Identification of Application-level Energy Optimizations

Kay Grosskop[1], Joost Visser[1]

[1]Software Improvement Group (SIG), Amstelplein 1 1096 HA Amsterdam, Netherlands
k.grosskop@sig.eu, j.visser@sig.eu

ABSTRACT

Advances in the energy-efficiency of hardware and datacenters may be in vain if application software is developed without awareness of the resources that are consumed in operation. We introduce a pragmatic method for identifying energy-efficiency optimizations in software applications. The method involves the creation of an energy model of an application that allows calculation of energy savings for optimization scenarios. We discuss two industrial cases where such *Green Software Scans* were applied to an e-government and a mobile banking application. Among the lessons drawn from these cases are that significant energy savings can be realized with targeted modifications in software code, architecture, or configuration, that development of energy-efficient applications requires attention to detail throughout the development process, and that a close collaboration between operations and development is one of the main success factors for energy-efficient applications.

Keywords

Software engineering, simulation and modeling, application software, green and sustainable computing, energy-efficiency.

1. INTRODUCTION

Wirth's Law [1] was coined in the mid 1990's to express the unease with a long running trend in system development:

"Software is getting slower more rapidly than hardware becomes faster"

The exponential growth of hardware capacities had made software become ever more unconscious about its resource consumption. Up to the point that often gains in hardware performance where cancelled out by software resource greediness and from the perspective of the end user the performance for a given functionality decreased with a next release.

Arguably today the same is true for the energy consumed by computers. The fact that in socket-powered devices energy was available in abundance caused that considerations of its efficient use never entered the horizon of programmers' awareness. As a consequence, today's software systems use far more energy then they have to and most system engineers are unaware of the energy efficiency of their products.

That is about to change. Of course there have already been exceptions to the rule in the area of mobile computing, where batteries put constraints on the energy available. But it is only recently that this platform is getting a dominant position. A more important factor is probably the fact that power provisioning and heat dissipation in datacenters get problematic from a technical, financial and environmental point of view. While initially optimization efforts where mostly directed at data-center infrastructure and hardware levels, the awareness is slowly growing that also software plays a crucial role.

Addressing the problem at the software level proves to be hard for various reasons of social and technical nature. Energy is spent in the hardware at the data center, but engineers responsible for the design of the software are often working in a different organizational unit. Application owners are billed by server or floor space, but generally not directly for the energy consumed. Hence the energy costs are not part of the total cost of ownership of applications and pose no incentive for change. Even when there are economic or environmental concerns, it is not trivial to actually determine the energy footprint of the application. There is a lack of knowledge, conceptual frameworks and tools to measure or model the energy consumed.

Visibility of energy consumption at the application level is essential because this is the level of architectural design and of management decisions that have the most influence on the software running in data centers. For this reason optimization efforts on this level have the best chance to be effective.

In this article we present the *Green Software Scan*, an approach to effectively model software application energy consumption and to arrive at recommendations for optimization. This approach is developed in order to be executable in a very limited timeframe on an existing IT system, taking into account limitations of organizational and technical nature. It is applicable to a wide spectrum of existing systems that account for a large part of today's IT electricity consumption. We report on two case studies that we have executed for clients in the industry and that showed that this approach is viable and leads to the desired results

The paper is structured as follows. After the introduction, Section 2 discusses background and related work. In this section we shortly characterize other attempts related to software energy measuring and modeling. Section 3 presents our method of identifying application-level energy-optimization opportunities. We describe how an energy-model of the application is build up and how it is used to evaluate optimization scenarios. Section 4 describes two case studies where the approach was used and what we learned from it. Section 5 concludes with a discussion of scoping and design decision regarding our method and we provide suggestions for further work

2. BACKGROUND

There is a considerable amount of literature on the field of energy optimization at the software level. This is especially the case in areas where energy shortage was an issue like mobile or

ICT4S 2013: Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, ETH Zurich, February 14-16, 2013. Edited by Lorenz M. Hilty, Bernard Aebischer, Göran Andersson and Wolfgang Lohmann.
DOI: http://dx.doi.org/10.3929/ethz-a-007337628

embedded architectures or wireless sensor networks. But only a limited number of approaches target the application level and most are focused on a specific platform and application domain. To our knowledge, there exists no other work that attempts the modeling and optimization of energy usage at the application level and that is applicable to a wide range of existing software systems.

