Towards an energy-consumption based complexity classification for resource substitution strategies

Hagen Höpfner
Bauhaus-University of Weimar
Bauhausstraße 11
99423 Weimar, Germany
hoepfner@acm.org

Christian Bunse
University of Mannheim
A 5, 6, Gebäudeteil B, Seminargebäude
68131 Mannheim, Germany
Christian.Bunse@informatik.uni-mannheim.de

ABSTRACT

Protecting the environment by saving energy and thus reducing carbon dioxide is one of today's hottest topics and is of a rapidly growing importance in the computing domain. In addition, to ecological reasons here issues such as the up-time of mobile or embedded devices, battery charge cycles etc. are key problems. Recent studies have shown that software has a major impact onto the energy consumption of the device it is executed on. Thus, intelligent selection and assembly of software components promises significant savings. However, this requires knowledge about how much energy a (software) component consumes. In other words, a classification scheme following the idea of the European Union energy label is required. This paper discusses recent findings and first ideas of establishing an energy classification scheme for software, using the 'big-O' notation as its general metaphor. The scheme is motivated, introduced and validated by using resource substitution strategies, as one means for optimizing energy consumption via software adaptation. We demonstrate that the classification scheme can be used to characterize the fitness of a strategy and/or algorithm. Furthermore, we discuss to use such energy labels/classes when estimating the energy consumption of systems assembled from different components.

1. INTRODUCTION AND MOTIVATION

The basic idea of computing is that software utilizes hardware in order to solve well-defined problems and ease human life. This started by supporting the task of calculating trajectories of missiles and now covers nearly all aspects of human life. Almost all standard computers and even modern mobile devices like smart phones, laptops, or embedded systems are based on the well known Von-Neumann-Architecture [19]. Data and program code is stored in main memory. The central process unit (CPU), comprised of a control and arithmetic-logic unit, accesses the memory (i.e., a single separate storage structure) over a bus system. Nowadays it is common, that computers are connected to networks like the Internet. This enables information exchange and task distribution among devices with different in-built resources. Current trends in software development like .NET make use of this cloud of heterogeneous devices in order to solve more complex tasks than it would be possible with a single device. Clouds are either used to store data or to distribute workload. Thus, various hardware resources with different properties are utilized for data processing and storage. One essential difference between these resources is their energy consumption in relation to their performance. Large data centres already started to optimize their 'Power Usage Effectiveness' (PUE) that determines the energy efficiency and that is determined by dividing the amount of power entering a data centre by the power used to run the computer infrastructure within it. For single devices this step is to be done in the near future.

Especially for mobile devices that are typically battery-driven, energy consumption has to be optimized/reduced since the amount of energy consumed is correlated to the up-time of the device. Recent research efforts have shown that such resources can be substituted by each other to optimize one resource. Hence, it is possible to redesign a software system in such a way that the available resources are used in an energy efficient manner. In other words, one resource might be substituted by one or more other resources (e.g., exchanging memory consumption against processing power) in order to optimize the usage of the first resource. Unfortunately, there is no formal basis that classifies the 'fitness' of a chosen substitution strategy regarding energy efficiency.

This paper presents first ideas towards an energy-consumption based complexity-classification for resource-substitution-strategies. We therefore generalize our previous findings regarding the energy consumption of basic algorithms (i.e., sorting and basic DB algorithms such as joins) and the application of resource-substitution strategies. The goal is to define a formal classification scheme, similar to the big-O Notation that can be used to classify the 'energy-fitness' of strategies and algorithms. By having such a classification it will be possible to not only characterize the performance and memory usage but also the energy consumption of a system.

The remainder of this paper is structured as follows: Section 2 introduces the principle of resource substitution and covers related work. Section 3 focuses on energy consumption and closely investigates the actual energy consumption of the considered resources. Section 4 discusses our ideas on the energy complexity classification. Finally, section 5 summarizes the paper and gives an outlook on future research.

GvD Workshop'10, 25.-28.05.2010, Bad Helmstedt, Germany.

