Sandy Bridge* Architektur können pro Sekunde acht Gleitkomma-Operationen auf 32-bit Datensätze gleichzeitig durchgeführt werden [12], dank der Vektorisierung und Anwendung von SIMD Befehlen.

Allgemeiner können spezifische erweiterte Sätze von Anweisungen (engl. instruction set extensions) wie MMX oder SSE für rechenintensive Applikationen benutzt werden. Ein Beispiel für die Effizienz von Intels "Erweiterte Anweisungen" Technologie wie SSE2/4 wäre folgende: bei der Optimierung einer Anwendung zum Abspielen von Medien konnte eine 2.15x Verbesserung der Performanz und eine 30% Verbesserung des Energieverbrauchs beobachtet werden [12], dank dieser Erweiterungen.

Außerdem können speziell für Performanz optimierte Bibliotheken bei der Lösung von rechenintensiven Aufgaben benutzt werden. Diese Bibliotheken wie z.B. Intel Performance Primitives (IPP) oder die Intel Math Kernel Library sind plattform- sowie betriebssystemunabhängig und lassen sich dadurch in jede Anwendung integrieren. Diese Bibliotheken stehen hochoptimierte, parallelisierte Funktionen/Algorithmen für spezielle Aufgabentypen zur Verfügung (beispielweise Audio/Video Enkodierung und Dekodierung, Kryptographie oder Signalverarbeitung). Durch die Verwendung von hochoptimierten Codebibliotheken könnte das System schneller im Leerlauf gesteuert werden, weil die Berechnungen erheblich schneller ausgeführt werden können.

Das heißt, dass Software auch die letzten Innovationen der Ziel-Plattform benutzen sollte, um eine wesentlich erhöhte Energieeffizienz (z.B. bis zu 1.6x effizienter mit Hilfe der Intel AVX Technologie [11]) und Performanz zu erstreben.

B. Dateneffizienz

Dateneffizienz bezieht sich auf das Erkenntnis, dass eine effiziente Behandlung der für die Anwendung benötigten Daten oft die für die Lösung der Aufgabe verlangte Energie verringern kann. Einen guten Überblick von Dateneffizienz im Vergleich zu rechnerischer Effizienz wird zusätzlich auch in [21] gegeben.

Eine erste Überlegung, welche die Dateneffizienz einer Applikation verbessern kann, ist die Verringerung von "Datenbewegungen" – das heißt, es muss gestrebt werden, die Übertragung von Daten vom System zu einem typischen Speichergerät wie eine Festplatte oder eine optische Speicherplatte auf einem Minimum zu senken. Die Lösung dafür wäre die Verwendung einer Zwischenspeicherung während der Übertragung [20]. Zudem können auch die Daten *a priori* übertragen werden (engl. prefetching) [11].

Die "Datenbewegungen" hängen vom Ziel-Plattform ab – die meisten Hardware-Plattformen (bzw. Prozessoren wie Intel oder ARM) besitzen eine "Aufladen-Abspeicher" Architektur (engl. load-store) [10]: aus diesem Grund können gar keine Daten bearbeitet werden, ohne das eine Datenbewegung stattfindet (von einem Platz zu einem anderen und öfter anschließend wieder in die umgekehrte Richtung). Zum Beispiel müssen Werten aus dem Hauptspeicher erstmal ins Registern für die Abarbeitung bewegt werden; anschließend müssen die neue Werten (die Ergebnisse) zurück ins Hauptspeicher aufgeladen werden.

Hinsichtlich der Datenübertragung werden in [10] folgende Empfehlungen beschrieben: die relativen Kosten (bezogen auf dem Energieverbrauch) eines Zugriffs im Hauptspeicher senken je tiefer oder "näher" sich der Hauptspeicher (oder ein allgemeiner Zwischenspeicher) vom Kern eines Prozessors befindet. Diese Perspektive muss dann entsprechend bei dem Entwurf des Rechensystems berücksichtigt werden.

Weiterhin wird es aus der Sicht der Optimierung auch vorgeschlagen, die einzelnen Zugriffe auf dem Hauptspeicher zu einem Minimum zu senken – diese Optimierung ist wichtiger als die Versenkung der Anzahl einzelner Anweisungen (wie im V-A erläutert). Diese Konzepten können mithilfe fortgeschrittener Technologien des Ziel-Plattforms realisiert werden, wie z.B. "Scratchpad Memory" bei der ARM-Architektur, die z.B. eine 170x energieeffizienterer Speicher-Zugriff erlaubt, im Vergleich zu einem Hauptspeicherzugriff [10].

Andere Empfehlungen betrachten die oft eintretenden Lesezugriffe – in mehreren Einheiten (z.B. in Blöcke, die etwa 8KB groß sind) auszulesen benötigt wenigere Energie und bringt den Prozessor einen geringeren Last [17]. Dafür ist clientseitig die Verwendung von Defragmentiertungstechniken empholen. Auch bei der Speicherung von Dateien sollen

mehrere aufeinanderfolgende Dateien vorbelegt werden, z.B. mittels die ”SetLength()” Funktion in .NET Anwendungen.

Erweiterte Technologien können auch bei der Dateneffizienz erheblich beitragen: asynchrone Input-Output Operationen zusammen mit ”Native Command Queueing” (NCQ) bringen eine erhöhte Energieeffizienz [17]. Dafür sollen Anwendungen, die mit zufälligen Input-Output Operationen sowie Input-Output Operationen auf mehreren Dateien rechnen müssen, alle Lesevorgänge in einer Warteschlange verwalten und dann asynchron diese auslösen. Anhand von Ereignissen (z.B. Events oder Callbacks) auf den einzelnen Lesenoperationen können individuelle Bestandteile eines größeren Lesevorgangs den Fortschritt mitteilen.

C. Context Awareness

Context Awareness bezieht sich im weitesten Sinne auf die Fähigkeit von Software, ihre Umgebung[8] wahrzunehmen und darauf zu reagieren. Beispiele vom ”context-aware” Benehmen in Software wären: die Erkennung der Typ des aktuellen verwendeten Strom (Wechselstrom oder Gleichstrom) und die darauffolgende Aktion – beispielsweise die Abblendung des Displays. Andere Sensoren wie ein Beschleunigungsmesser können auch vorhanden sein: die Festplatte eines Laptops kann ausschaltet werden, wenn einen Fall des Geräts entdeckt wird, laut [17].

Im Raum der Energieeffizienz könnte eine über den Kontext bewusstseine Software wichtige Energieersparnisse erbringen; ein vereinfachtes Beispiel könnte eine Anwendung sein, die aufhört, Bilder zu berechnen und ausgeben (engl. render) falls das Bildschirm ausgeschaltet wird. Weitere theoretische sowie praktische Aspekten der Domäne ”Context-Aware Computing” werden auch in [22] erläutert.

Die Interaktion zwischen eine context-aware Anwendung und dem Benutzer wird von einer Änderung des Kontextes erfordert. Diese kann aktiv oder inaktiv erfolgen [11] – einerseits wäre eine passive Interaktion z.B. die Darstellung einer Mitteilung über das Ereignis, zusammen mit möglichen, vom Benutzer auszuwählenden Schritte für eine weitere Behandlung.

Eine aktive Interaktion würde andererseits automatisch die beste Entscheidung treffen, ohne eine explizite Antwort vom Benutzer zu benötigen. Weiterhin können die aktive Interaktionen direkt die Fähigkeiten der Hardware benutzen (z.B. ein Sensor für Nebenlicht) sowie einstellbar ausgelöst werden (beispielsweise keine Defragmentierung laufen lassen, falls batteriebetrieben).

D. Zusammenfassung

Zusammenfassend beeinflusst Software die Energie Effizienz jeder Computing-Plattform, von eingebetteten Systemen bis hin zum Supercomputers [23].

Während die Plattform mit einer Arbeitsbelastung beschäftigt ist, lassen sich auf der Software-Schicht bestimmte Techniken, die im Laufe dieses Kapitels vorgestellt wurden einsetzen, sodass die Aufgabe schneller erledigt werden kann und damit das System zurück in einen energiesparenden Zustand gebracht werden kann.

Verfolgend die erwähnten Richtlinien und benutzend dedizierten Tools (wie in dem entsprechenden Kapitel aus [20] aufgelistet), kann die Energieeffizienz von Software verbessert werden: weitere Empfehlungen betonen die Wichtigkeit von Energieeffizienz in Softwareprojekten, die als eine wichtige Metrik bei der Projektverwaltung angesehen werden soll [15].

Im Endeffekt können sogar kleine Änderungen, falls auf Millionen von Systemen vergrößert, einen dramatischen Unterschied machen [12].

VI. Ausblick und Vision

Die historische Entwicklung der Energiespareinrichtung ist stark in kleineren Rechensystemen und die Evolution von mobilen Geräten [16] verankert: diese Systemen beinhalteten meistens nur eine geringe Anzahl von Komponenten wie z.B. ein Prozessor mit einem Kern und eine Festplatte, deren Geschwindigkeit beeinflusst werden könnte.

