Advances in Energy Aware Computing: Lessons Learned from the EN-ACT Project

Luis Corral[1] Anton B. Georgiev[1] Boban Celebic[2] Ruth Breu[2]
Falko Dressler[3] Azamat Orazov[2] Klaus Kofler[2] Thomas Fahringer[2]
[1] Free University of Bolzano, Italy
[2] Institute of Computer Science, University of Innsbruck, Austria
[3] Institute of Computer Science, University of Paderborn, Germany
{luis.corral, anton.georgiev}@unibz.it
{boban.celebic, ruth.breu}@uibk.ac.at
dressler@ccs-labs.org, a_orazov@bk.ru
{klaus.kofler, thomas.fahringer}@uibk.ac.at

ABSTRACT

EN-ACT (Energy Aware Computing) is a European project that aims to drastically reduce energy consumption of applications and systems based on Information and Communication Technology (ICT). Its objective consists in defining specifications for the design of energy aware software for IT infrastructures and mobile platforms. EN-ACT applies different strategies, for instance characterizing operations of mobile devices, generating development practices for energy aware code, offloading computation from mobile devices in the cloud, or analyzing IT infrastructures in terms of metrics that are independent of the implementation platform. These practices may enable developers and administrators to choose between different options when running an application, or to choose between different modes with varying energy consumption to reduce the total energy investment. EN-ACT has created a framework at different levels that allows to identify energy critical pieces of code, or to define metrics for the evaluation of performance at system level.

KEYWORDS

Energy, power, energy consumption, energy optimization, energy efficiency, mobile, smartphone, Android, applications, algorithms, performance, EA, metrics, KPI, IT, green IT.

1 INTRODUCTION

The increasing amount of services that can be executed on computing platforms such as cloud servers, personal computers, cellular phones, wireless sensor networks, etc., requires to define strategies for energy efficiency. This is required in particular by sustaining and environmental requirements, and to extend the autonomy of ubiquitous devices such as mobile phones. A number of hardware platforms are already equipped with software and operating systems (OS) that are able to reconfigure the hardware real-time for efficient energy use. However, designing and running complex applications on these platforms can be quite challenging if one wants to run an efficient yet high performance application with a minimum use of energy resources, while meeting non-functional requirements such as quality of service.

In scientific literature, there are mainly two strategies to save energy in computing devices: the first one covers mainly hardware; that is, determining how to save energy from the point of view of the architecture, materials and manufacturing electronic components. The second strategy focuses on the consumption of energy generated from the point of view of software execution which is able to optimize the hardware resources usage at run-time. Nonetheless, many issues remain open regarding energy saving strategies at application level and infrastructure level. Moreover, given the limited computational resources of certain platforms, e.g. mobile devices, the need to save energy aims at different points, for instance minimizing electronic garbage, extending battery life, and increasing the autonomy of a device.

In the EN-ACT[1] project we tried to tackle the aforementioned challenges. To do so, the project was distributed over several departments on different universities so each of them could contribute to the project in their individual field of expertise. In order to coordinate the course of action, regular meetings involving all partners have been held where the achieved progress and future steps were discussed and agreed on.
The rest of the paper is organized as follows. In Section 2 we present the services installed on the mobile device and discuss optimization techniques that aim to reduce the energy consumption of a mobile application. Section 3 shows how we measured the energy consumption of mobile devices in order to build an accurate energy model for sending and receiving data with it. In Section 4 we describe the cloud infrastructure that we used to run our experiments and presents an energy model for two exemplary programs. Section 5 outlines the achieved results for reengineering and optimization of energy consumption and other costs for large IT infrastructures, based on use of IT KPIs and powerful visualizations. Finally, we conclude and outline future work.

2 DESCRIPTION OF THE SERVICE INSTALLED IN THE DEVICE

2.1 Implemented energy aware techniques

The need of energy-efficient applications rose before the beginning of the smartphone era. The impact of the energy consumption has always affected mobile devices, but became a bigger issue with the introduction of modern mobile operating system such as iOS or Android, which promoted the utilization of hand held devices from simple communication and gaming features to more comprehensive computing applications. Device manufacturers strive to progress in providing better battery technologies but it is difficult to keep up with the evolution of the computing capabilities and the demand of energy required to operate modern smartphones. From the software point of view, large and complex applications tend to heavily stress different components of the device like CPU, memory, storage, etc.. There are different approaches to accomplish mature energy aware applications. After we surveyed the literature we selected two energy aware techniques - Method reallocation and Method offloading.

2.2 Method reallocation

A way of optimizing the energy invested to execute computational job is method reallocation [1]. Method reallocation refers to the analysis of a software product (i.e., an application) as means of a stack (for instance, kernel level, library level, API level, native code level, virtualized level, etc.) Having this structure facilitates the analysis of the energy costs associated to the components of each level of the stack (Figure 1).

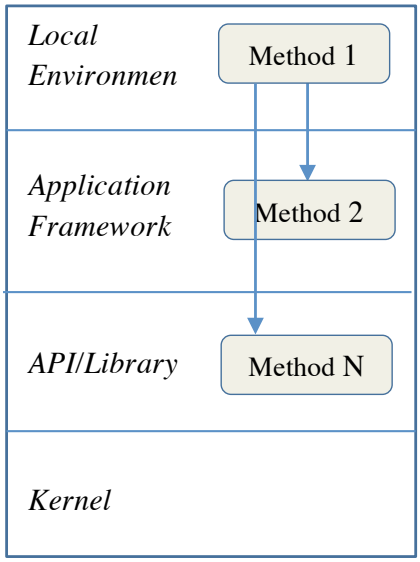

Figure 1. Software representation as a stack.

2.3 Method offloading

Process offloading (also known as cyber foraging or remote execution) [2–5] is a technique in which heavy computation jobs are sent to a remote computer; after the end of the remote execution, the result is sent back to the local machine (Figure 2).
The aim of offloading is to improve the performance of applications running under the low computational capabilities of laptops, moving

[1] http://www.en-act.eu

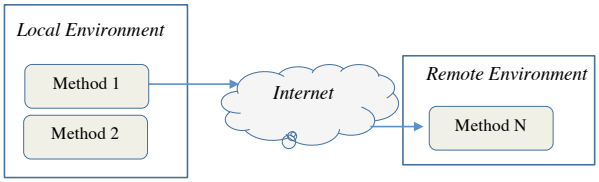

Figure 2. Schematic depiction of method offloading to a remote host.

the largest computations to more powerful surrogate computers. The application of the offloading technique became highly relevant with the advancement of smartphones aiming to extend their battery lifetime. This reorganization of the system aims to economize processor and memory, upon expenses of network usage and communication overhead [6–10]. In addition, method offloading can be orchestrated by self-adapting algorithms that may decide whether to or not to offload a computational job.