Williams and Smith [12] present PASA, a pragmatic approach for performance optimization that is targeted at the software architectural level. The idea is to identify key scenario's and develop the most simple architectural models for these scenarios that can illustrate the performance problems of the system. The models, once sufficiently refined, are then used to evaluate proposed changes to the system. The approach is very similar in methodology and spirit to the approach presented here. However, energy optimization needs to look at the *total* energy consumption of the system and *idle* states of the system, in terms of not executing a certain scenario, are particularly interesting. For this reason the use of key scenarios for the detection of bottlenecks is not generally useful. Instead our models are organized along the concept of a *node map*.

There have been recently developed several fine-grained power measurement methods and tools for software. Examples are Kansal et al [3], Do et al [2]. Most of them are based on the idea of analyzing resource counters provided by the operating system and calculate energy usage with a power model of the machine hardware components. The performance counters can be related to specific processes, which allow separating energy usage of single processes in a shared environment. Those approaches either are directed at supporting real time resource management algorithms or profiling of applications for development and aim at processes on single machines. At the contrary our approach specifically supports distributed and networked applications because this are rather standard characteristics of todays software programs. Also our method does not require installation of specific software or even software or hardware instrumentation. This greatly improves its applicability for systems deployed for production or even components used by the system that belong to other organizations.

Zhao et al. [4] expand this approach of performance counter based metrics to estimation of energy consumption for virtual environments. While still requiring platform specific software installation and not leveraging a view on the application as a whole, this approach is interesting in the light of the current trends to virtualization and cloud computing and the resulting isolation of server hardware energy consumption and their virtualized logical view.

Seo et al. [11] have proposed a framework to model the energy consumption of software architectures in the design phase based on principal energy characteristics of the underlying architectural pattern. They explicitly target distributed systems. Even though the framework can be parameterized with platform and application specific data it is designed to support decisions between specific architectural styles. For assessment and optimization of existing systems the characteristics of the architectural pattern must be extracted and the associated energy costs must be formalized. This prevents direct application of the framework in many cases and offers only limited flexibility in identifying and optimizing energy hotspots in existing systems.

The CO2Stats [16] and Greenalytics [15] initiatives offer online estimation of the energy consumed by websites and make also suggestions for optimization. They developed a pragmatic model for application-centric energy costs on the web-client and network side. While also partly including the server side into their models and advertising an 'end to end' estimation of the energy costs, the models are mostly based on data size and request statistics for web content and model back-end system resources only in a trivial way. Real consumptions of those systems as well as architectural trade offs remain opaque.

Some authors, notably Barroso and Hölzle [5], Ranganathan [13] and Saxe [14] have made attempts to describe some general principles for energy efficiency that are important to understand and can be leveraged when looking for optimizing alternatives for existing system architectures.

An interesting example for application-level analysis and optimization of energy consumption is presented in [8] and [9]. Harizopulos, Sha, Meza and Ranganathan [9] argue for the importance of software-level energy tuning for data management applications and identify possible strategies. Tsirogiannis, Harizopoulos and Shah [8] however have investigated the power consumption of databases for scale-out architectures focusing on a single server. They conclude that for this specific scenario a system optimized for performance will also be optimized for energy efficiency. These papers illustrate the non-trivial relationship between performance and energy efficiency optimization.

Koomey et al. have published a series of widely sited reports on estimation of energy consumption of IT infrastructure. Particularly interesting in this context is one report [7] that states that in-house facilities still dominate the installed base of server population. This is a class that is typically less optimized for energy and where our approach can add much value. In [6] an estimation of the energy intensity of Internet network infrastructure is made. This is an interesting undertaking of estimation in itself and the results have been used in our case studies as well as in [15].

The GreenGrid DCeP metric [17] and Intel's EnergyChecker [18] represent two initiatives that also attempt to express the energy efficiency of computing systems in terms of functional work units that are delivered for a given energy consumption. While the considerations and frameworks are conceptually interesting, We doubt that they will be applicable in many cases because [17] ends up to use a non-functional proxy for its estimation of productivity and [18] requires extensive instrumentation of the application code.

3. METHOD

In order to estimate the energy footprint of a software system and evaluate alternative design choices and optimization scenarios, we construct an *energy model* of the application. Such a model allows understanding various energy aspects of the application and shows where 'energy hotspots' are located.