Aus dieser Mangel von verfügbaren Ressourcen lassen sich die ehemaligen Rechensystemen in einer ”binären” Art verwenden: die Ressourcen waren einfach entweder ein oder aus. Damit könnte z.B. eine Verwaltung der verbrauchten Energie vereinfacht gestaltet werden – falls das System eine größere Zeitspanne von Inaktivität feststellte, dann wurden die Frequenz des Prozessors sowie die Geschwindigkeit der Festplatte reduziert. Dadurch könnte die Energieeffizienz des Systems verbessert werden, ohne die Integration mit anderen Subsystemen zu erfordern.

Im Vergleich damit ist die Topologie von modernen Rechensystemen heutzutage deutlich komplexer geworden. Andererseits wird aus der Software-Sicht in [16] eine verbreitete Einordnung von Software erwähnt: Verwalter (Produzenten) sowie Antragsteller (Verbraucher) von Ressourcen. Durch das Zusammenspiel von verschiedener Komponenten der System kommt es oft zu Situationen, in denen nur ein Teil des Systems eine Aufgabe erledigt, während die anderen Teilen sich im Leerlauf befinden. An diesen Stellen müssen die Produzenten verschiedene Energisparmechanismen ausführen, die die Energieeffizienz des Systems verbessern; die Produzenten arbeiten aber innerhalb der Beschränkungen und Aufforderungen der Verbrauchern.

Falls die Beschränkungen hoch sind und die Ressourcen überlastet oder nicht effizient von den Anwendungen benutzt werden, sind die Produzenten hilflos – die Effizienz des gesamten System wird damit beeinträchtigt.

Aus diesen Gründen muss die Entwicklung von ”gut benehemender” (z.B. hinsichtlich ihrer Verhältnismäßigkeit IV-B1) Software vorangetrieben werden. Überlegungen müssen vorab aus der Sicht des gesamten Systems stattfinden, wie in IV beschrieben. Außerdem können spezifischer Hinweise oder Leitfaden der energieeffizienter Softwareentwicklung verfolgt werden, wie in V beschrieben.

Einen guten Übersicht von einer Rechensystem-übergreifenden Lösung wurde in [24] gegeben: anhand

[8] Die ”Umgebung” von Software enthält meist Hardware-Komponenten, wie z.B. Sensoren

einer Energieprofil[9] der Ziel-Plattform arbeiten Software und Hardware zusammen – anhand der aktuellen Arbeitsbelastungen, die ihre Bedürfnisse an Performanz dem System mitteilen (entweder durch direkte Kommunikation oder durch Beobachtungen durch das System), optimiert das System dynamisch die Konfiguration der einzelnen Komponenten zu einem meist energiesparenden Niveau.

Weitere spannenden Ideen in der Domäne versuchen Energieeffizienz schon während der Programmierung einzubetten: in [25] wird eine mögliche Lösung für Anwendungen vorgestellt, die energieeffizienteren sein können, wenn ein Verlust an Präzision annehmbar ist. Der Grundprinzip hinter EnerJ[10] ist die explizite Anmerkung (Annotation) von Daten in Quelltext, die mithilfe einer Annäherung berechnet werden könnten. Zu diesen Daten werden angenäherte Variabeln zugeordnet; weiterhin erfolgt die Speicherung mit geringerer Energie und mittels Anweisungen mit geringerem Energieverbrauch. Die Isolierung von genauen und angenäherten Quelltexten ist vorhanden, der Entwickler muss nur noch eine Schnittstelle für die Übergang von angenäherten zu genauen Daten bereitstellen. Bei der Portierung von mehreren Anwendungen auf EnerJ-angetriebenen Quelltext wurden Energiegewinne von 10% bis 50% beobachtet, zusammen verknüpft mit geringerem Verlust an Präzision der durchgeführten Berechnungen.

Komponenten wie Festplatten können auch von einem technologischen Aufschwung profitieren: z.B. haben Solid State Disks (SSDs) eine gute Potential für Energieeinsparung im Kontext von Datenbanksystemen gezeigt [26].

Aus diesen Gründen stellt sich die Entwicklung von energieeffizienter Software als eine großer Bestandteil von Energy Aware Computing [16] – während die Techniken zur Senkung des Energieverbrauchs vom zu gering genutzten Ressourcen in jede Energiespareinrichtung auch im größeren Rechensystemen schon tiefgreifend sind, verbleiben die verschiedenen Software-Ebenen, die diese Ressourcen verwalten, noch nicht entfaltet.

Um Bewusstsein für diese erwünschte Evolution von Anwendungen hinsichtlich ihrer Energieverbrauch zu schaffen, müssen Softwareentwickler zuerst versuchen, auf die Energieeffizienz des Softwareprodukts zu achten, da nur 13.4% der Organisationen ihr Energieprofil bzw. ihr Energieverbrauch überprüfen [27]. Während sich die Studie nicht unbedingt auf Software bezieht, ist diese Erkenntnis immer noch bedenklich, sehend dass fast alle Domäne vom ubiquitären Computing geprägt sind.

Die im Raum dieser Arbeit vorgeschlagenen Herangehensweisen versuchen den aktuellen Stand der Recherche in der Domäne zu verfolgen: eine "macro", systemübergreifende Strebung sowie eine "micro" Untersuchung auf Niveau der einzelnen Anwendung sind diejenige in der Domäne untersuchten Techniken.

Diese Vision von Energy Aware Computing befindet sich immer noch am Anfang des Weges [24]: für eine wirksame Optimierung des Energieverbrauchs muss Software in der Lage sein, ihre Performanzbedürfnisse hinsichtlich der aktuellen Konfiguration der Hardware abzubilden. Außerdem müssen Methoden für die voraussagende Abschätzung einer Arbeitsbelastung tiefer untersucht und weiter entwickelt werden – dafür verantwortlich ist wieder die Software-Ebene.

[9]Ein solcher Profil würde die Aufsicht auf die Ressourcen führen, indem die Art und Weise sowie Stelle des Energieverbrauchs bekanntgegeben werden. Damit können energiesparende Maßnahmen ergriffen werden.

[10]Eine im Raum der zitierten Arbeit entwickelte Programmiersprache, die die Einbettung von angenäherten, energieeffizienteren Berechnungen in Quelltext einer Anwendung erlaubt

REFERENCES

[1] GeSI. (2008) SMART 2020 Addendum Deutschland: Die IKT-Industrie als treibende Kraft auf dem Weg zu nachhaltigem Klimaschutz GeSI 2008. GeSI, BMWi. [Online]. Available: http://www.gesi.org/Media/GeSINewsFullStory/tabid/85/smid/503/ArticleID/43/reftab/37/t/The%20Need%20For%20a%20SMART%20Alliance/Default.aspx

[2] F.-I. für Zuverlässigkeit und Mikrointegration and F.-I. für System-und Innovationsforschung. (2009) Abschätzung des Energiebedarfs der weiteren Entwicklung der Informationsgesellschaft. Fraunhofer-Institut, BMWi. [Online]. Available: http://www.e-energie.info/documents/abschaetzung_des_energiebedarfs_der_weiteren_entwicklung_der_informationsgesellschaft.pdf

[3] W. Nebel, M. Hoyer, K. Schröder, and D. Schlitt. (2009) Untersuchung des Potentials von rechenzentrenübergreifendem Lastmanagement zu Reduzierung des Energieverbrauchs in der IKT. [Online]. Available: http://www.offis.de/fileadmin/Chefredakteur_files/PDFs/Pressemitteilungen/2009-11-19_OFFIS-Studie_zum_Lastmanagement_in_Rechenzentren__Veroeffentlichung_.pdf

[4] A. L. Couch and K. Kumar, "Summary of workshop on power aware computing and systems (hotpower08)," *Login*, 2008.

[5] P. Ranganathan, "Recipe for efficiency: Principles of power-aware computing," *Communications of the ACM*, 2010.

[6] W.-C. Feng, "Making a case for efficient supercomputing," *ACM Queue*, 2003.

[7] J. W. Smith. (2010) Green cloud - a literature review of energy-aware computing. [Online]. Available: http://www.cs.st-andrews.ac.uk/~jamie/files/LiteratureReview.pdf

[8] K. V. Palem, "Energy aware computing through probabilistic switching - a study of limits," *IEEE Transactions On Computers*, 2005.

[9] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design, Fourth Edition: The Hardware/Software Interface*. Morgan Kaufmann, 2008.

[10] C. Shore, "Developing power-efficient software systems on arm platforms," *IQ Magazine*.

[11] B. Steigerwald and A. Agrawal, "Developing green software," *Intel Press*, 2010.

[12] B. Steigerwald, C. D. Lucero, and A. Agrawal, "Writing energy-efficient software," *Intel Press*, 2011.