2. RESOURCE SUBSTITUTION — STRATEGIES & TECHNIQUES

In general, a resource is any physical or virtual entity of limited availability. Three main aspects can characterize resources: utility, quantity, and use. This is especially true for mobile and embedded systems that are characterized by a scarcity of resources. Such systems typically depend on the resources that are provided by the device the system is executed on, or the environment infrastructure. In detail, the following resources are in the main focus:

- The *Central Processing Unit* (CPU) is responsible for executing the instructions of a computer program. In principle each CPU consists of an arithmetic logic unit (ALU), a control unit, and registers (on-CPU-memory). There are various CPUs available on the market, e.g. XScale, SnapDragon, other processors based on the ARM architecture, Intel's x86 processors or the processors of the PPC family.
- The *memory* of a computing system stores data and program code. In general it can be divided into main memory (RAM), secondary memory (hard discs) and tertiary memory (DVD, tapes, etc.). Storing data on mobile devices differs from storing data on classic computers. Mobile devices typically use flash memory. Especially recent Smart-Phones or even some variants of Apple's 'MacBook Air' use this media as secondary memory. Nearly all current mobile devices allow the use of additional storage media in form of memory cards or Microdrives.
- While a bus system is responsible for in-computer communication, *networks* are used for inter-computer communication. The overall principle is that the sender prepares and addresses the data to send and transmits it electronically or optically via wire or through the air (wirelessly). Today, a device can have basically network access almost everywhere. Wireless local-area networks are hot-spotted networks with a reasonable speed. GSM-networks (and extensions like GPRS, EDGE or HSDPA) are slower but widely available. UMTS-infrastructure has been installed all over Europe but full coverage is far from existing.

Different techniques for handling data in mobile information systems have different resource consumptions characteristics (CPU or memory focused, etc.). Interestingly, it is possible to exchange or substitute resources by each other (i.e., resource substitution strategies) in order to optimize the use of one. Table 1 gives an general overview about possible substitutions that holds for all distributed systems, and thus also for client/server information systems with mobile clients. According to this table, the resource communication might be replaced by the resources CPU or memory (e.g., do computations and store data locally). Unfortunately, Table 1 does not include energy as a separate resource due to its orthogonal nature being affected by other resources. However, due to ever increasing functionality of mobile devices, limited battery capacities and environmental protection, energy consumption has to be addressed, too.

In the following we discuss in-depth the substitution possibilities with respect to energy consumption. Even if these ideas originally were examined in the context of mobile information system (mIS) [12], the findings can be generalized to all kinds of networked data management applications.

CPU vs. Communication: A high CPU load can be substituted by data communication. CPU-intensive calculations might be outsourced to a server [14] or distributed among many computers [1]. For example, semantic caching is a CPU-intensive approach. [10] presents an approach for migrating the task of finding reusable data in the cache to the server. In addition, [11] analyzes server site cache invalidation. Although, both approaches require complex software operations on the server, they drastically reduce client CPU load. On the other hand, we can easily reduce data transmission by using compression, which requires additional CPU usage.

CPU vs. Memory The substitution of CPU load by memory access is the well known as view materialization in the area of database systems. In general this represents the decision regarding the storage of temporary results. In order to save CPU cycles one can also use alternative access paths (indexes) to the data. Furthermore, standard compression approaches might be used for saving memory.

Memory vs. Communication Replication and caching can reduce the amount of communication. However, full replication reduces communication, whereas the efficiency of caching depends on the re-usability of cached data [6]. The benefit of caching not only depends on the memory but also on the CPU. In general one it can be stated that data, available on a device, must not be transmitted. But, redundant data might be outdated. Hence, synchronization mechanisms that connect to a server or receive updates via a broadcast channel are required. The substitution of the resource memory by communication is common for data centres where data is stored at central places. If a client needs the data accesses the server and reads the data. There are certain approaches following this substitution strategy (e.g., the Internet message access protocol (IMAP) allows to read email without prior download).

As discussed earlier, the idea of using resource substitution to optimize the use of a specific resource stems from the domain of mIS. Here, especially the resource memory is oversimplified. It is common to reduce the usage of secondary memory by caching data in primary memory. It is also common to swap data to secondary memory if the primary memory does not have any space left.

3. ENERGY CONSUMPTION REGARDING RESOURCES AND RESOURCE SUBSTITUTION

In this paper we consider the following resources: *CPU*, *primary memory (RAM)*, *secondary memory*, *network*, whereby the latter represents the communication aspects. As tertiary memory is mostly used for backup purposes, we do not consider it in the addressed context of this paper regarding the use of resource substitution for reducing the energy consumption of a system.