2.4 Device OS: Android OS

Android OS is an Open Source mobile operating system, which makes it a perfect fit for our project. It also provides a development platform with the necessary tools to create mobile applications (i.e., *apps*). The Android software stack includes a runtime environment, in which each application runs its own process using the Dalvik Virtual Machine, optimized to use less system resources. We leveraged this architecture to apply the discussed energy-aware approaches, namely method reallocation and method offloading. We developed an application that can access the different levels of the software stack to reallocate different implementations of a certain method, and to develop a software interface able to communicate with an external server to require the remote execution of the same method. To create an implementation that permits us to effectively sense out the energy consumed by exercising a method, we selected the implementation of a mobile app that executes a series of *energy-hungry* software benchmarks.

2.5 Benchmarks

The benchmark algorithms utilized on our experimentation were:

- *N-body:* Performs an n-body simulation of the Jovian planets. The n-body model was chosen as a benchmark as it represents the requirements of many simulations, and it requires memory access patterns that are representative of many operations like picture renderings, gaming, etc.

- *Fasta:* Generates and writes random DNA sequences in Fasta format. This algorithm features different instructions on array management, which can be found in string searches, sorting, record lookup and other usual operations in mobile devices.

- *Chameneos-redux:* Requests symmetrical thread rendezvous. This algorithm shows important characteristics on thread usage. In the context of mobile device, thread management is of paramount importance, since many tasks are executed asynchronously, and concurrently. This leads to the need of effective thread management to keep applications highly responsive.

- *Binary-trees:* Allocates and de-allocates binary trees, attempting to model properties of allocation requests. This algorithm exercises the management of data structures, which are typically used in searches, sorting, record lookup and other operations in mobile devices.

- *Fannkuch-redux:* Accesses a sequence of integers from 1 to 7 through indices and generates different permutations. The algorithm illustrates computer usage through comparisons and permutations. This benchmark is beneficial in the context of mobile devices since it exercises highly I/O operations and CPU power in a general scope, which is required for the good performance and experience of regular mobile apps.

- *Matrix multiplication* - the input is a text file with two matrices. A separate Java program was developed to generate such file with random values of the elements of the matrices (allows the adjustment of the dimension of both arrays. The output of the operation is two-dimensional array of integers written to another text file and stored in the phone with a different name.

- *Image filter* - the input is an image file of the most used formats (i.e. PNG, BMP, JPG, JPEG, GIFF). In this case the actual input parameter of the system's functions is a string indicating the full path to the file, but when the execution is offloaded to the remote server, all the bytes of the file are sent over the Internet. The server method reads the input file and applies the same algorithm with the same settings for filtering. In both cases the resulting image file is stored on the phone.

These benchmarks were selected to ensure different levels of complexity, resource usage (e.g., number of variables allocated, memory management), and programming techniques.

2.6 Execution scopes

Original Scope: Java

Java is the programming language officially supported by Android to develop applications. The Android software development kit (SDK) provides the necessary tools and APIs to create applications that are able to run on the Android platform in a Dalvik virtual machine, permitting to use the Java programming language to exploit most of the hardware and software resources of the mobile device (Figure 3).

Method reallocation: JNI-Enabled C Code

Thanks to the possibilities offered by the Java Native Interface (JNI), Android provides the Android Native Development Kit (NDK) which allows to write code in another programming language (known as *native code*) to fully interact with a Java application using a shared

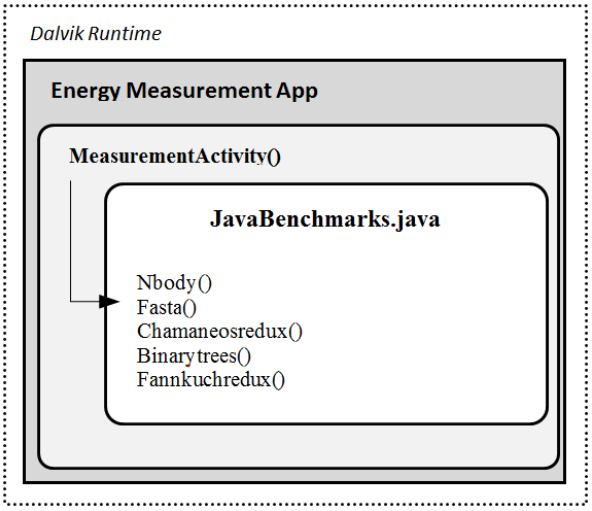

Figure 3. Schematic view of an Android App execution in the original scope. The app is written in Java and executed inside the Dalvik virtual machine.

library. The primary goal is to achieve binary compatibility of native method libraries across all Java implementations on a given platform. This permits interoperating with applications and libraries written in other programming languages (e.g., C, C++, and assembly) and lets native code access Java classes and objects. The use of native code helps to access otherwise unavailable lower-level functionalities, and to speed up critical pieces of an application (Figure 4).

Method Offloading

The third execution scope resides outside the mobile device. By offloading a method, the Android application solicits the execution of a remote method in lieu of an equivalent local method. In this way, the app makes the request to the remote machine, prepares the necessary arguments to get the job done, sends the name of the requested method and the arguments through the network, and waits for the answer calculated by the remote machine. The interaction between the Android application and the remote server is done utilizing a dynamic web project, in which the app can request the execution of a remote Java method using the http GET verb (Figure 5).

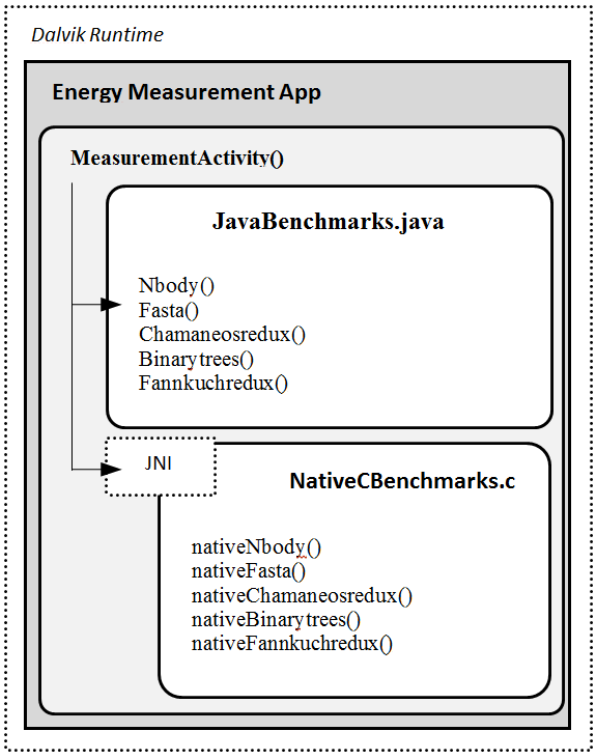

Figure 4. Schematic view of an Android App with method reallocation using JNI. The Dalvik virtual machine executes an application written in Java. This application can either execute the Java implementation of a function or or its version written in native code.