[13] R. N. Mayo and P. Ranganathan. (2003) Energy consumption in mobile devices: Why future systems need requirements-aware energy scale-down. [Online]. Available: http://www.hpl.hp.com/techreports/2003/HPL-2003-167.pdf

[14] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *IEEE Magazine*.

[15] E. Capra, G. Formenti, C. Francalanci, and S. Gallazzi. (2010) The impact of mis software on it energy consumption. [Online]. Available: http://is2.lse.ac.uk/asp/aspecis/20100023.pdf

[16] E. Saxe, "Power-efficient software," *Communications of the ACM*.

[17] B. Steigerwald, R. Chabukswar, K. Krishnan, and J. D. Vega, "Creating energy-efficient software," *Intel Press*, 2007.

[18] M. Sabharwal and E. Chang. (2010) Impact of 'idle' software on battery life. [Online]. Available: http://www.intel.com/content/dam/doc/guide/idle-software-battery-life-idf2010-presentation.pdf

[19] M. Robben. (2010) Designing energy efficient applications. [Online]. Available: http://download.microsoft.com/download/F/4/9/F49C6DE3-3BD1-405B-BA03-539EB58B57E8/SOW-T109.pptx

[20] P. Larsson, "Energy-efficient software guidelines," *Intel Press*, 2008.

[21] G. Arnout, "Data-efficient software and memory architectures are essential for higher performance and lower power," *Information Quarterly*, 2005.

[22] A. Singhal, U. Chaudhary, and C. Prasad. (2011) Energy conservation theory in context aware computing. [Online]. Available: http://ssrn.com/abstract=1898438
[23] B. Steigerwald, C. D. Lucero, C. Akella, and A. Agrawal, "Impact of software on energy consumption," *Intel Press*, 2011.
[24] D. J. Brown and C. Reams, "Toward energy-efficient computing," *ACM Queue*, 2010.
[25] A. Sampson, W. Dietl, E. Fortuna, and D. Gnanapragasam. (2011) Enerj - approximate data types for safe and general low-power computation. [Online]. Available: http://www.cs.washington.edu/homes/djg/papers/Enerj-pldi2011.pdf
[26] D. Schall, V. Hudler, and T. Härder. (2011) Enhancing energy efficiency of database applications using ssds. [Online]. Available: http://wwwlgis.informatik.uni-kl.de/cms/fileadmin/SHH10.C3S2E.pdf
[27] N. C. Centre, *The Green IT Paradox: Results of the NCC Rapid Survey*. National Computing Centre Ltd, 2008. [Online]. Available: http://www.ncc.co.uk/article/?articleid=13267

Towards an Energy Abstraction Layer

Mirco Josefiok
Carl von Ossietzky Universität
Department für Informatik
Oldenburg, Deutschland
mirco.josefiok@se.uni-oldenburg.de

***Abstract*—Applications in mobile computing have become more and more important nowadays. Increasing complex tasks are handled on the go. But there is a lack of power saving techniques regarding applications and user behavior. This paper will lay out the basics for a layer between the operating system and applications to improve energy efficiency in mobile computing.**

***Index Terms*—component; application; energy saving; energy aware computing;**

I. Introduction

Energy-efficiency and Green-IT are topics which are given great attention. With the recent UN-Climate Change Conference in Durban saving energy has become more important then ever.

Energy consumption in mobile devices is highly interesting because of the fact that battery development cannot keep pace with the increasing demand for longer usages of mobile devices and the also increasing performance of mobile devices [1].

At Christmas 2011 about 6.9 million Android and iOS devices have been activated[1]. Information and communication technology have an amount of about 8 % of the energy consumption in Germany as mentioned in [2]. There is a huge potential for saving energy with the increasing number of mobile devices .

It is known that the possibility to die shrink semiconductors is limited and that there will be barriers that cannot be overcome. Also in [3] is mentioned that there is already a lot of research on low level computing. Therefore in other areas of information and communication technology great efforts are made to design every used component as efficient and wisely as possible [4], [5].

For mobile devices it is not possible to build a closed system in which hard- and software are a perfect match because of the great variety of available applications for mobile devices, for example Apple's App Store holds over 500.000 different applications. Methods and techniques to reduce energy consumption covered by regular personal computers cannot be adapted without changing them, because they are always connected to a power source and the main goal is to reduce heat output so the cooling effort does not need to be that high [6].

Concerning the operating system there are less research strategies than in the fields of hardware design and low level energy optimization and most do not consider the user and his activities. In this paper a first outline for an energy abstraction between the operating system and the applications will be presented. The intended purposes of this layer is to monitor user activities and to build application profiles for optimized energy consumption. To do so dynamic analysis of given applications is needed.

The idea to recognize entities by their energy profile comes from researches regarding smart grid technology. It is assumed that the same can be done for applications. The energy abstraction layer therefore should be aware of the running, applications, used components and the connection from applications over components to energy consumption. This all with a user profile in mind.

An energy abstraction layer as described above should contain the following parts:

- Measurement: A specification for measuring energy consumption of a device and monitor the user behavior. A valid user profile can be created with these information to, for example, determine when which application is used in which manner and how much energy is consumed.
- Management: Part of the specification is a management part as well. In this part necessary interfaces and functionality are defined to influence the devices behavior.
- Energy API: An API provides advanced functionality on top of the specification for measurement and management.

A complementary task is to provide the possibility to run re engineering services on a given application or even handle the scheduling for certain applications.

In this paper the basic concepts and technologies for creating an *Energy Abstraction Layer (EAL)* in the mobile computing domain are described.

II. Structure of the Paper

The paper consists of eight chapters. The first chapter, no III, after the introduction is **Background and Related Work** where an overview of the related work and the search domain in which this work is located is given. The next chapter, no IV, is **Energy Management in mobile Computing** which contains a summary of the energy management technologies in modern mobile operating systems. The following chapter

[1] http://goo.gl/J9FvM visited February 2012

Necessary Technologies for an EAL, no V, deals with the technologies needed to create an EAL. An overview of a possible architecture for an EAL in chapter VI **Architecture of the Energy Abstraction Layer** is given. The next chapter VII **Possible Use Cases for an EAL** focuses on possible usage scenarios for the EAL and gives an outlook for practical application possibilities. A conclusion complements this paper. It also contains an outlook outlook for further research in this area. Possible next steps are briefly described.

III. Background and Related Work

As mentioned before creating an EAL touches various domains of information science and that is why the related work is somehow scattered. For creating an API there are several basic works. But an overview is given in [7] which describes the way of creating an API suited for developers needs.

In [8] and [9] dynamic program analysis are described and an overview is given on how software engineering can benefit from using these techniques. In [8] it is especially outlined with which methods one can identify certain features within an application by using dynamic analysis and aspect orientated programming .

In addition in [10] reducing energy consumption with static user states is shown.

The COMPLEX project - COdesign and power Management in PLatform-based design space EXploration [5] at the OFFIS e.V. is a project to create a design framework for embedded hard- and software. During this project a framework for static and dynamic analysis of Android applications was developed.

A survey about source code analysis is given in [11]. It is described how to transform a source code in an abstract syntax and how to perform a graph analysis for better understanding of the application behavior.

There are quiet a few works which describe how to influence energy consumption in mobile computing. For example [12]–[15] and many more focus mostly on one group of components or one part of the system in their optimization efforts.

At last how to analyse user behavior is outlined in [16], [17].

There is much more related work but the above mentioned literature was the one with the most impact on the work in hand.

IV. Energy Management in mobile Computing

Energy management in mobile computing has always been a topic but came into focus with the recent advent of portable devices like iOS and Android devices.

As described in [15] great efforts are made to reduce energy consumption for mobile devices. While it is possible to produce fairly efficient hardware with more and more optimization (e.g. special processors) or scaling semiconductor devices to a lower level, on the application layer not much work has been done yet. On the operating system level there are the older Advanced Power Management System (APM) and the newer Advanced Configuration and Power Interface (ACPI). While APM is BIOS based ACPI is integrated into the operating system [18].

Besides hardware optimization (e.g. special processors and tighter integrated components) more modern operating systems contribute a lot to increase battery life on smartphones [19]. But there are several other important areas mentioned in [15]. This are *Smart Batteries* where batteries learn to adapt to certain situations (e.g. low level of energy left). *Energy-Efficient GUI Design* describes strategies and technologies on how create GUIs more energy efficient. *Sleep to Save Energy* focusses on the processor and how tasks can be fulfilled as fast as possible to keep the processor in an idle state as long as possible. *Power Efficient Communication* summarizes techniques of the Wireless LAN standard. *Proxy Assisted Energy Saving* summarizes concepts of a server in between the users device and the internet and how that could help to make devices more energy efficient. *Source-level Power Control* describes on how to manage individual components and control them better. *TCP Based Energy Saving* brings its own energy saving techniques which are described as well. *Upper-level Power Management* focuses on the application level and how to develop applications in a way that they consume less energy. *Virtual Memory on the Network* is popular again due the increasing demand for cloud based applications. Therefore the impact of the energy consumption is discussed under this topic. *Programming and Compilation Techniques* focusses on modern programming languages and compiler techniques to save energy. *Integrated Power Management* describe energy saving techniques on different computation levels. *Energy Estimation Models* gives an overview of energy awareness in certain systems.