Energy consumption is usually correlated with hardware properties. However, there are strategic issues and findings by other researchers that examine and argue about software properties that have an impact on energy. Energy consumption associated with wireless data-transmission is not negligible [7]. CPU usage needs comparatively less energy than memory storage [16, 4]. Compressing a file by more then 10%, transmitting and decompressing it requires less energy than transmitting it uncompressed [18]. File

substitute	CPU	Communication	Memory
CPU	-	Migration of computations to the server	Materialization and re-usage of temporary results
Communication	Local execution of calculations	-	Local data storage
Memory	Data compression and compact data structures	Data management on the server only	-

Table 1: Substitutable resources [3]

compression also reduces the energy consumption of operations on hard disks [15].

CPU based energy consumption E_c strongly depends on the used processor architecture (includes also the basic differences of RISC or CISC based systems) and clock rate. For Intel desktop processors the energy consumption per instruction reaches from $10 \cdot 10^{-9}J$ for an i468 to $48 \cdot 10^{-9}J$ for an Pentium 4 (Cedarmill) processor [8]. Our own experiments based on the ATMega128L processor and the measurement environment described in [4] resulted in $1.058 \cdot 10^{-2}J$ for comparing two arrays of 1,000 integer values stored in an SRAM module (128K, 12 ns). By normalizing this value after removing the energy required for reading both arrays, we calculated that $5.33 \cdot 10^{-6}J$ are required for one comparison on this processor. The energy consumption E_p of *primary memory* operations can be divided into the energy that is required for reading E_p^r, for writing E_p^w and for keeping the state of this volatile memory E_p^s. As shown in [9] the following relation holds: $E_p^r < E_p^w$. The author calculated for one write access to SDRAM an energy consumption of $3 \cdot 10^{-17}J$ whereas one read access consumed $1.811 \cdot 10^{-17}J$. The results of our experiments with the ATMega128L processor are shown in Figure 1. Obviously, there is also a difference between accessing the on-chip memory and the external SRAM module.

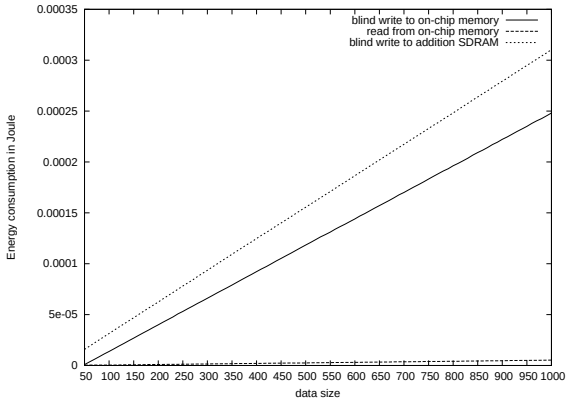

Figure 1: Reading and writing data from and to memory

Since the energy (consumption) is a function over time, it does not make sense to take E_p^s into account while thinking about resource substitution. Furthermore, we have to consider the CPU instructions required for reading and writing memory cells as well as the energy consumptions required for transferring data between CPU and memory. Therefore, we can extend the first relation to $E_c < E_p^r < E_p^w$.

Similar to primary memory, one can divide the energy consumption of *secondary memory* E_s into E_s^r, E_s^w, and E_s^s. As a rule of thumbs one can estimate the energy consumption for reading data from a hard disk as $n_{LBA}^3 J$. For one logical block this means 1 Joule, for two blocks 8 Joule, etc. [13]. Interestingly sequential writing of huge amounts of data requires 4-5 times less energy than reading the same amount of data as the read/write heads movements are reduced. However, even if the real values differ extremely for different devices [20] one can extend the previous relation to $E_c < E_p^r < E_p^w < E_s^r < E_s^w$ for single operations.

Finally, we have to consider network communication. It's energy consumption can be divided into the energy required for sending E_n^s and for receiving one Byte E_n^r. Especially wireless networks vary in their speed. Therefore, energy consumption for receiving a certain amount of data also depends on the time for the "download". Regarding [2] receiving 50KB with a 20-second interval requires $12.5J$ (UMTS), $5.0J$ (GSM), and $7.6J$ (WiFi). We have to note, that these values are pure transmission values. Sending data requires more energy. The authors of [17] calculated idle:receive:send ratios as 1:1.05:1.4. Obviously, more energy (from CPU and main memory) is required if packaging and addressing is taken into account. To summarize this, we get the following relation for energy consumption of the resources involved in resource substitution:

$$E_c < \underbrace{E_p^r < E_p^w}_{E_p} < \underbrace{E_s^r < E_s^w}_{E_s} < \underbrace{E_n^r < E_n^s}_{E_n}$$

CPU usage requires less energy than using primary and secondary memory, and network communication requires even more energy. The goal of this paper is to quantify the energy consumption of strategies and algorithms in distributed environments. One can specify the overall energy consumption of such an algorithm or strategy as the sum $E = E_c + E_p + E_s + E_n$.