An overview of the evaluation process is shown in Figure 1. Initially interviews are conducted with system designers and operators to get an architectural understanding of the application. Also the available insight into load/usage and energy consumption characteristics of the components is gathered. Then the energy model of the application is constructed and alternative scenarios for improvement are evaluated using this model. Finally recommendations are made to the owner of the system.

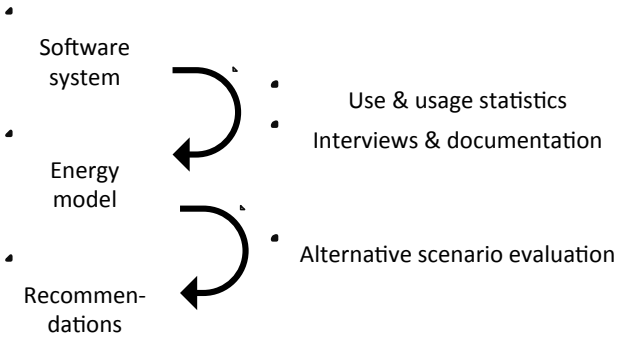

Figure 1. Overview of the process of a green software scan

3.1 Building the energy model

Because we want to reduce the required effort and get useful insights as fast as possible it is crucial to start modeling at an early stage. This allows identifying directly where additional data gathering is needed in order to make the estimation sufficiently accurate.

The principal ingredients of the model are

1. Developing a *node map* that identifies all hardware components or nodes that are used by the application.
2. Determine the main 'units of work' of the application.
3. Collect data on the energy usage of all components in relation to the amount of work done by the system

The basic idea of the energy model is to break up the system into subcomponents in order to be able to localize in more detail where the bulk of the energy is spent and to analyze the effect of the system architecture on the energy spending. The result is called a *node map*.

The node map can be close to other views on the system architecture like the deployment view or the software component architecture but may also be different depending on what are the interesting elements from an energy point of view.

What a node is depends on the system being evaluated. The goal is to cover all parts of the IT infrastructure that an application needs for its functioning. Nodes are typically socked-powered servers. But nodes can also be parts of the infrastructure like the company network, 'the Internet' or wireless devices combined with their battery chargers. Also a remote service (e.g. a web service) can be considered a node, since from the viewpoint of the application it constitutes a single resource.

After identifying all resources needed for the functioning of the application and mapping them to nodes in the model, energy costs and load data are attributed to the nodes. The strategies to model the energy behavior of different types of nodes are covered in the next subsection.

While constructing the node map, it may turn out that some information is lacking. Sometimes a part of the infrastructure must be further clarified or additional data must be collected.

After attributing the energy consumed to the nodes, the map becomes a high-level *heat map* of the application

3.2 Interviews

The scan starts with one or more interviews of system architects or operational staff in order to get an initial understanding of the architecture of the system, its physical deployment and the data flows.

We also need to get some insight into hardware characteristics of the infrastructure the system is running on, if and to what degree this infrastructure is shared with other applications and what is the amount of workload that has to be processed.

Another important subject is the identification of principal usage scenarios of the software and the principal *unit of work* that allows expressing the energy use in functional terms.

Many of the questions would likely also be posed during a typical performance assessment as described in [12].

Examples of questions:

- Are parts of hardware virtualized?
- Which remote services are consumed?
- What are the most common use cases?
- What are the numbers of session or transactions?
- What is known about energy consumed by the servers?
- Which other applications share the platform?

In addition to the interviews and depending on the availability it is often useful to extract some knowledge about the system from architectural documentation and analysis of the source code.

3.3 Data collection and estimation

Two types of data are interesting for modeling the energy behavior of the application: *work load* and *energy consumption*. The need for information on energy consumption is obvious since we want to characterize the energy footprint of the system and detect the biggest consumer among its subcomponents.

An example of information collected for each node in the node map is given in Figure 2.

Network traffic/week	14.7 GB
Energy intensity of network	0.6 kWh/GB
Number of user sessions/week	5200
Calculated energy per session	0.0017 kWh

Figure 1: Energy estimation for the node: Internet communication between client and server

The need for load data is less obvious and has two reasons: First, from a total system point of view it can be used to express the energy efficiency of the system in terms of a *functional unit of work*. Second, it is needed for a characterization of the *energy scaling behavior* of the application and its parts. This expresses how the energy consumption and hence the efficiency changes according to different load levels. As prominently stated in Barroso and Hölzle [5] the *energy proportionality* of systems is a crucial property of energy efficient system because it allows the system to scale down its resource use when underutilization occurs. So this means that especially on components with large consumption and substantial underutilization it is interesting to model the scaling behavior more closely.