Todays operating systems are derived from their desktop pendants and thus share a lot of the known features of their desktop pendants. For Example Android, in this case, is merely a mobile version of Linux with a more aggressive power management pre-set [20]. iOS as well is based on MacOS X but neither Android (in the ARM Version) nor iOS bring an own implementation of ACPI with them [19], [21]. But at least in Android an energy management kernel extension, which is called *PowerManager*, is obviously designed with ACPI in mind [22].

Because of the open source nature of Android and the good and growing documentation for the operation system specifics all further examples for mobile devices are made with Android in mind. Other modern operating systems like iOS bring concepts that are fairly common to the better documented Android features [23].

A. Advanced Configuration and Power Interface (ACPI)

ACPI is the successor of the Advanced Power Management (APM) system. In APM powered management is done on BIOS level. ACPI provides an interface for the operating system to control all components registered at the interface. ACPI provides different states for the system components and the processor.

Value	CPU	Screen	Keyboard
PARTIAL	ON	Off	Off
SCREEN DIM	ON	Dim	Off
SCREEN BRIGHT	ON	Bright	Off
FULL WAKE	ON	Bright	Bright

TABLE I
WAKELOCK OVERVIEW

System states are working, sleeping, soft off or mechanical off. The sleeping state is further divided into four sub states (S1 to S4) with gradations regarding the components which are shut down.

Each device can be set in four different states from D0, which is fully on to D3 which is off. The processor itself can be in four states as well. These are C0, the operating state to C3, which is a sleep state where the cache is necessarily kept coherent any more.

ACPI is used by most desktop operating systems as a main implementation for energy management and the monitoring of energy consumption.

B. How energy management in mobile computing differs from ACPI

Mobile operating systems are derived from their desktop pendants. They bring own concepts inspired by conventional solutions mostly re-implementing common features [19]. For example the Android developers replaced the basic *glibc* library which handles basic system functionality (e.g. system calls, open, malloc, printf, exit etc) in the standard JVM for desktop systems. Androids main development language is JAVA. The custom *Dalvik* JVM uses a library called bionic which is designed for low energy consumption, fast execution and low memory usage [22]. It supports native Android services as well as a very fast threading implementation [19].

On a lower level Android has an kernel extension the *PowerManager* which extends basic Linux energy management features with the *WakeLocks* concept for mobile devices and various other memory and energy saving features.

Applications can request a specific *WakeLock* and therefore force the system into a specific state. There are two additional *WakeLocks*, one that provides the possibility to wake the devices from power off and another to lighten the screen for alerts and notifications. A brief overview of *WakeLock* is given in table I. Complementary one can register device drivers for *WakeLocks* so that they are notified when a specific lock is requested from user space [22]. This part offers a possibility to interact with the system and maybe schedule locks in more efficient way. Other mobile operating systems bring similar features but are less well documented.

V. NECESSARY TECHNOLOGIES FOR AN EAL

For creating an EAL as described in chapter I one needs specific technologies. This chapter will outline which technologies are necessary to create an EAL. A in deep description of the functionality of these technologies is not part of the paper but references for more information are given.

These technologies are either necessary for monitoring applications, components or the whole system and the energy consumption in a specific situation or they are needed to manipulate the energy consumption on a device directly or indirectly.

A. Modelling a Mobile Device

Modelling a device can be helpful to analyze the energy consumption of a device in pre defined states [24]. A model of a device can also be used in simulations for better understanding its behavior. On a very low level a model driven approach is followed in [5]. The described COMPLEX framework contains the following parts:

- MDA design entry
- Executable Specification
- Estimation & model generation
- Simulation
- Exploration & optimization

After designing and generating the model and using it for a simulation the results can be analyzed. The framework contains functionalities for gathering dynamic and static power information. But one have to keep in mind that this a very low level approach. Mapping the running application to the results gained by these analysis is a rather difficult task.

In [24] an energy consumption is created by utilizing finite state machines. Each state in the finite state machine represents an amount of information about the system that was analyzed. The model created was well suited for analyzing applications regarding to network communication. Other applications were examined but the given approach did not seem to match.

An additional approach was presented in [17]. A user study was performed to analyze user behavior and generate a model representing the users activity.

For an EAL one needs probably two models. One similar to the model generated with the COMPLEX framework to acquire information about the system behavior [5].

Another model should represent certain states of the system alike the approach in [10]. This model should be generated by the insights obtained with the various system analysis. A possible representation for this kind of model are Markov chains. Markov chains consist of a finite number of states where the system can move randomly in a certain states. State changes are called transitions and have a probability. So every state could represent states of components and applications and there is probability that the system could move to another state where components and applications behave differently [25].

B. Dynamic Analysis for Program Comprehension

Dynamic program analysis is a way to evaluate program characteristics at run-time. In [8] is described how to understand application behavior without doing complete reverse engineering. It is described how to combine static, dynamic and concept analysis to not only gain an understanding of how an application works but also how to identify relationships between certain parts of the application structure.

They define a set of features (features are in this case realized functional requirements), identify a suitable scenario in which the defined features are used, execute the application and generate a profile, do a concept analysis for the result and identify relationships between scenarios and parts of the application. A complementary step is to perform a static dependency analysis as the last step.

The principals described in [8] are only suited for functional features as they can be more easily assigned to components of the application then to non functional features. In this case concept analysis is used mainly to recognize which part (sub program) of the application is used when specific features are used.

As described in [26] concept analysis can add more to the software engineering process. It can be used to extract the knowledge of the application structure into triples (analogous to ontologies). This information can then be used for graph based queries to gain further insight.

1) Aspect Oriented Programming: Aspect Oriented Programming (AOP) is a technique typically used if specific functionality is scattered through an application. A basic example is a logging service. Such a service would be defined once and used in various areas of the application. The program logic would probably interfere with the program analysis (or logging) logic. AOP provides a set of techniques to handle such use cases more elegant. The aspect would be created once and then the code used by the logger would be called from within [27].

In the given case, if one can identify certain parts of the program using features that are thought to consume a high amount of energy, an aspect can be added to further investigate the case. The usage of AOP to identify certain features in applications is described in [8].

AspectJ is a JAVA Framework for AOP and the de facto standard aspect oriented programming in the JAVA world. Every JAVA program can be utilized as an AspectJ program because AspectJ works in a way that the JAVA VM recognizes it as a valid JAVA program. Therefore no additional requirements for the usage of AspectJ are needed.

Listing 1 shows a simple example of an aspect created with AspectJ. The aspect is executed before the execution of *exampleMethod*. This works solely by annotating the class and the method. Other annotations are @After, @AfterReturning or @Around.

Listing 1. Simple AOP Example

```
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;

@Aspect
public class LoggingAspect {

        @Before("execution(* ExampleClass.exampleMethod(..))")
        public void logBefore(JoinPoint joinPoint) {

                        //do logging
        }
}
```

The configuration can be done via XML as well. What is the best solution will turn out as the work on the EAL progresses.

C. Electrical Pattern Recognition for Applications

Another approach is to measure electrical consumption of a system and included entities directly. During the measurement of the system a graph with characteristic peaks is created and it is possible to identify entities in this system by the characteristics of the peak. This technology origins in electrical household management but can certainly transformed for the usage of measuring energy consumption of mobile devices. Originally smart meters are used to measure the energy consumption of a whole household and afterwards so called sub meters are used to measure the energy consumption of individual devices. This procedure needs to be adapted for the domain of mobile computing.

Once this is done and the characteristic peaks of specific applications are identified it is relatively easy to make a long term analysis and gain information about the exact energy consumption of the whole system and every application in specific situations [28].

If one can combine these results with the results of the dynamic analysis mentioned in chapter V-B statements about the energy consumptions about parts of the implementation of monitored applications can be made.

D. User State Recognition

In order to influence a system in a proper way one needs information about the state in which the user is situated. For example it should be possible to turn 3G connection off at home because Wi-Fi is available and consumes much less energy [10]. To gain these information one needs to use data provided by the device. Modern social applications like Facebook, MySpace etc. provide location based services. A similar approach can be done with data gathered by sensors and provide profiles for applications and components.

According to [10] simple user states like *walking, vehicle, resting, home* etc are identified with accelerometer, Wi-Fi, GPS and microphone with an accuracy of over 90%. Using this information and a static behavior for each states reduces the energy consumption about of 70%.