3 ENERGY CONSUMPTION OF COMMUNICATION TO THE CLOUD

In the following, we report on the energy modeling of the wireless communication.In order to accurately model the energy consumption related to the communication from the smartphone to the cloud, we empirically studied [11] different communication modes in an extensive set of experiments. Our measurements included WiFi, 2G, and 3G networks as well as a set of two different devices.

For the experiments we developed an Android application to conduct reproducible measurements. The application can automatically initiate file transmissions to avoid human interactions with the phone while performing the measurements. When Android APIs are used, instead, the only metric that can be collected is the percentage of battery consumed [12].

Regarding measurement methodology, due to the lack of functionality of the Android OS, the only possible way is by connecting a physical measurement device to the smart phone. The most common way is by connecting a voltmeter and an ampere-meter between the battery and the phone. A photograph of the assembly with a Samsung Galaxy S2 is shown in Figure 6. The measurement campaign has been conducted with Samsung Galaxy S2 and S3 smart phones, running Android v4.0.3 and v4.1.2 respectively. The phones were downloading and uploading files of different sizes from and to our web server, using different wireless technologies: 2G (EDGE), 3G (UMTS), and WiFi. Selected results are shown in Figure 7. As can be seen, 2G is by far the most consuming technology among the analyzed ones, followed by 3G, and then WiFi. We further derived that uploading is more expensive than downloading for cellular networks. This holds for WiFi too, but the difference is not so pronounced.

4 DESCRIPTION OF THE CLOUD INFRASTRUCTURE

To evaluate the method offloading functionality of our framework we set up a cloud test infrastructure. It consists of one node a dual socket CPU (Intel Xeon X5650) with two GPUs (NVIDIA GeForce GTX480) attached. It is reachable over a web interface where the user can upload input files to the provided services. The cloud infrastructure will automatically start the requested calculations once the data is uploaded and make the result available for download after the calculations are finished. For the data exchange we use files in JSON format. For our testing purposes we implemented two programs in OpenCL [13]:

- Color image to gray-scale conversion
- Gaussian blur filter for images

Both of these programs are working on image files. In order to read and write images, we use the QDBMP [14] image library. The two applications were made accessible over the internet. Once a mobile device uploads some input data for one of those programs, the cloud triggers the execution of it on one of the GPUs in the

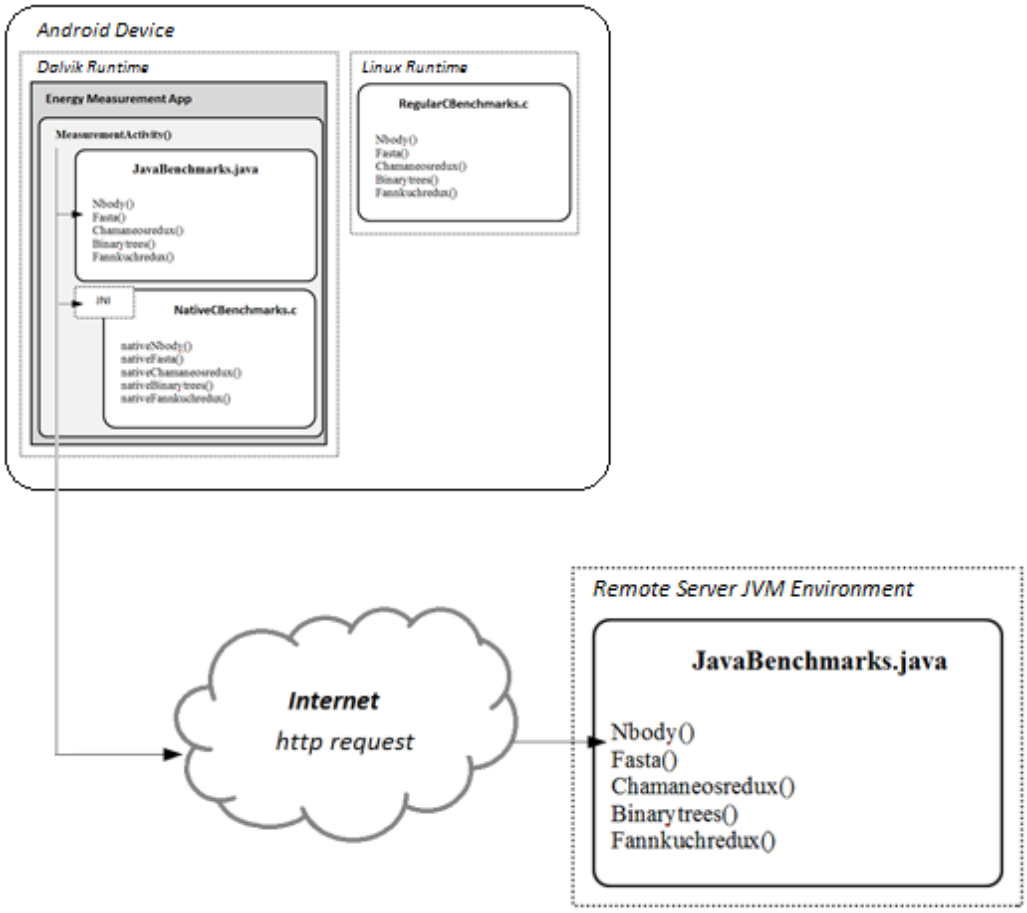

Figure 5. Schematic view of an Android App which can be offloaded and executed remotely. The picture shows all three options available to the measurement application: Th benchmarks can be executed locally either in Java or in native code, or offloaded to a remote host which performs the requested operation.