Load typically varies strongly over time and hence another aspect of data gathering is the choice of a representative time range. That may be a day or months depending on the application.

Our approach is suited to cope with the fact that fine-grained energy and load data is not immediately available. It is often sufficient to use motivated estimations or reference values in the model. Where exact power consumption metrics are not available, they can be deduced by hardware specifications or replaced by reference metrics of comparable hardware. When the hardware configuration is not known, typical hardware specifications can be used instead. It is dependent on the system at hand which strategy of estimation is chosen and when the results will be 'good enough' to obtain a useful result.

If actual measured data become available at a later stage, the model can be updated to become more accurate.

Sometimes, it is however crucial to have some exact measurement data because the error of the estimations would have too much influence on the outcome. For a variety of reasons it is often difficult to obtain additional measurements during the scan. For this reason the requests for such data from the developers and operators of the system must be reduced to a minimum and must be well chosen.

The selection of more data points can be guided by questions as:

- What are big spenders in the model?
- What can be estimated without a big error margin?
- Is the missing part of the model economically interesting or otherwise relevant for the owner of the application?
- Is it technically and organizationally feasible to collect more detailed information?

3.4 Scenario-based recommendations

The model is not only used to identify energy drains in the overall system infrastructure, but also to evaluate scenarios of changes to the system. The goal is to identify recommendable alternatives to the systems architecture or implementation. Typically during the model construction process some interesting hypotheses have surfaced that can be verified with the model. This might for example be a replacement of some hardware, some virtualization or a software architecture change like reducing data structures, traffic frequency or protocols for certain communications. These hypotheses may be formed by comparing the application setup with known best practices or be triggered by initial results of the estimation itself that located energy 'hot spots' in the application architecture. At this point it might also be necessary to go back and refine the model for some specific scenario evaluation.

4. CASE STUDIES

In this section we describe two cases where the evaluation method has been applied. These concern an e-Government application and a mobile banking application.

4.1 Case study: e-Government application

4.1.1 Functional characterization of the application

The application that we analyzed supports inserting requests for permissions to be granted by the municipality. The interface is publicly available on the Internet, but was also used by call center employees in the back offices of municipalities.

4.1.2 Technical Characterization of the application

The system is build using a Microsoft application stack and is deployed separately for every municipality. Most parts of the servers are virtualized and hosted on a data-center by a contracted hosting provider. Several external services are consumed. Data is stored on a centralized storage area network (SAN). The middleware is partly also used by another application. The node map that we constructed for the application is shown in Figure 2.

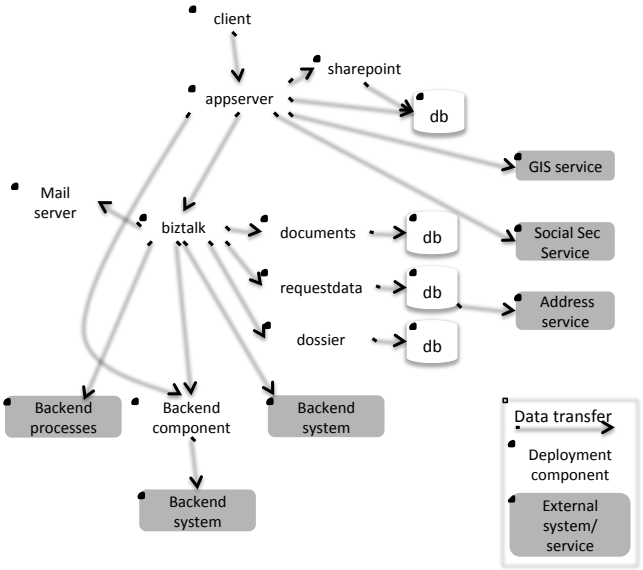

Figure 2: Node map of an e-Government application

4.1.3 Description of particularities of the case

After making an inventory of all servers, services and connections between them as well as some basic usage data, we observed that most of the energy was consumed in the middleware of the back end, but for many municipalities the installations where strongly underutilized. In addition to that, the middleware components where not able to scale down energy use in low utilization. Hence we investigated the possibility to run a single instance of the application for multiple municipalities. The organizational restrictions allowed for such a scenario and the required changes to the software design or implementation were assessed to be feasible. The outcome of an estimation showed that 82% of the energy could be saved in a shared deployment scenario compared to the current dedicated installs.