4. ENERGY COMPLEXITY CLASSIFICATION

In general, a complexity class is a set of problems of related resource-based complexity and is defined by the set of problems that can be solved by an abstract machine M using $O(f(n))$ of resource R, where n is the size of the input. In the context of this paper we are interested in the usage of resources. This is typically covered by the big-O notation that describes the limiting behaviour of a function. It allows to simplify functions in order to concentrate on growth rates. Thus, big-O can be used to describe an algorithm's usage of computational resources (e.g., the worst case or average case running time or memory usage of an algorithm is of-

ten expressed as a function of the length of its input). This allows algorithm designers to predict the behaviour of algorithms and to choose the best fitting algorithm (regarding its complexity class), in a way that is (nearly) independent of the computer architecture or clock rate. Regarding the definition of complexity class for energy we have to consider the resources involved in the computation. Therefore, we have to distinguish and explicitly separate four subcomponents: $O_c(f_c(n))$ is the complexity for the CPU which is equivalent to the well known running time, since the energy consumption of a CPU is related to the number of instructions which, in turn depends on the size of the input. Similarly one can say that the complexity $O_p(f_p(n))$ corresponds to memory-usage complexity, since the energy consumption of primary memory depends on the number of read/write operations, which are derived or based on the input size. We have to point out that memory-usage complexity is *not* equivalent to the memory complexity known from theoretical computer science. The memory complexity describes the required amount of memory but does not include the memory accesses. Obviously, an algorithm could use only a constant amount of memory but read/write it several times. The same holds for the complexity $O_s(f_s(n))$ class characterizing the energy consumption of secondary memory. As the number of data transmissions is a function over the input size, too, $O_n(f_n(n))$ might express the energy complexity for the network resource. We learned from the previous chapter that energy consumptions of resources vary. Hence, we have to use the energy characteristics of the resources as scaling factors. For simplification we assume that E_c is the energy required for processing one input element, E_p and E_s for storing and retrieving one input element to/from main memory or secondary storage, respectively, and E_n for sending/receiving one input element via the network. Therefore, the energy complexity of an algorithms is given as sum:

$$O^E(f_e(n)) = O_c(E_c \cdot f_c(n)) + O_p(E_p \cdot f_p(n)) + O_s(E_s \cdot f_s(n)) + O_n(E_n \cdot f_n(n))$$

Finding the energy complexity (sub)functions. As mentioned earlier, the overall energy complexity class can be notated as the sum of the class functions of the involved resource. Obviously, it is possible to analyze the program code in order to find the respective (sub)functions. However, the running time complexity for most algorithms is known. Considering only CPU and RAM usage one can find the memory-access complexity by analyzing energy measurements. We described the measurement approach used in the following in [4]. Hence, $E_c \cdot f_c(n)$ is the function of the runtime complexity multiplied with the energy required for one operation. For example, for Mergesort $f_c(n) = n \cdot \log(n)$ holds. We measured the energy for one operation E_c (in this case for one value comparison) as well as the energy $E(n)$ required by the algorithm for various n. We know, that $E(n) = E_c \cdot f_c(n) + E_p \cdot f_p(n)$ holds. Hence, we can use model fitting techniques and tools like Eureqa[1] for finding $f_p(n)$.

Example: Given the running time complexity $O(n \cdot \log(n))$ of Mergesort, the energy of $5.33 \cdot 10^{-6} J$ consumed for one comparison and the normalized overall energy $E(n)$ ($E(10) = 0.00006042J \ldots E(1000) = 0.0100312J$ consumed by executing the algorithm for various n [5], we can find the memory-access complexity $f_p(n)$ by using the target function $E(n) = 5.33 \cdot 10^{-6} \cdot (n \cdot \log(n)) + f_p(n)$. To simplify the task for the model fitter we first calculated $E(n) - 5.33 \cdot 10^{-6} \cdot (n \cdot \log(n)) = f_p(n)$, copied the data to Eureqa, and searched for $f_p(n)$. The tool found various functions, e.g. $1.02553 \cdot 10^{-5} \cdot n - 5.1406 \cdot 10^{-5} \cdot \log(0.0033665 \cdot n - 0.0236113) - 0.000269206$. Almost all of them have shown an $n - \log(n)$ structure. As we know that each comparison requires reading the comparison partners from memory, the memory access complexity of our Mergesort implementation is $O(n \cdot \log(n) + n - \log(n))$, which equals $O((n-1) \cdot \log(n) + n)$.