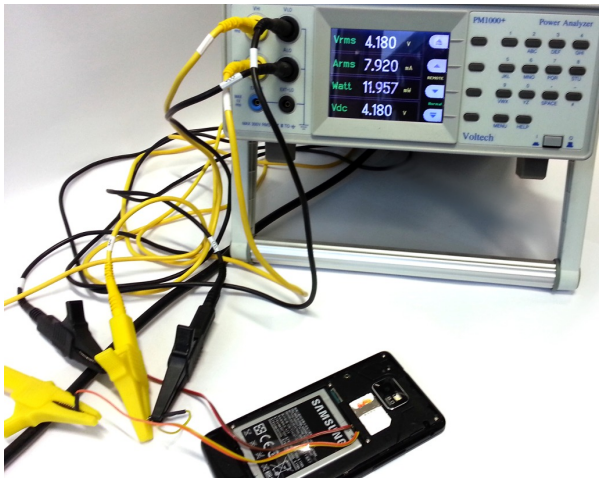

Figure 6. Photograph of the measurement setup: we connect a Voltech PM1000+ to directly intercept current from (and measure voltage at) the battery of the phones.

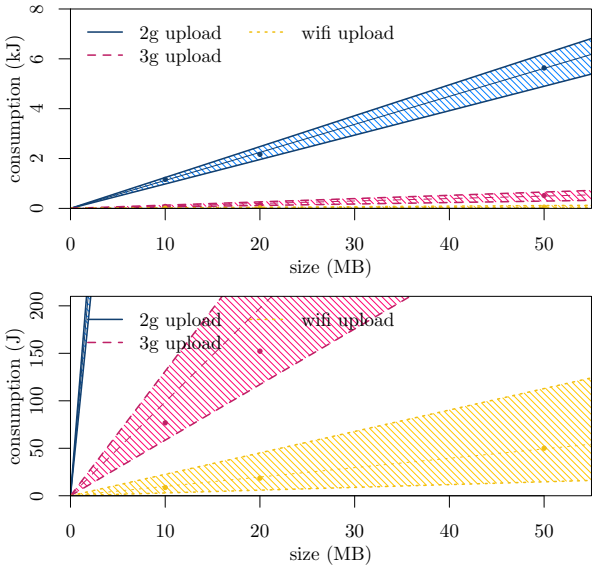

Figure 7. Energy consumption model for upload as function of transfer size. The shaded areas illustrate the variance, the center lines correspond to the prediction. The bottom figure is a zoom-in to for illustration purposes.

system. The result is then stored on a local, temporal file in the cloud which can be downloaded by the mobile device.

In order to gain a feeling when it pays off to offload an application from the mobile device to the cloud infrastructure with regard to the overall CO_2 footprint, we measured and evaluated the energy consumption in the cloud infrastructure. To measure it we used a Voltech PM1000+ in conjunction with the instrumentation library presented in [15]. This measuring device records the energy consumption of the entire cloud node at the power plug. This suits our use-case very well, as we are interested in the overall energy needed to perform the requested calculations. All energy measurements are given in Joule. To make the measurements as robust as possible and to avoid both measurement errors and fluctuations caused by operating system interventions, each measurement is repeated five times. The correctness of the result is validated using the Student's T-test [16].

From these measurements we constructed an energy consumption model for the two test programs on our cloud hardware. To generate that model we use the linear regression provided on [17]. This model could be used in the offloading decision process of the mobile device when aiming at minimizing the program's CO_2 footprint.

4.1 Color Image to Grayscale Conversion

This program takes a colored image and converts it to a gray-scale one. To do so, we use the*luminosity method* presented in [18]. This method calculates a weighted sum over the three color channels and sets all channels to this value. In our OpenCL implementation each *work item* [13] calculates one pixel of the output image.

Figure 8 shows the measured energy consumption of our cloud node when performing the gray-scale conversion on pictures of various sizes along with the prediction of our generated model. The dots represent the measured values, the line is the energy consumption that our model predicts for the various input sizes. The formula for our energy model is

$$y = 1.6633080164 + 3.1749274824 \times 10^{-7} x$$

where x is the size of the image in pixels and y is the consumed energy in Joule. Our model achieves a high accuracy with an means squared error of 7.7732 Joule^2. The average deviation of the model's result to the actual measurement is 7.35%.

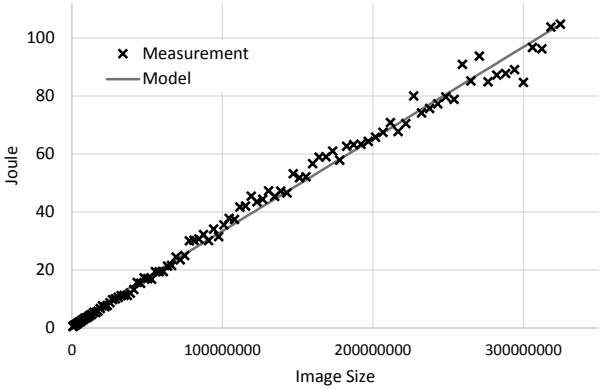

Figure 8. Energy measurements and energy prediction of our model of a color image to grayscale conversion with varying image size

4.2 Gaussian Blur Filter for Images

Our second testing applications calculates a Gaussian blur filter for the input image. To do so, it uses a Gaussian kernel to replace each pixel by a weighted sum of itself and it's neighboring pixels. In this algorithm, every color channel is treated separately. Again, each *work item* processes one pixel of the image. The size of the 2D Gaussian filter is $1/1000$ of the image height/width (with a minimum size of 3×3). The results of our measurements as well as the prediction of our model calculated by linear regression is shown in Figure 9. The formula that represents our model for the Gaussian blur filter is

$$y = 0.3311311580 + 5.1099443107 \times 10^{-7} x$$

where x is the size of the image in pixels and y is the consumed energy in Joule. Our model shows a mean squared error of 0.0933 Joule^2 over the entire dataset, which is equivalent to an average deviation of 3.43% from the measurement.