Another scenario concerned the archiving of data. Even though a request made by a citizen was normally handled and finalized within months, the request and associated documents had to be conserved for years for legal reasons. We calculated how much energy it would save to archive data into a less energy intensive storage system. But it turned out, that the moderate amount of data and the relatively good efficiency of the storage solution did not support this scenario to contribute in a significant way to a better energy efficiency.

4.2 Case study: mobile banking application

4.2.1 Functional characterization of the application

The application allows users to perform mobile payments, and do some administration concerning payment both on a mobile app and a normal browser. During the assessment the application was still in a beta phase and had only a small number of users.

4.2.2 Technical Characterization of the application

The application consists of an iPhone client, a generic mobile client and a normal website for conventional browsers. The mobile interfaces are served by another provider than the conventional website and business process request where forwarded from this mobile proxy to the back-end. This proxy infrastructure was shared with many other applications. Various communication data formats (proprietary binary, soap, html) and

runtime environments (iOS, JEE, .Net) where used. The node map that we constructed for the application is shown in Figure 3.

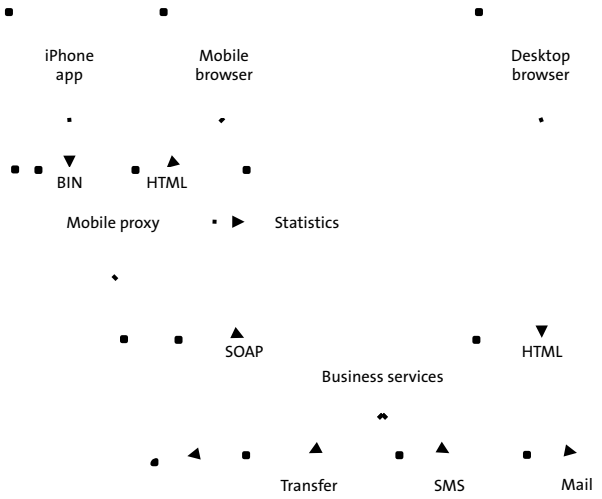

Figure 3: Node map of a mobile banking application.

4.2.3 Description of particularities of the case

Strategies for energy attribution of the different nodes of the system included the use of reference numbers (based on Koomey [7]) for estimation of energy intensity of data traffic over the Internet and the mobile network and attribution of the energy share of the mobile proxy infrastructure based on server log statistics. For the mobile devices, we added the charger loss to the estimated energy drawn from the battery while the application was in use.

The model showed that most energy was consumed at the servers and the various parts of the network. We made two recommendations for changes in the system based on modeling the energy gains. The first was to remove the intermediate proxy server and to connect all clients directly to the same backend. The other recommendation was to reduce the amount of data exchanged with the mobile devices in various ways. Those optimizations where estimated to reduce the energy footprint of the system by respectively 30% and 15%.

4.3 Lessons learned

Apart from the findings and recommendations specific to the applications under evaluation, a number of lessons can be learned from the case studies. We briefly list a number of such lessons.

Energy-savings through application-level optimizations can lead to substantial savings. In the case studies, potential savings of 30% and 80% were identified.

In the initial stages of the evaluations, we discovered that most stakeholders had a fairly low level of awareness that software is actually an important factor in Green IT. But when made aware, the importance of software for energy-efficiency was considered non-controversial. This phenomenon was also found in a wider population in a previous survey regarding green software [19].

We also found that carrying out a Green Software Scan is an effective means of raising awareness, not only for optimizing the system being scanned, but also stimulating better practices in subsequent projects involving other applications.

Application of the scan requires combining information from stakeholders that are often part of different organizations or of different organizational units that do not communicate intensively. On the other hand, effective communication and collaboration between development and operations seems to be an important success factor for energy-efficient software.

The measurement plan employed in an evaluation does not necessarily need to be very detailed or precise to obtain usable optimization results. On a non-optimized system limited measurement possibilities can largely be compensated with estimation.

Attention to detail is important to develop energy-efficient applications. A small change, for instance the resolution of a picture that is transmitted regularly, may have a large influence that is not immediately noticed.