[1] `http://ccsl.mae.cornell.edu/eureqa`

For analyzing the secondary storage complexity and the network complexity one could use monitoring tools for file systems and network traffic in combination with model fitting techniques. However, we did no experiments in this regard so far.

5. SUMMARY, CONCLUSIONS, AND OUTLOOK

Energy consumption is becoming a central issue for most user and manufacturers of computing devices and especially of mobile or embedded systems. Reasons are manifold and range from protecting the environment to extending the up- and operating-time of a system. However, the development and use of such, energy aware, systems, requires clear indication on the energy it consumes and, for the sake of easy comparison and selection, the distinction of energy classes. This becomes even more important, when it comes to software, since software has a major impact onto the energy consumed by a system. Within this paper we presented our ideas regarding the definition of such software classification approach that uses the 'classic' big-O notation as its metaphor. As a starting point we concentrated on resource substitution strategies/algorithms since those have shown their potential in making a system energy-aware [3]. We introduced different strategies and discussed their energy-consumption related characteristics. Based on that we introduced and applied a classification scheme for energy consumption. The scheme provides valuable information about the fitness of specific algorithms and strategies regarding the optimization of energy consumption of a system. However, these ideas have to be applied in practice (e.g., defining the energy class of different algorithms, etc.). The next step then is to come up with techniques for determining the energy class of assembled components and or systems. Furthermore, we plan to define techniques and tools for the development of energy-aware software systems that make use of energy classes. Finally, we are currently discussing a realistic case study to demonstrate the feasibility and applicability of our ideas in practice.

6. REFERENCES

[1] D. P. Anderson, E. Korpela, and R. Walton. High-Performance Task Distribution for Volunteer Computing. In *Proceedings of the First International Conference on e-Science and Grid Computing*, pages 196–203, Washington, DC, USA, 2005. IEEE Computer Society. available online: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.840&rep=rep1&type=pdf`.

[2] 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*, pages 280–293, New York, NY, USA, Nov. 2009. ACM. available

online: http://www.cs.umass.edu/~arunab/paper/tailender-imc09.pdf.

[3] E. Buchmann. Konzeption und Implementierung eines Speichermanagers für ein konfigurierbares, leichtgewichtiges DBMS. Diplomarbeit, FIN, Otto-von-Guericke Universität Magdeburg, Magdeburg, Germany, Feb. 2002. in German, supervised by Hagen Höpfner.

[4] C. Bunse, H. Höpfner, E. Mansour, and S. Roychoudhury. Exploring the Energy Consumption of Data Sorting Algorithms in Embedded and Mobile Environments. In *Proceedings of the 10th International Conference on Mobile Data Management: Systems, Services and Middleware (MDM 2009), 18-20 May 2009 Taipei, Taiwan*, pages 600–607, Los Alamitos, CA, USA, 2009. IEEE Computer Society. ISBN: 978-0-7695-3650-7, http://doi.ieeecomputersociety.org/10.1109/MDM.2009.103.

[5] C. Bunse, H. Höpfner, S. Roychoudhury, and E. Mansour. Choosing the "best" Sorting Algorithm for Optimal Energy Consumption. In B. Shishkov, J. Cordeiro, and A. K. Ranchordas, editors, *Proceedings of the 4th International Conference on Software and Data Technologie (ICSOFT 2009), July 26-29, 2009, Sofia, Bulgaria*, volume 2, pages 199–206, Setúbal, Portugal, July 2009. INSTICC, INSTICC press. ISBN: 978-989-674-010-8, available online: http://www.hoepfner.ws/images/papers/icsoft09.pdf.

[6] A. Caracaş, I. Ion, M. Ion, and H. Höpfner. Towards Java-based Data Caching for Mobile Information System Clients. In B. König-Ries, F. Lehner, R. Malaka, and C. Türker, editors, *Proceedings of the 2nd conference of GI-Fachgruppe MMS*, volume P-104 of *LNI*, pages 97–101, Bonn, Germany, 2007. GI, Köllen Druck+Verlag GmbH. available online: http://subs.emis.de/LNI/Proceedings/Proceedings104/gi-proc-104-009.pdf.