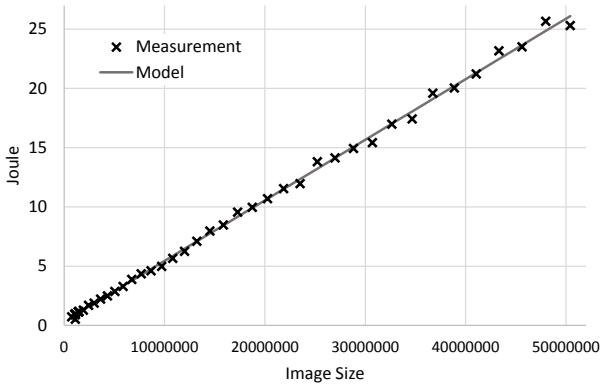

Figure 9. Energy measurements and energy prediction of our model of a Gaussian blur filter with varying image size

5 ENERGY AND COST OPTIMIZATION FOR LARGE IT INFRASTRUCTURES

Within another part of the EN-ACT project we have primarily focused on reengineering and optimization of energy consumption and other costs for *large IT infrastructures* and *data centers*[2]. Optimization of IT infrastructures is based on proposed green EA model (described in the sequel) and use of "green" KPIs.
The primary goal[3] is to develop a specialized energy management software for planning and optimizing IT infrastructures in terms of configuration, energy consumption and other costs. This software should support the specification, forecast, optimization and monitoring of energy-related optimization goals within IT infrastructure environments, based on advanced use of "green IT" KPIs. The envisioned optimization software itself will be based on *txture* (http://txture.org), an EA tool for textual IT-Architecture documentation and analysis.
Within the EN ACT project, concepts for this prototypical module, in form of a plugin for the txture tool, and a prototype of the interface were developed. In the sequel, the conducted activities and achieved results are outlined[4]:

5.1 Green Enterprise Architecture Model

At the beginning, the available business models were evaluated to identify the weak spots in those models and to use the findings to formulate our own EA (meta) model, displayed in Figure 10.
The EA model has three layers: technical, information and business layer. These layers are well integrated and inter-dependable. They show the interactions between the technical resources of the enterprise and the business level going through the software environment. *Technical Layer* is the lowest layer of the meta model. Infrastructure elements are components like servers, networks, storage units, devices, CPUs, etc. These components communicate with each other and enable information systems to run on them. *Information Layer* consists of information systems, applications and IS objects. *Business Layer* describes the relations among high level elements of the business view.
In order to monitor the effectiveness of the business processes and to analyze the efficiency of different layers, key performance indicators[5]

[2]This direction is justified with the following facts:

- The ICT industry is responsible for approximately 2% of the global CO_2 emissions (IT emissions will rise from 3% of the total global CO_2 emissions in 2012 to 6% in 2020 [19]). - Source: Gartner/HP/Mckinsey/WWF.
- IT adds up to 40% of the average corporate electricity bill; IT electricity demand is ever-growing; [19]
- Simple no-cost decisions made in the design of a new data center can result in savings of 20- 50% of the electrical bill, and with systematic effort up to 90% of the electrical bill can be avoided [20]; Upgrade of a server can reduce energy consumption by 15%; [19]
- The industry average for server utilization is only about 20%, with 32% running at or below 3% peak utilization. Through the implementation of the latest virtualization techniques the utilization factor can rise up to 80%. Enterprises can reap up to 65% reduction in server count through virtualization. [19]

[3]The main goal is out of EN-ACT scope and funding.

[4]Detailed discussion on the proposed EA model and framework is beyond the scope of this paper - detailed report, including definitions of all KPIs, discussion and a *case study*, can be found in one of EN-ACT project's deliverables, available online at: http://www.en-act.eu/images/EN-ACT_WP3-Deliverable.pdf - chapter 3.4. Detailed description of the energy optimization module can be found in [21].

[5]KPIs are metrics whose role is to help monitor vari-

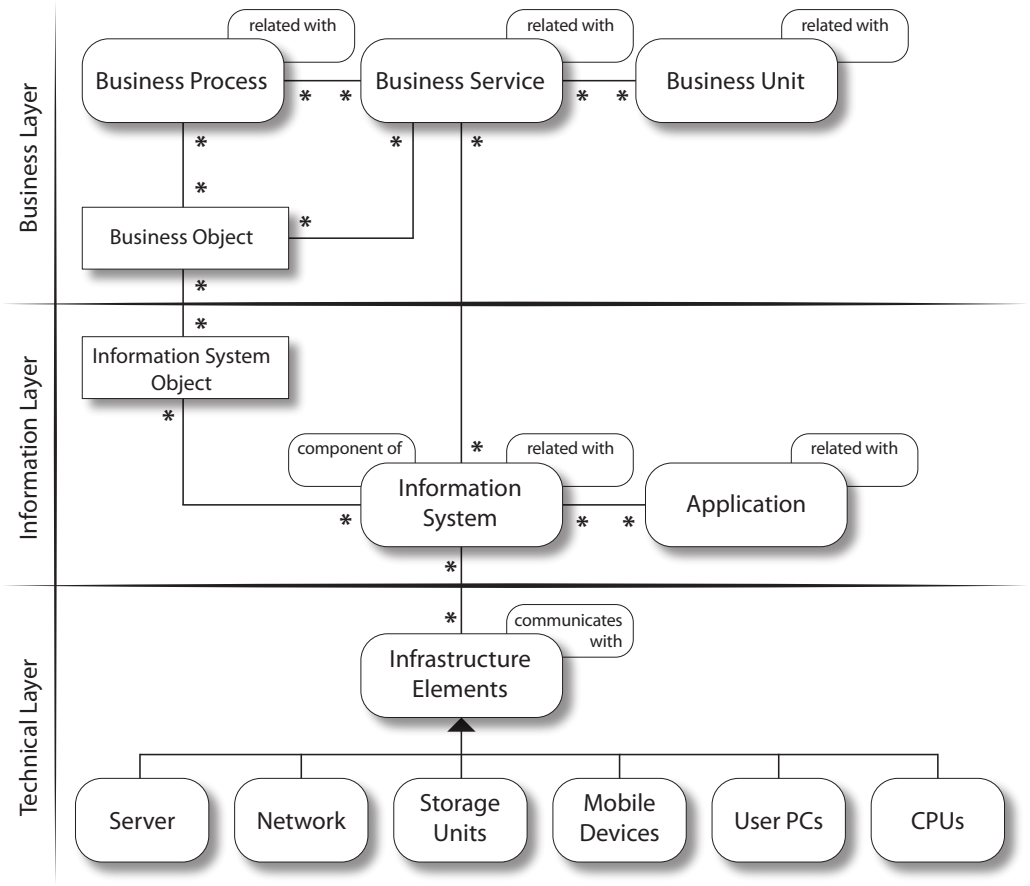

Figure 10. Layers (technical, information, business) of the "green" EA (meta) model

(KPIs) were extensively investigated and defined for each layer. Table 1 shows the designation of KPIs to layers in which they can be applied.