An approach to energy-efficient application development is needed that considers the entire lifecycle. Early attention to energy characteristics, e.g. in the requirements and design stage, may have a better ROI than trying to optimize an application when it has already been built and deployed.

Low-level algorithmic optimizations are often not the most promising source of optimization. Rather, changes in architecture, design, or deployment tend to have a larger impact.

When considering non-multimedia data communication, the energy costs of processing at end points (e.g. xml serialization) are typically much higher than the network traffic itself.

Though performance and energy-efficiency might in theory be interacting characteristics, in the cases at hand we discovered that they actually do not interfere.

By reporting on energy consumption in terms of energy consumed per useful unit of work, we link energy consumption to the functional perspective that all stakeholders share. This facilitates a common discussion where all people involved can align their objectives.

5. CONCLUSIONS

We list contributions, strengths, limitations, and avenues for future work.

5.1 Contributions

In this paper, we have made the following main contributions:

- We have described a pragmatic method to estimate the energy footprint of a software application and identify optimizations at the application level.
- Our method includes a general framework for constructing energy models, based on a node map that represents the structure of the application and its supporting infrastructure and onto which information on work load and energy consumption is projected.
- We have demonstrated the applicability of our method in two case studies.
- We have formulated lessons regarding green software, ranging from specific savings potentials to general observations on technical and organizational factors.

5.2 Strengths

Our method has a number of properties that we deem desirable for any valuable energy analysis of software systems:

- The view on the system is *holistic* in the sense that it includes all energy consumed by IT components on behalf of the system.
- Functional perspective. The results are expressed by relating energy to a *useful unit of work.* This is the perspective of the end user and facilitates discussion between different stakeholders. It also makes it easier to compare systems among each other.
- Application centric. Being able to get more visibility of the energy used at the application level is essential because that is the level of architectural design and management decisions. And hence optimization efforts will be most effective on this level.
- Wide applicability. The scan can work for very different classes of software. In particular, it is applicable to distributed applications, which represent a big part of today's installed systems.

5.3 Limitations

The proposed method has a number of limitations. For example, it is focused on energy, rather than CO_2, on energy consumed in operation, rather than development or entire lifecycle, and on direct rather than rebound effects. These and other limitations are listed below.

- We have chosen to confine ourselves to electricity consumption because that can be regarded as a scarce resource by itself. See e.g. [15] for a region-dependent calculation of e.g. *CO_2 emission equivalents* on basis of electricity used.
- A more comprehensive view on the impact of an IT system would be an analysis of its total life cycle (*LCA*). This includes e.g. the build phase of software or the required hardware. We chose to concentrate on the consumption in use. The results can later be leveraged in a total LCA.
- In a similar way we left out effects on the surrounding business process and systemic or *rebound effects* that an IT system can have. Again, an assessment of the direct IT energy use constitutes a necessary part of a broader analysis.
- We where interested in consumption of the IT infrastructure. This is also called 'greening *of* IT' as opposed to 'greening *by* IT'. Using it to green other processes may have even a bigger effect on global energy consumption but does not undermine the usefulness of the former.
- Energy use is only calculated as far as it is consumed by IT infrastructure. This includes the data center infrastructure and mobile chargers, but not losses in the electric grid.

5.4 Future work

We regard the method described in this paper as a modest starting point that needs to be refined and complemented in many ways.

- It would be interesting to apply the method in case studies where the project is still in the initial architectural design phase and no software system has yet been built or deployed.
- To enable energy-efficient software development, remote services used in by an application could be labeled with their energy footprint as a part of Quality of Service (QoS) information.
- Accessibility and visualization of energy-consumption data to developers and other stakeholders is crucial for improvement on the long term. It would be interesting to conduct case studies where such information feedback is established.

The construction of reliable and precise energy models of applications could be supported with energy benchmarks of commonly used building blocks such as programming languages, operating systems, database management systems, run-time environments, etc

6. ACKNOWLEDGMENTS

The work presented in this paper was partially supported by Agentschap NL.

7. REFERENCES

[1] Niklaus Wirth. 1995. A Plea for Lean Software. *Computer* 28, 2 (February 1995), 64-68. DOI=10.1109/2.348001 http://dx.doi.org/10.1109/2.348001

[2] T. Do, S. Rawshdeh, and W. Shi, "ptop: A process- level power profiling tool," in Workshop on Power Aware Computing and Systems (HotPower '09), 2009.