[7] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proceedings IEEE INFOCOM 2001, The Conference on Computer Communications, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Twenty years into the communications odyssey, 22-26 April 2001, Anchorage, Alaska, USA*, volume 3, pages 1548–1557, Los Alamitos, CA, USA, 2001. IEEE. available online: http://www.sics.se/~lmfeeney/publications/Files/infocom01investigating.pdf.

[8] E. Grochowski and M. Annavaram. Energy per instruction trends in intel®microprocessors. Technical report, Microarchitecture Research Lab, Intel Corporation, Santa Clara, CA, USA, Mar. 2006. available online: http://support.intel.co.jp/pressroom/kits/core2duo/pdf/epi-trends-final2.pdf.

[9] T. Hirth. Energiecharakterisierung rekonfigurierbarer Rechensysteme. Diplomarbeit, Friedrich-Alexander-Universität Erlangen-Nürnberg, Institut für Informatik, 2006. in German. available online: http://www4.informatik.uni-erlangen.de/DA/pdf/DA-I4-2006-06-Hirth.pdf.

[10] H. Höpfner. Serverseitige Auswertung von Indexen semantischer, clientseitiger Caches in mobilen Informationssystemen. In P. Dadam and M. Reichert, editors, *INFORMATIK 2004, Band 1*, volume P-50 of *Lecture Notes in Informatics (LNI)*, pages 298–302, Bonn, Germany, Sept. 2004. GI, Köllen Druck+Verlag GmbH. in German, available online: http://cs.zblmath.fiz-karlsruhe.de/LNI/Proceedings/Proceedings50/GI-Proceedings.50-64.pdf.

[11] H. Höpfner. Update Relevance under the Multiset Semantics of RDBMS. In T. Kirste, B. König-Ries, K. Pousttchi, and K. Turowski, editors, *Mobile Informationssysteme*, volume P-76 of *LNI*, pages 33–44, Bonn, Germany, 2006. GI, Köllen Druck+Verlag GmbH. available online: http://subs.emis.de/LNI/Proceedings/Proceedings76/GI-Proceedings-76-3.pdf.

[12] H. Höpfner and C. Bunse. Ressource Substitution for the Realization of Mobile Information Systems. In J. Filipe, M. Helfert, and B. Shishkov, editors, *Proceedings of the 2nd International Conference on Software and Data Technologie (ICSOFT 2007), July 22-25, 2007, Barcelona, Spain*, volume Software Engineering, pages 283–289, Setúbal, Portugal, July 2007. INSTICC, INSTICC press. available online: http://www.hoepfner.ws/images/papers/icsoft07.pdf.

[13] A. Hylick, R. Sohan, A. Rice, and B. Jones. An Analysis of Hard Drive Energy Consumption. In E. L. Miller and C. L. Williamson, editors, *Proceedings of the 16th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2008)*, pages 103–112, Los Alamitos, CA, USA, Sept. 2008. IEEE Computer Society. available online: http://www.cl.cam.ac.uk/~acr31/pubs/hylick-harddrive2.pdf.

[14] I. Ion, A. Caracaş, and H. Höpfner. MTrainSchedule: Combining Web Services and Data Caching on Mobile Devices. *Datenbank-Spektrum*, 21:51–53, May 2007. available online: http://www.datenbank-spektrum.de/pdf/dbs-21-51.pdf.

[15] A. Kansal and F. Zhao. Fine-grained energy profiling for power-aware application design. *ACM SIGMETRICS Performance Evaluation Review*, 36(2):26–31, Sept. 2008. available online: http://research.microsoft.com/~zhao/pubs/hotmetrics08joulemeter.pdf.

[16] P. Marwedel. *Embedded System Design*. Springer, 2007.

[17] M. Stemm and R. H. Katz. Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices. *IEICE Transactions on Communications*, E80-B(8):1125–1131, July 1997.

[18] J. Veijalainen, E. Ojanen, M. A. Haq, V.-P. Vahteala, and M. Matsumoto. Energy Consumption Tradeoffs for Compressed Wireless Data at a Mobile Terminal. *IEICE Transactions on Communications*, E87-B(5):1123–1130, May 2004.

[19] J. von Neumann. First Draft of a Report on the EDVAC. *IEEE Annals of the History of Computing*, 15(4):27–75, Oct. 1993. http://dx.doi.org/10.1109/85.238389.

[20] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Wang. Modeling Hard-Disk Power Consumption. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pages 217–230, Berkeley, CA, USA, 2003. USENIX Association. available online: http://www.cs.princeton.edu/~rywang/papers/fast03/dempsey.pdf.