1) Gather User Information Through Sensors: Today's mobile computer devices carry a lot of different sensors that can be used to gain information about user behavior and or user state. As described in [10] modern smart phones sensing capabilities include Wi-Fi, Bluetooth, GPS, audio, video, light sensors, accelerometers and more. With these capabilities it is possible to monitor the user's surroundings.

All these sensors are accessible through regular APIs as described in [17].

VI. Classification of the Energy Abstraction Layer

As shown in figure 1 the EAL should be located above the OS layer because its design should be as OS independent as possible. That includes an abstract specification of all the

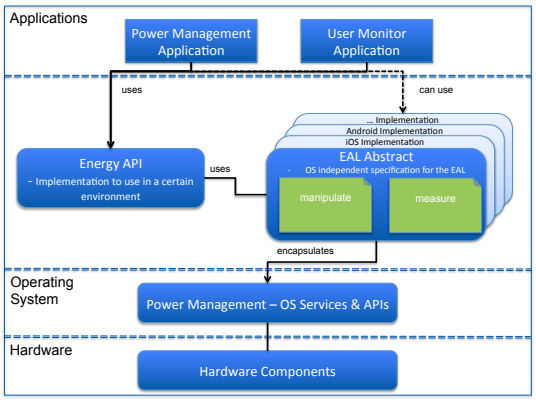

Fig. 1. EAL Architecture

functionality it should include. This specification can be instantiated for a specific environment (e.g. Android, iOS etc...). This specification encapsulates all energy saving features of possible operating systems into one interface.

The specification itself is divided in a *measurement* and *management* part. The measurement part should include the functionality to collect information about system behavior and energy consumption in specific situations. This can be done either on a real device or in a simulation environment as mentioned in [24]. Similar to the concept outlined in [10] where sensing energy consumption on a Nokia N95 was done in cooperation with Nokia one needs to evaluate the energy consumption of individual components in connection to specific applications. This can be done via the measurement part of the EAL. A possible use case for the results gained by this process is outlined in chapter VII.

Whereas the measurement part bundles only methods to monitor system behavior the management part delivers the functionalities to influence the energy consumption of a device. Component steering, state changing and possibilities to influence the scheduling are located in this part of the application. Modern operating system normally don not grant developers access to this part of the system [14] but to improve energy efficiency on mobile devices one needs to overcome such boundaries.

The API at last aggregates advanced functionalities and presents a structured layer for application developers so one can use the benefits of the EAL without digging too deep. Also the methods provided by the API are more advanced and have much less interface character then that what is provided by the EAL.

Actually if one wants one can directly access the EAL as well.

VII. Possible Use Cases for an EAL

The EAL does not stand for its own but is designed with certain use cases in mind. Some of the use cases mentioned in chapter VI are presented below.

A. Improve Energy Consumption via Dynamic User States

One possible application which is inspired by the work presented in [10] and can be realized with the energy abstraction layer is a user activity logging application which recognizes user states and gathers information about the user behavior and environment. For better analysis of the information gathered by this application an interface for a remote database of user states is added to the architecture.

The application is divided into different parts. As mentioned above there will be a regular sync with a remote server to update the server information and add possibly new states and their descriptions to the local database. The *Energy API* will provide the interfaces for devices sensors and will unify the access to these.

A classification module is able to determine the actual user state from the date gained by the devices sensors and information stored in the local database. The user state influences the devices behavior and will also be displayed in the user interface. Regular updates to the user about the work of the application is necessary to create an insight about the impacts of the energy abstraction layer.

User states can possibly be modelled with the help of Markov chains as mentioned in chapter V-A. Markov chains consist of a finite number of states in an environment where these states can change randomly. The state changes are called transitions and can happen with a certain probability [29].

For each state a component and or application of the target devices can be defined. With a suitable model and the appropriate simulation one can make predictions about the expected energy consumption in certain situations (states of whole the device).

B. Improved Power Management and Scheduling

Another possible application using either the implemented specification or the *Energy API* is an advanced power manager with an optional scheduling component. As described in chapter IV todays mobile operating systems are derived from their desktop pendants but are far away from using as effective energy saving techniques as they could. A basic example is the handling of components. They can be either on, off or sleeping. If an application *wakes* a certain component and a short while after another applications does so as well a huge amount of energy is wasted.

Here an advanced power manager could come in handy, either by providing a possibility that applications can register themselves or by monitoring the whole system.

Another possibility would be that this power manager can schedule component wake up much more differentiated then it is done by now.

VIII. Conclusion

This paper gave an survey about the possibilities and requirements for creating an energy abstraction layer based on information from user behavior and the knowledge about characteristics of certain applications and components.

Dynamic Analysis as mentioned in chapter V-B can be used to gain information about applications at runtime and how they correspondent with user behavior. Talking of user behavior, this can be monitored with various different sensors as described in chapter V-D.

The architecture that might bring all these aspects together is described in chapter VI. With these concpets hopefully further research activities will follow which refine the concept. A sample implementation on one specific platform (e.g. Android as mentioned in chapter IV) would be very helpful to prove that the concept does actual work.

In chapter IV it is mentioned that there are various works in different fields touching the topic of an energy abstraction layer but never bringing user behavior analysis and state recognition together with optimisation of applications using software reengineering services.

A. Next Steps

The next steps for creating an EAL are probably the creation of a valid model of a device to determine which data is really needed. There is a danger that a huge amount of data is collected but no helpful information can be generated because there is no easy way to analyse so much data. Regarding to the experiences mentioned in literature only the amount of data collected via dynamic analysis is very high [9].

Another step is to define platform dependent specifications for the information that can be obtained in the platform at hand and then unify the specifications to an abstract definition for the EAL. This may take some time because one has to evaluate at least a few different platforms (e.g. iOS, Android etc...). Afterwards one can begin to specify the interfaces needed for information retrieval on the various platforms.

Maybe at this point it makes sense to build a prototype on only one platform to see how it all works out. So one can create a platform dependent model and start analysing the systems behavior.

At the very last reengineering service should be designed to:

- Modifying an application for using the energy abstraction layer
- Removing or restructuring energy wasters found with the dynamic analysis

Hopefully it is possible to make a significant impact on energy consumption in the domain of mobile computing.

REFERENCES

[1] L. S. Brakmo, D. A. Wallach, and M. A. Viredaz, "usleep: A technique for reducing energy consumption in handheld devices," in *IN PROC. INT. CONF. MOBILE SYSTEMS, APPLICATIONS, AND SERVICES*, 2004, pp. 12–22.

[2] W. Nebel, M. Hoyer, K. Schroeder, and D. Schlitt. (2009) Untersuchung des potentials von rechenzentrenuebergreifendem lastmanagement zu reduzierung des energieverbrauchs in der ikt. [Online]. Available: http://www.offis.de/fileadmin/Chefredakteur_files/PDFs/Pressemitteilungen/2009-11-19_OFFIS-Studie_zum_Lastmanagement_in_Rechenzentren__Veroeffentlichung_.pdf

[3] K. Roy and M. C. Johnson, "Software design for low power," in *Low power design in deep submicron electronics*, W. Nebel and J. P. Mermet, Eds. Berlin: Springer, 1997, ch. 6.3, pp. 433–460.

[4] W. M. F. et al. (2008, 10) Performance conserving method for reducing power consumption in a server system. Google. [Online]. Available: http://www.google.de/patents/US7444526?dq=reducing+power+consumption+hardware

[5] (2011) COMPLEX - COdesign and power Management in PLatform-based design space EXploration. Offis.

[6] M. B. Motlhabi, "Advanced android power management and implementation of wakelocks," 2008.

[7] M. P. Robillard, "What Makes APIs Hard to Learn? Answers from Developers," *IEEE Software*, vol. 26, no. 6, pp. 27–34, Nov. 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5287006

[8] T. Eisenbarth, R. Koschke, and D. Simon, "Aiding program comprehension by static and dynamic feature analysis," in *Software Maintenance, 2001. Proceedings. IEEE International Conference on*. IEEE, 2001, pp. 602–611. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=972777

[9] T. Gschwind, "Improving dynamic data analysis with aspect-oriented programming," *Software Maintenance and Reengineering, 2003*, no. 26-28 March 2003, 2003. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1192434

[10] Y. Wang, J. Lin, M. Annavaram, Q. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh, "A framework of energy efficient mobile sensing for automatic user state recognition," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*. ACM, 2009, pp. 179–192. [Online]. Available: http://dl.acm.org/citation.cfm?id=1555835

[11] D. Binkley, "Source Code Analysis : A Road Map Source Code Analysis : A Road Map," *Future of Software Engineering, 2007. FOSE '07*, no. 23-25 May 2007, pp. 104 – 119, 2007.