[3] Aman Kansal and Feng Zhao. 2008. Fine-grained energy profiling for power-aware application design. *SIGMETRICS Perform. Eval. Rev.* 36, 2 (August 2008), 26-31. DOI=10.1145/1453175.1453180 http://doi.acm.org/10.1145/1453175.1453180

[4] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka A. Bhattacharya. 2010. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM symposium on Cloud computing* (SoCC '10). ACM, New York, NY, USA, 39-50. DOI=10.1145/1807128.1807136 http://doi.acm.org/10.1145/1807128.1807136

[5] Luiz André Barroso and Urs Hölzle. 2007. The Case for Energy-Proportional Computing. *Computer* 40, 12 (December 2007), 33-37. DOI=10.1109/MC.2007.443 http://dx.doi.org/10.1109/MC.2007.443

[6] Weber, C. L., Koomey, J. G. and Matthews, H. S. (2010), The Energy and Climate Change Implications of Different Music Delivery Methods. Journal of Industrial Ecology, 14: 754–769. doi: 10.1111/j.1530-9290.2010.00269.x

[7] J. G. Koomey. GROWTH IN DATA CENTER ELECTRICITY USE 2005 TO 2010. Technical report, Analytics Press, 2011

[8] Dimitris Tsirogiannis, Stavros Harizopoulos, and Mehul A. Shah. 2010. Analyzing the energy efficiency of a database server. In *Proceedings of the 2010 international conference on Management of data* (SIGMOD '10). ACM, New York, NY, USA, 231-242. DOI=10.1145/1807167.1807194 http://doi.acm.org/10.1145/1807167.1807194

[9] S. Harizopoulos, M. Shah, J. Meza, and P. Ranganathan. Energy efficiency: The new holy grail of data management systems research. In CIDR, pages 4--7, 2009.

[10] Hui Chen, Shinan Wang, Weisong Shi.. Where does the power go in a computer system: Experimental analysis and implications. In *Green Computing Conference and*

Workshops (IGCC), 2011.
DOI=10.1109/IGCC.2011.6008598

[11] Chiyoung Seo, George Edwards, Daniel Popescu, Sam Malek, and Nenad Medvidovic. 2009. A framework for estimating the energy consumption induced by a distributed system's architectural style. In *Proceedings of the 8th international workshop on Specification and verification of component-based systems* (SAVCBS '09). ACM, New York, NY, USA, 27-34. DOI=10.1145/1596486.1596493 http://doi.acm.org/10.1145/1596486.1596493

[12] Lloyd G. Williams and Connie U. Smith. 2002. PASA^{SM}: a method for the performance assessment of software architectures. In *Proceedings of the 3rd international workshop on Software and performance* (WOSP '02). ACM, New York, NY, USA, 179-189. DOI=10.1145/584369.584397 http://doi.acm.org/10.1145/584369.584397

[13] Parthasarathy Ranganathan. 2010. Recipe for efficiency: principles of power-aware computing. *Commun. ACM* 53, 4 (April 2010), 60-67. DOI=10.1145/1721654.1721673 http://doi.acm.org/10.1145/1721654.1721673

[14] Eric Saxe. 2010. Power-Efficient Software. *Queue* 8, 1, Pages 10 (January 2010), 8 pages. DOI=10.1145/1698223.1698225 http://doi.acm.org/10.1145/1698223.1698225

[15] J.L. Zapico, M. Turpeinen, N. Brandt. Greenalytics: a tool for mash-up life cycle assessment of websites. In: *Proceedings of the 24th International Conference on Informatics for Environmental Protection* (EnviroInfo 2010). Cologne/Bonn, Germany, Shaker, Aachen, 2010, pp. 754–763. ISBN: 978-3-8322-9458-8. http://www.greenalytics.org

[16] CO2Stats. http://www.co2stats.com

[17] D.Anderson, et al. *A Framework for Data Center Energy Productivity*. 2008.White paper #13: Metrics & Measurements. Retrieved February 2012 from http://www.thegreengrid.org/Global/Content/white-papers/Framework-for-Data-Center-Energy-Productivity

[18] Intel's EnergyChecker SDK. http://software.intel.com/en-us/articles/intel-energy-checker-sdk/

[19] Green IT consortium region Amsteram and Software Improvement Group, Green Software Awareness Survey, 2009. http://www.sig.eu/blobs/Nieuws/2011/Results Survey-201109.pdf