5.2 Strategy for data center energy efficiency optimization

Our focus is on IT and data center reengineering, which necessitated a study on how exactly can energy usage in a data center be optimized. In short, the strategy for achieving energy efficiency within data centers consists of the following steps (activities):

1. Determining the data center efficiency - achieved by defining the initial set of energy efficiency metrics and determining the energy baseline; As a result, it becomes possible to measure the result of any improvement while gaining the ability to pinpoint problematic areas fast;

2. Forecasting IT growth and power usage

Table 1. Layer specific metrics

Technical Layer
Power Usage Effectiveness and Data Center Infrastructure Efficiency [22–25], Green Energy Coefficient [23, 25], Energy Reuse Factor [23], Carbon Usage Effectiveness [23, 25], Deployed Hardware Utilization Ratio [24], Deployed Hardware Utilization Efficiency [24], Compute Power Efficiency [24], CO_2 Emission [24], IOPS/Watt [24, 26], Bandwidth/Watt [24], Capacity/Watt [24]
Information Layer
Response Time [26], Energy Aware Application Performance [26], Process Time/Job Duration [24], Throughput [24], Availability Rate [24], Reliability [24], Recoverability (Repairability) [24], Application Performance Indicators [24]
Business Layer
Green Energy Coefficient [23, 25], Human Resource Indicator [24], Compliance Indicator [24], Infrastructural Costs Indicator [24], Carbon Credit [24], Return of Green Investments [24], Paper/Digital Ratio, Paper Print Indicator

ous parameters of importance to the optimization strategy. Implementing these metrics allow the enterprise to determine areas to improve operational efficiency and to measure *current* and predict and plan *future* power usage, so that existing IT infrastructures can be optimized or new IT infrastructures can be planned so to be energy efficient from the start.

(based on consumption history and other variables) - good forecasts lead to better planning and more effective energy solutions;

3. Analysis of the current environment - implies examination of infrastructure adjustments to maximize energy efficiency and helps building a list of guidelines for future power reduction;

4. Energy management - implies constant monitoring (with calculation of KPIs) and management of energy consumption and application of necessary measures for correcting potential imbalances.

What needs to be emphasized, in regard to previous activities, is that they all must be driven by metrics (KPIs) and derived guidelines.

5.3 Energy & costs optimization module for the Txture tool

As a next step, we conducted an extensive investigation of existing EA tools and analyzed their support for optimizing IT energy efficiency. We found out that very few of the existing tools pay attention to energy efficiency optimization; in addition, their functionality is usually limited to (rudimentary) measurement and monitoring of energy usage. As a result, our focus was directed towards the development of an innovative *energy management software*.
First, we used the analysis of existing EA tools to formulate concepts and requirements for such a software. Afterwards, a graphic user interface prototype for the module was developed, with focus on use of powerful visualizations.
In continuation of the project, the defined concepts, requirements and prototype of the user interface should serve as a basis for future development of the energy & costs optimization module for *txture*[6], an EA tool for textual IT-architecture documentation and analysis with ability to cope with 100.000+ documentation items. With the envisioned energy optimization module, txture will gain functionality for planning and optimization of IT infrastructures based not only on standard criteria, but also on energy consumption, performance and other costs, and thus should provide enterprises of all sizes with a distinct possibility to plan their energy and other costs ahead or to optimize the existing consumption and costs.

Energy module's functionality. Typical *scenarios for optimization* (of energy consumption and other costs) of IT infrastructures we consider are:

1. Planning of a *new* IT infrastructure based on specific requirements which have potentially been implemented (and can thus be reused) in some other, existing infrastructure(s);

2. Planning of a *new* IT infrastructure, with requirements that haven't been previously implemented (no re-use) in another, existing infrastructure;

3. Expanding (significantly) an *existing* IT infrastructure;

4. Optimization of an *existing* IT infrastructure or its part(s) (full or partial optimization).

The envisioned energy optimization module should provide the following *functionalities*:

- Ability to build and visualize *different configurations* (variations) of the same IT infrastructure and to compare them.

- Ability to calculate, compare and visualize energy consumption and other performance indicators for different infrastructural configurations, based on available information[7], so as to allow the user to easily comprehend differences between these configurations. Using the energy module's

[6] Txture (currently in version 2), consists of a multi-user Eclipse-based modeling tool and a web-app to flexibly visualize the architecture for different stakeholders in any organization. A tool demo is available at http://txture.org.

[7] For example, (lowest, average and peak) energy consumption of a hardware device, calculation power, etc.

GUI (Figure 11) and its interactive controls, user can choose KPIs of interest from a set of predefined KPIs and can set minimum, maximum and weight (importance) values for each KPI, to better reflect the requirements. User can then additionally adjust these values, and eventually decide which infrastructural configuration suits his needs best.

- Clear and understandable (visual) representation of KPIs of interest for the energy optimization and monitoring strategy (Figure 12). One of the visualizations used in the txture energy optimization module for representing KPIs is the so-called *Radar graph* (Figure 13), as proposed in the Green Grid Productivity Indicator Tool [27]. Radar graph allows for clear and understandable visual assessment of how well the available resources are utilized, are the business objectives achieved and where are the potential areas for improvement. It can be used to show existing state, change progress, and the impact of "what if" scenarios in different configurations (variations of the same base IT infrastructure, as described previously).

  Target KPIs can be interpreted as business value parameters. Several metrics can be designated to a single business value in the radar graph; *each* business value parameter of interest to the stakeholder should have at least one metrics attached. Each axis of the example radar graph shown in Figure 13 represents a performance indicator defined to have a value range of 0%-100%[8], where these two boundaries represent theoretical minimum and maximum values for the KPI, i.e. the least and most desirable values of the KPI. Performance indicators represented on a graph may or may not be related, as is with any metric that has several components. Selecting indicators with interdependencies and relationships is strongly advised, however - it can be very effective for holistic impact analysis. On the other side, it also implies that a particular system change may result in some actual points becoming worse while others improve[9].

  Multiple performance indicators can be visualized for *several* different configurations at the same time (represented as lines and polygons of different colors in Figure 13.). By providing such a combined visualization of all performance metrics for several configurations, radar graph allows for easy visual comparison of these infrastructures, in terms of how well they fulfill all the desired target metrics.