[12] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 2009, pp. 280–293. [Online]. Available: http://dl.acm.org/citation.cfm?id=1644927

[13] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*. USENIX Association, 2010, pp. 21–21. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855861

[14] M. Motlhabi, "Advanced Android Power Management and Implementation of Wakelocks," *cs.uwc.ac.za*, pp. 1–5. [Online]. Available: http://www.cs.uwc.ac.za/~mmotlhabi/apm2.pdf

[15] K. Naik, "A survey of software based energy saving methodologies for handheld wireless communication devices," *Department of Computer and Electrical Engineering, University of Waterloo, Tech. Rep*, vol. 3, 2010. [Online]. Available: http://swen.uwaterloo.ca/~knaik/energy.pdf

[16] C. Harris and V. Cahill, "Exploiting user behaviour for context-aware power management," in *Wireless And Mobile Computing, Networking And Communications, 2005.(WiMob'2005), IEEE International Conference on*, vol. 4. IEEE, 2005, pp. 122–130. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1512959

[17] A. Shye, B. Scholbrock, G. Memik, and P. Dinda, "Characterizing and modeling user activity on smartphones: summary," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 1. ACM, 2010, pp. 375–376. [Online]. Available: http://dl.acm.org/citation.cfm?id=1811094

[18] I. Cooperation, "ACPI Component Architecture User Guide and Programmer Reference," *Interface*, 2011.

[19] F. Maker, "A Survey on Android vs. Linux," *University of California*, pp. 1–10, 2009. [Online]. Available: http://handycodeworks.com/wp-content/uploads/2011/02/linux_versus_android.pdf

[20] M. Motlhabi, "Android Power Panagement," *cs.uwc.ac.za*. [Online]. Available: http://www.cs.uwc.ac.za/~mmotlhabi/apm.pdf

[21]

[22] K. Paul and T. K. Kundu, "Android on Mobile Devices: An Energy Perspective," *2010 10th IEEE International Conference on Computer and Information Technology*, no. Cit, pp. 2421–2426, Jun. 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5578292

[23] A. Developer. (2011) ios developer library. Apple. [Online]. Available: https://developer.apple.com/library/ios/#documentation/iphone/conceptual/iphoneosprogrammingguide/Introduction/Introduction.html

[24] R. Palit and A. Singh, “Modeling the energy cost of applications on portable wireless devices,” *Modeling, analysis and simulation of wireless*, 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1454562

[25] W.-K. Ching and M. K. Ng, *Markov Chains: Models, Algorithms and Applications (International Series in Operations Research & Management Science)*, softcover reprint of hardcover 1st ed. 2006 ed. Springer, 11 2010. [Online]. Available: http://amazon.com/o/ASIN/1441939865/

[26] T. Tilley, R. Cole, P. Becker, and P. Eklund, “A survey of formal concept analysis support for software engineering activities,” *Formal Concept Analysis*, pp. 250–271, 2005. [Online]. Available: http://www.springerlink.com/index/0dnu56mmegnxk6ee.pdf

[27] J. D. Gradecki and N. Lesiecki, *Mastering AspectJ: Aspect-Oriented Programming in Java*, 1st ed. Wiley, 3 2003.

[28] T. Klingenberg, “Smart Submetering,” *Messtechnik*, 2010.

[29] S. Chen, “CONSISTENCY AND CONVERGENCE RATE OF MARKOV CHAIN QUASI MONTE CARLO WITH EXAMPLES,” no. August, 2011.

Messen und Beeinflussen des Energieverbrauchs von Android-Systemen

Michael Falk
Carl von Ossietzky Universität
Department für Informatik
Oldenburg, Deutschland
michael.falk@informatik.uni-oldenburg.de

***Zusammenfassung*—Da mobile Endgeräte immer leistungsfähiger werden und mehr Energie verbrauchen, wird das Thema Energieeffizienz zunehmend wichtiger. Aktuell fehlt es allerdings noch an Techniken und Methoden, um auf hoher Ebene, zum Beispiel auf Applikationsebene, effizient Energie zu sparen. Um zu sehen, an welchen Stellen Energie gespart werden kann, ist es erst einmal notwendig, den Energieverbrauch messen zu können und Möglichkeiten der Beeinflussung zu kennen. Am Beispiel von Android soll dieser Vortrag einen kurzen Einblick geben, welche Möglichkeiten bereits vorhanden sind, den Energieverbrauch zu messen und zu beeinflussen.**

***Index Terms*—Android; Energieverbrauch; Energieeffizienz;**

I. Motivation

In den letzten Jahren hat sich der Stand der Technik rasant weiterentwickelt. Diese Entwicklung zeigt sich auch im Sektor der mobilen Endgeräte und der eingebetteten Systeme. Dass man mit Smartphones außer dem Telefonieren beispielsweise im Internet surfen, Musik hören, Spiele spielen, Videos anschauen und GPS-Tracking durchführen kann, war früher nicht vorstellbar, ist aber für viele heutzutage ganz normal. Um diesem Trend zu begegnen, verlangt es nach mehr Leistung und einem größeren Funktionsumfang. Mehr Leistung bedeutet, dass mehr Energie verbraucht wird und die Akkulaufzeit sich somit verringert. Eine hohe Akkulaufzeit ist aber etwas, worauf Anwender von Mobilgeräten zwecks Mobilität und Flexibilität nicht verzichten möchten. Der Fortschritt bei der Entwicklung von Akkus ist momentan nicht so groß, wie die wachsenden Bedürfnisse an Energie [1]. Deswegen ist es wichtig, dass Techniken und Methoden entwickelt werden, um den Energieverbrauch zu reduzieren. Als Beispiel sei hier die Idee des *Energy Abstraction Layer* von Mirco Josefiak [2] anzumerken.

Für die konkrete Umsetzung ist es erst einmal wichtig zu wissen, wie der Energieverbrauch auf einer Zielplattform gemessen und beeinflusst werden kann. Da Android die meistgenutzte Plattform ist, liegt mein Schwerpunkt auf der Android-Architektur.

Es geht in dieser Ausarbeitung darum, herauszufinden, welche Möglichkeiten bereits vorhanden sind und unter Umständen schon genutzt werden, um den Energieverbrauch messen und beeinflussen zu können. Diese Ausarbeitung hat keinen Anspruch auf Vollständigkeit in Bezug auf die vorhandenen Möglichkeiten, vielmehr möchte sie einen Einblick in die Thematik vermitteln.

Um Android besser zu verstehen, wird in Abschnitt II die Android-Architektur mit ihren einzelnen Schichten vorgestellt. In Abschnitt III wird vorgestellt, welche Möglichkeiten es gibt, den Energieverbrauch zu messen. Der Fokus richtet sich auf die Nutzung der Android-API. Abschnitt IV beschäftigt sich damit, welche Möglichkeiten es gibt, den Energieverbrauch zu beeinflussen. Die Ausarbeitung schließt mit einer Zusammenfassung.

II. Die Architektur von Android

Aus Anwendersicht ist Android ein Betriebssystem, das von vielen auf mobilen Geräten genutzt wird. Ein Entwickler würde Android allerdings eher als eine Plattform bezeichnen, mit deren Hilfe Anwendungen programmiert werden können. Diese Plattform stellt dann Betriebssystem, Zwischenschichten und Applikationen bereit. Da Android von Google als Open-Source-Projekt entwickelt wird, lässt es sich auf eigene Bedürfnisse hin anpassen.

In Abbildung 1 ist die Architektur von Android gezeigt. Es handelt sich um eine Schichtenarchitektur, die aus vier Schichten besteht. Im Folgenden werden die einzelnen Schichten kurz beschrieben.

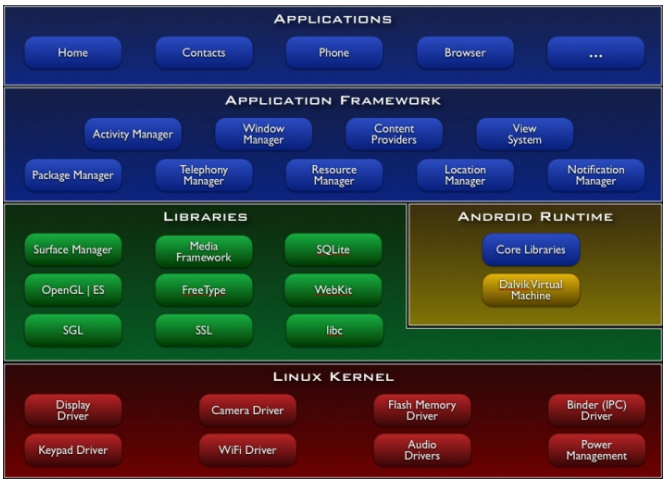

Abbildung 1. Android Architektur [3]

Linux-Kernel:
Android basiert auf der Version 2.6 des Linux-Kernels. Diese wird allerdings von Google für mobile Anforderungen etwas

angepasst. Der Linux-Kernel übernimmt grundlegende Betriebssystemfunktionalitäten, wie zum Beispiel Speicherverwaltung, Prozessverwaltung, Energiemanagement und Verwaltung der Treiber. Er kapselt damit die Hardware von den oberen Schichten ab.