5.4 Validation of the concepts and project

In order to evaluate the proposed green EA model, framework and KPIs, a case study was performed. Two web servers, hosting a fictional project, have been compared, in order to find out which one is more energy efficient. One server uses 5 year old technology, while the other one uses new, modern technology. The previously described metrics are applied on both of the servers and results are then compared. The goal was to show that there are significant differences in energy efficiency, which can be identified by applying these metrics. The details on the case study can be found in the aforementioned Deliverable for EN-ACT project. Eventually, we found out that our approach, with established concepts, worked fine for different situations and scenarios.
On the other side, extensive discussions with our industry partners showed great potential of the envisioned energy optimization software, as well as some space for improvement. Most importantly, the energy module will be enhanced by implementing full automation of the optimization process. This automation implies the ability of the energy optimization module to

[8] In some cases, clear maximum values don't exist (for example, there is no maximum value for PUE). In such a case, the axis end points have to be established based on target values or other estimates.

[9] For example, improving server utilization may actually decrease UDC (Data Center Utilization), which in turn may lower DCiE (Data Center Infrastructure Efficiency) if the data center doesn't scale with load. [27]

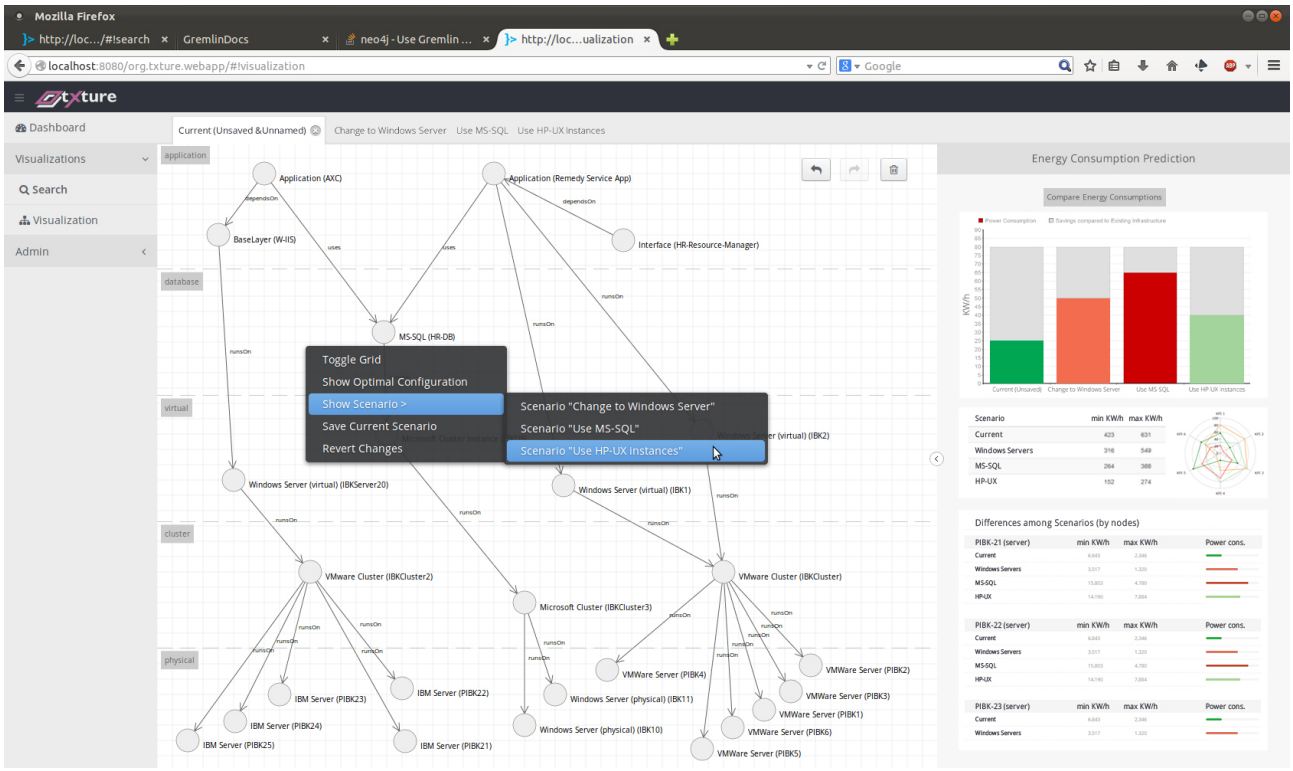

Figure 11. GUI of Txture's energy optimization module, with visual comparison of different configurations (right-side panel), including comparison of KPIs, list of configurations with statistics on energy usage (min, max) and differences (in nodes/items) among configurations.

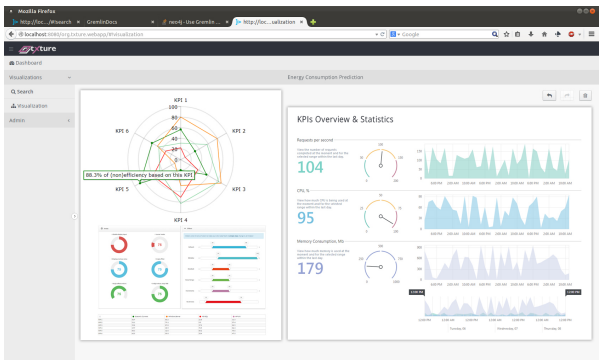

Figure 12. Powerful interactive visualizations, including radar graph and various statistical charts, of KPIs for different configurations.

automatically calculate and propose the most optimized infrastructural configuration, for either complete or partial optimization of the infrastructure, on any chosen set of KPIs (with set minimum, maximum and weight business values), based on available information. For partial optimization, user can choose which architectural nodes and items can be optimized and which are not to be affected.

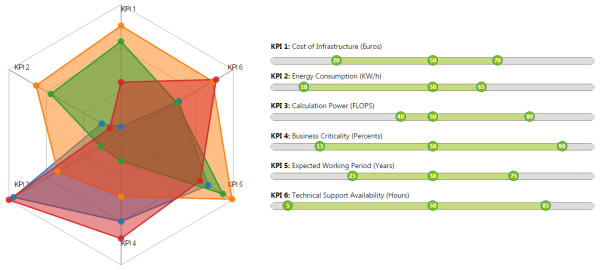

Figure 13. Radar graph with interactive selection (range) slider. Names of business values and corresponding KPIs (one or more - in this example each business value has exactly one attached KPI) are displayed above selector bars. In real-world scenarios, the left and right handle (min, max values) on the selectors can take values according to possible business values. For example, cost of infrastructure would be represented in real currency, energy consumption in KW/h, etc. Weights (indicating importance of the KPIs) always range between 0 and 100.

6 CONCLUSION AND FUTURE WORK

This paper presented the activities conducted by research teams involved in the EN ACT project, as well as the achieved results.

First, software optimization techniques were analyzed and was investigated how each one

can contribute to reduce the overall energy consumption of a mobile application. Furthermore, we also investigated the energy needed to transfer data to the network from a mobile device.
Second, compute intense applications have been made available to the mobile device in our own cloud infrastructure. We also measured the energy consumption for those applications and constructed an energy model for them. In future we plan to accelerate the applications and optimize their energy consumption in the cloud by using the Insieme compiler and runtime framework [28].
Last, challenges and our approach - a specialized software - to optimizing energy usage and other costs within large IT infrastructures, based on use of green KPIs and advanced visualization solutions, were sketched. Our approach supports not only the *energy-aware* optimization of existing, but also *energy-aware* planning of new IT infrastructures. In the continuation of the project, we have started to prototypically implement our approach within the context of EA tool *txture*. The implementation and testing phases will be followed by additional evaluation activities and further improvements of the framework.

REFERENCES

[1] Corral, L., Georgiev, A.B., Sillitti, A., Succi, G.: Method reallocation to reduce energy consumption: an implementation in android os. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, ACM (2014) 1213–1218

[2] Shivarudrappa, D., Chen, M., Bharadwaj, S.: Cofa: Automatic and dynamic code offload for android. University of Colorado, Boulde (2011)

[3] Zhang, Y., Huang, G., Liu, X., Zhang, W., Mei, H., Yang, S.: Refactoring android java code for on-demand computation offloading. In: ACM SIGPLAN Notices. Volume 47., ACM (2012) 233–248

[4] Chen, H.Y., Lin, Y.H., Cheng, C.M.: Coca: Computation offload to clouds using aop. In: Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on, IEEE (2012) 466–473

[5] Kumar, K., Liu, J., Lu, Y.H., Bhargava, B.: A survey of computation offloading for mobile systems. Mobile Networks and Applications **18**(1) (2013) 129–140

[6] Chen, G., Kang, B.T., Kandemir, M., Vijaykrishnan, N., Irwin, M.J., Chandramouli, R.: Studying energy trade offs in offloading computation/compilation in java-enabled mobile devices. Parallel and Distributed Systems, IEEE Transactions on **15**(9) (2004) 795–809

[7] Yang, K., Ou, S., Chen, H.H.: On effective offloading services for resource-constrained mobile devices running heavier mobile internet applications. Communications Magazine, IEEE **46**(1) (2008) 56–63

[8] Yang, S.: Manageable granularity in mobile application code offloading for energy savings. In: Green Computing and Communications (GreenCom), 2012 IEEE International Conference on, IEEE (2012) 611–614

[9] Flinn, J., Satyanarayanan, M.: Managing battery lifetime with energy-aware adaptation. ACM Transactions on Computer Systems (TOCS) **22**(2) (2004) 137–179

[10] Fei, Y., Zhong, L., Jha, N.K.: An energy-aware framework for dynamic software management in mobile computing systems. ACM Transactions on Embedded Computing Systems (TECS) **7**(3) (2008) 27

[11] Segata, M., Bloessl, B., Sommer, C., Dressler, F.: Towards Energy Efficient Smart Phone Applications: Energy Models for Offloading Tasks into the Cloud. In: IEEE International Conference on Communications (ICC 2014), Sydney, Australia, IEEE (June 2014) 2394–2399

[12] Petander, H.: Energy-Aware Network Selection Using Traffic Estimation. In: 1st ACM Workshop on Mobile Internet Through Cellular Networks (MICNET 2009), Beijing, China, ACM (September

2009) 55–60
[13] Khronos Group: OpenCL 1.2 Specification. (April 2012)
[14] Braudo, C.: Qdbmp: Quick n' dirty bmp library. http://qdbmp.sourceforge.net/ (2015)
[15] Ostermann, S., Eiter, T.S., Nae, V., Prodan, R.: A framework for region-based instrumentation of energy consumption of program executions. In: Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE, IEEE (2013) 4715–4720
[16] Gosset, W.S.: The probable error of a mean. Biometrika **6**(1) (March 1908) 1–25 Originally published under the pseudonym "Student".
[17] Arcidiacono, G.: Alcula, statistics calculator: Linear regression. http://www.alcula.com/calculators/statistics/linear-regression/ (2015)
[18] Cook, J.D.: Three algorithms for converting color to grayscale. http://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/ (2015)
[19] Hammond, C.: Introduction to green it. http://www.slideshare.net/thrashor/introduction-to-green-it (2012)
[20] Rasmussen, N.: Implementing energy efficient data centers (2006)
[21] Celebic, B., Breu, R.: Using green kpis for large it infrastructures' energy and cost optimization. In: International Workshop on Energy Management for Sustainable Internet-of-Things and Cloud Computing (EMSICC 2015), IEEE (to appear) Forthcoming.
[22] Belady, C.: Green grid data center power efficiency metrics: Pue and dcie. http://www.thegreengrid.org/~/media/WhitePapers/White_Paper_6_-_PUE_and_DCiE_Eff_Metrics_30_December_2008.ashx (2008)
[23] : The green grid: Harmonizing global metrics for data center energy efficiency. http://www.thegreengrid.org/~/media/WhitePapers/Harmonizing%20Global%20Metrics%20for%20Data%20Center%20Energy%20Efficiency%202012-10-02.pdf (2012)
[24] Kipp, A., Jiang, T., Fugini, M., Salomie, I.: Layered green performance indicators. Future Generation Computer Systems **28**(2) (2012) 478–489
[25] Mahmoud, S.S., Ahmad, I., et al.: Green performance indicators for energy aware it systems: Survey and assessment. Journal of Green Engineering **3**(1) (2012) 33–69
[26] Cappiello, C., Fugini, M., Ferreira, A.M., Plebani, P., Vitali, M.: Business process co-design for energy-aware adaptation. In: Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on, IEEE (2011) 463–470
[27] Belady, C.: The green grid productivity indicator. http://www.thegreengrid.org/~/media/whitepapers/white_paper_15_-_tgg_productivity_indicator_063008.pdf (2008)
[28] Jordan, H., Pellegrini, S., Thoman, P., Kofler, K., Fahringer, T.: INSPIRE: the insieme parallel intermediate representation. In: Proceedings of the 22nd International Conference on Parallel Architectures and Compilation Techniques, Edinburgh, United Kingdom, September 7-11, 2013. (2013) 7–17