Android-Runtime und Bibliotheken:
Diese Schicht stellt diverse Systembibliotheken bereit, die von der Laufzeitumgebung genutzt werden. Die Android-Runtime, die Dalvik-Virtual Machine, führt die Funktionalität der oberen Schichten aus. Sie enthält neben den wichtigsten Java-Bibliotheken zusätzlich androidspezifische Bibliotheken.

Application-Framework:
Das Anwendungsframework stellt Bausteine zum Entwickeln von Anwendungen auf oberster Schicht bereit. Die Verwaltung der Anwendungen beinhaltet z.B. die Steuerung des Lebenszyklus, die Regelung von Datenzugriffen oder die Verwaltung von Ressourcen.

Anwendungsschicht:
Die höchste Schicht sind die Anwendungen selbst. Diese werden normalerweise über das Android SDK [1] programmiert und sind für die Interaktion mit dem Benutzer zuständig. Android liefert schon viele Anwendungen standardmäßig mit.

Vgl. dazu [3].

III. Messen des Energieverbrauchs

Dieser Abschnitt konzentriert sich hauptsächlich darauf, welche Möglichkeiten Android direkt liefert, um den Energieverbrauch zu messen. Auf andere Verfahren sei aus diesem Grund lediglich hingewiesen.

A. Standardfunktionen von Android

Standardmäßig kann Android einige Energieinformationen anzeigen. Neben dem Batteriestand in Prozent kann die Spannung, die Temperatur, die eingesetzte Akkutechnologie, der Status, und die Intaktheit des Akkus angezeigt werden. Der Zugriff auf die Energiedaten erfolgt über ein Observer-Pattern, das durch die Klassen der Android-API [2] `BroadcastReceiver` und `Intent` realisiert wird. `BroadcastReceiver` müssen registriert werden. Sie horchen und reagieren dann auf Änderungen. Sie sind also die Observer. `Intents` legen fest, was beobachtet wird. Anders als beim klassischen Observer-Pattern legt man nicht fest, welche Klasse beobachtet wird, man legt lediglich fest, auf welches Ereignis man reagieren möchte. Das ermöglicht eine Abstrahierung von konkreten Implementierungen. Das `Intent ACTION_BATTERY_CHANGED` lässt einen `BroadcastReceiver` auf Änderungen des Akkus reagieren. Zum Auslesen und Interpretieren der Energieinformationen enthält die Klasse `BatteryManager` Konstanten.

```
public class Main extends Activity {
  private TextView contentTxt;
  private BroadcastReceiver mBatInfoReceiver = new BroadcastReceiver(){
    @Override
    public void onReceive(Context arg0, Intent intent) {
      // TODO Auto-generated method stub
      int level = intent.getIntExtra("level", 0);
      contentTxt.setText(String.valueOf(level) + "%");
    }
  };

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);
    contentTxt = (TextView) this.findViewById(R.id.monospaceTxt);
    this.registerReceiver(this.mBatInfoReceiver,
    new IntentFilter(Intent.ACTION_BATTERY_CHANGED));
  }
}
```

Abbildung 2. Batteriestand ermitteln [4]

Abbildung 2 zeigt, wie der Batteriestand ermittelt werden kann.
Die Methode `intent.getIntExtra("level",0)` liest den Batteriestand aus. Das Auslesen der anderen Energieinformationen erfolgt unter Angabe anderer Konstanten analog dazu. Neben den Standardfunktionen bieten viele Apps ähnliches und mehr. Als Beispiel sei hier die App *PowerTutor* anzuführen, die über ein physikalisch ermitteltes Energiemodell den Energieverbrauch von Apps und Komponenten anzeigen kann [5]. Ein anderen Weg gehen die Autoren von [6]. Sie stellen ein Verfahren vor, um zur Laufzeit, den Energieverbrauch abschätzen zu können. Indem sie ein System-Level-Modell verwenden, kommen sie ohne physikalische Messungen auf Komoponentenebene aus.

IV. Beeinflussen des Energieverbrauchs

In diesem Abschnitt geht es um das Beeinflussen des Energieverbrauchs. Der Abschnitt beschränkt sich auf einige Punkte, die hier erläutert werden sollen.

A. Nativen Code benutzen

Seit Juni 2009 ist es möglich, über das Android NDK [3] nativen Code zu schreiben und einzubinden, falls gewünscht. Laut Google resultiert das Schreiben nativen Codes nicht unbedingt in einer Performancesteigerung, dafür aber in einer Steigerung der Komplexität [7]. Es lohne sich also nicht immer nativen Code zu schreiben. Performancesteigerungen erreiche man vor allem bei CPU-lastigen Operationen, die wenig Speicher verwenden, wie zum Beispiel Signalverarbeitung oder physikalische Simulationen. Ein weiterer Vorteil, den das NDK mit sich bringt, ist, dass man über das NDK hochoptimierte Prozessormethoden benutzen kann. Erreicht man durch den Einsatz des NDK eine Performancesteigerung, beeinflusst das auch den Energieverbrauch. Man kann sagen, dass in der Regel eine Steigerung der Performance zu einer Senkung des Energieverbrauchs führt. Ob es nun Sinn macht, das Android

[1] SDK: Software Development Kit
[2] API: Application Programming Interface
[3] NDK: Native Development Toolkit

NDK zu verwenden, muss im einzelnen entschieden werden und hängt stark von dem Aufgabenbereich und den eigenen Anforderungen ab.

In [8] vergleicht der Autor iterative und rekursive Implementierungen des Fibonacci-Algorithmus von nativem und normalem Java-Code. Abbildung 3 zeigt, dass vor allem beim rekursiven Algorithmus ein immenser Geschwindigkeitsvorteil bei der Nutzung des NDK festzustellen ist. Beim iterativem Algorithmus ist der Unterschied laut Autor nicht so groß.

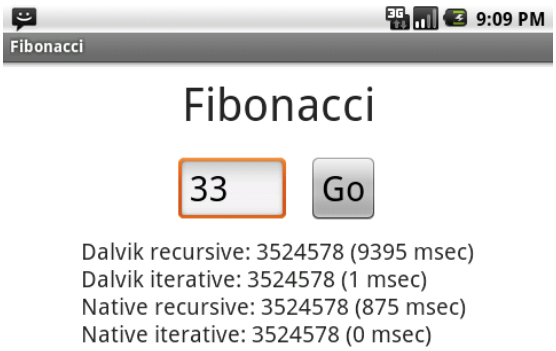

Abbildung 3. Vergleich: Nativer Code und Java-Code (Dalvik-Code) [8]

B. *Context-Aware Resource Control*

Die Idee des *Context-Aware Resource Control* ist, Komponenten und Dienste abzuschalten, wenn sie nicht gebraucht werden und dadurch den Energieverbrauch zu senken. Das Abschalten der Komponenten und Dienste soll dabei kontextbezogen, das heißt abhängig von der Situation und abhängig vom Nutzerverhalten, erfolgen. In [9] stellen die Autoren ein *Context-Aware Resource Control* System vor. Sie definieren verschiedene Benutzerszenarien und versuchen über die Sensoren des mobilen Gerätes den Kontext, in dem das Gerät benutzt wird, herauszufinden. Anschließend wird versucht zu berechnen, welche Ressourcen minimal benötigt werden. Überflüssige Ressourcen werden dann abgeschaltet und freigegeben. Sie schreiben, dass bei ihren Versuchen eine Reduktion des Energieverbrauchs von bis zu 45% festzustellen sei.

Generell lässt sich sagen, dass durch Abschalten von Komponenten der Energieverbrauch gesenkt wird. Über die Android-API kann man diese Komponenten kontrollieren. Im Folgenden ein Beispiel, wie man das WLAN, sofern vorhanden, über die Android-API deaktivieren kann:

```
WifiManager wifi = (WifiManager)
        getSystemService(Context.WIFI_SERVICE);
wifi.setWifiEnabled(false);
```

Wie das WLAN lassen sich auch andere Komponenten aktivieren bzw. deaktivieren. Momentan gibt es viele Apps, die automatisiert Komponenten aktivieren und deaktivieren. Forschungsbedarf gibt es in diesem Bereich aber sicherlich weiterhin.

C. *Bad Smells*

Schlechte Programmierung von Anwendungen führt zu einem höheren Energieverbrauch. Wenn man auf Energieverbrauch bezogene *Bad Smells* verhindert, hat das einen positiven Effekt auf den Energieverbrauch. Es ist deswegen wichtig, *Bad Smells* zu erkennen.
Im Folgenden werde ich als Beispiel für ein *Bad Smell* auf den Umgang mit Hardwareressourcen eingehen.

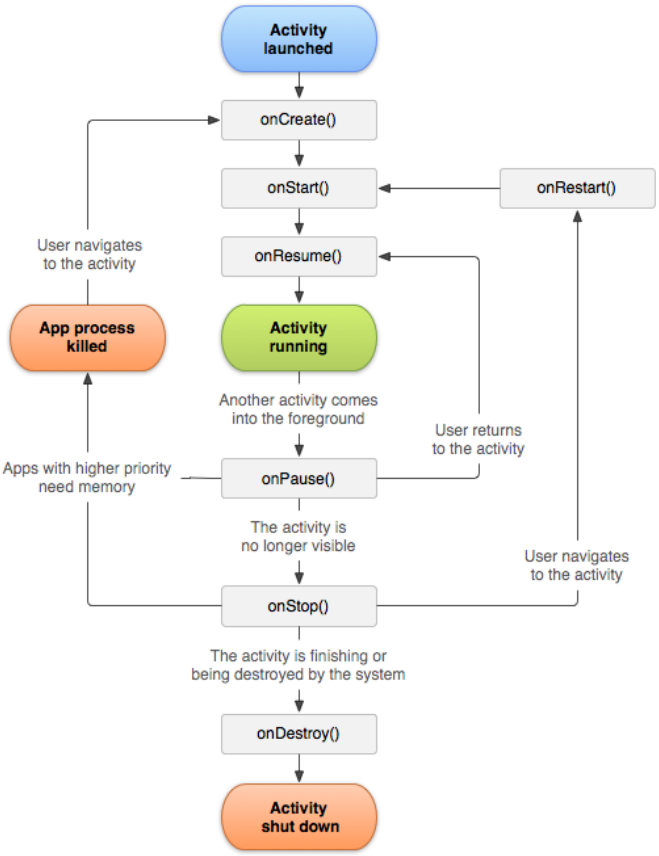

Abbildung 4. Lebenszyklus von Aktivitäten [7]

Um den Umgang mit Hardwareressourcen richtig zu verstehen, ist es wichtig den Lebenszyklus von Aktivitäten zu verstehen (siehe Abbildung 4). Beim Start einer Anwendung mit ihrer zugehörigen Aktivität, wird die Methode `onCreate` aufgerufen. Die Methode `onPause` wird beim Pausieren der Aktivität aufgerufen. Wird sie fortgesetzt, wird die Methode `onResume` ausgeführt. Nachdem eine Aktivität gestartet wurde, passiert es häufig, dass sie pausiert werden muss, weil eine andere Aktivität den Fokus des Vordergrundes erhält (zum Beispiel ein Telefonanruf). Der Anwendungsentwickler muss dafür sorgen, dass nicht mehr benötigte Ressourcen beim Pausieren freigegeben und beim Fortsetzen der Aktivität wieder angefordert werden. Das sollte in den Methoden `onPause()` und `onResume` geschehen.

Das nachfolgende Beispiel zeigt, wie man richtig mit Hardwareressourcen umgeht. Beim Pausieren der Aktivität wird die Kamera freigeben und verbraucht somit nicht unnötig Energie.

```
@Override
public void onResume() {
    ..
    camera = Camera.open();
    ..
}

@Override
public void onPause() {
    ..
    camera.release();
    ..
}
```

Im Gegensatz dazu zeigt das nächste Beispiel, wie man es nicht machen sollte. Da die Kamera beim Start der Aktivität angefordert wird und erst beim Zerstören der Aktivität wieder freigegeben wird, verbraucht die Kamera beständig Energie, auch wenn die Aktivität nicht aktiv ist.

```
@Override
public void onCreate(Bundle savedInstanceState) {
    ..
    camera = Camera.open();
    ..
}

@Override
public void onDestroy() {
    ..
    camera.release();
    ..
}
```

Die Beispiele sind aus [10] entnommen. Abbildung 5 zeigt, welche Auswirkungen ein falscher Umgang mit Hardwareressourcen hat. Man sieht, dass unnötigerweise Strom fließt. Grundsätzlich soll durch das Beseitigen von *Bad Smells* der Energieverbrauch gesenkt werden.

V. Zusammenfassung

Diese Ausarbeitung gibt einen Einblick in die Möglichkeiten in Android-Systemen, den Energieverbrauch zu messen und zu beeinflussen. Außerdem wird versucht, ein besseres Verständnis für die Android-Architektur zu vermitteln. Es wird aufgezeigt, dass die Android-API entsprechende Methoden liefert, die von Anwendungsentwicklern genutzt werden können.

Festgestellt wurde, dass zur Beeinflussung des Energieverbrauchs unter Umständen das Android NDK genutzt werden kann. Eine weitere Möglichkeit, um den Energieverbrauch zu senken, ist das Abschalten von einzelnen Komponenten. In diesem Bereich gibt es sicherlich noch Forschungsbedarf. So könnten zum Beispiel nicht nur Benutzerszenarien erstellt werden, sondern auch Nutzerprofile, die versuchen, die Gewohnheiten des Nutzers in die Ressourcenkontrolle mit einfließen zu lassen. Außerdem könnten noch mehr Informationen zur Ressourcenkontrolle mit einbezogen werden.

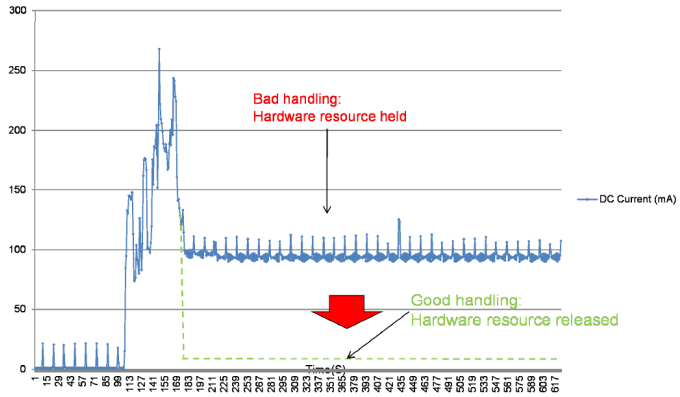

Abbildung 5. Umgang mit Hardwareressourcen [10]

Als letzter Punkt wurde das Thema *Bad Smells* behandelt. Als Beispiel wurde der Umgang mit Hardwareressourcen behandelt. Dieses Thema ist sehr wichtig, da *Bad Smells* den Energieverbrauch negativ beeinflussen und es daher wünschenswert ist, diese zu beseitigen. Der nächste Schritte wäre, erst einmal weitere *Bad Smells* zu entdecken. Danach kann dann versucht werden, ein Refactoring für die erkannten *Bad Smells* zu entwickeln.

Abschließend hoffe ich, dass diese Ausarbeitung anderen Material und Ideen liefert, von denen sie und die Domäne profitieren können.

Literatur

[1] F. Ding, F. Xia, W. Zhang, X. Zhao, and C. Ma, "Monitoring Energy Consumption of Smartphones," in *Proc. Physical and Social Computing Internet of Things (iThings/CPSCom) Int. Conf. and 4th Int. Conf. Cyber*, 2011, pp. 610–613.

[2] M. Josefiak, "Towards an Energy Abstraction Layer," 2012.

[3] P. Plakhin, "Sicherheitsarchitektur von Android," 2011.

[4] A. Mendoza, "Getting Battery Information on Android ," Januar 2009, zuletzt abgerufen am 31.03.2012. [Online]. Available: http://www.tutorialforandroid.com/2009/01/getting-battery-information-on-android.html

[5] M. Gordon, L. Zhang, B. Tiwana, and L. Yang, "PowerTutor," 2010, zuletzt abgerufen am 29.03.12. [Online]. Available: http://ziyang.eecs.umich.edu/projects/powertutor/index.html

[6] Y. Xiao, R. Bhaumik, Z. Yang, M. Siekkinen, P. Savolainen, and A. Ylä-Jääski, "A System-Level Model for Runtime Power Estimation on Mobile Devices," in *Proc. Physical and Social Computing (CPSCom) Green Computing and Communications (GreenCom) IEEE/ACM Int'l Conf. & Int'l Conf. Cyber*, 2010, pp. 27–34.

[7] "Android Developers," Google Inc., zuletzt abgerufen am 29.02.12. [Online]. Available: http://developer.android.com

[8] M. Gargenta, "Using NDK for Performance - Dalvik Versus Native," April 2010, zuletzt abgerufen am 29.03.12. [Online]. Available: http://marakana.com/forums/android/examples/96.html

[9] K. Nishihara, K. Ishizaka, and J. Sakai, "Power Saving in Mobile Devices Using Context-Aware Resource Control," in *Proc. First Int Networking and Computing (ICNC) Conf*, 2010, pp. 220–226.

[10] "Android Application Coding Guidelines," Sony Developer World, April 2010, zuletzt abgerufen am 29.02.12. [Online]. Available: http://dl-www.sonymobile.com/cws/download/1/788/263/1271920135/dw-300012-Android_Power_Save.pdf