U.P.B. Sci. Bull., Series C, Vol. 72, Iss. 3, 2010 ISSN 1454-234x

HIGH-LEVEL METRICS FOR ENERGY EFFICIENCY EVALUATION

Marius MARCU[1], Dacian TUDOR[2]

În această lucrare încercăm să surprindem aspectele ce sunt importante pentru implementarea sau evaluarea tehnicilor de eficientizare a consumului de energie de către dispozitivele de calcul mobile. Scopul principal al lucrării este acela de a propune, implementa şi valida un set de metrici de nivel înalt pentru evaluarea eficienței energetice a sistemelor de calcul şi a aplicațiilor framework de management al consumului. Prin utilizarea acestor metrici încercăm să identificăm sistemele hardware şi software care sunt cele mai eficiente pentru a executa o anumită aplicație. Aspectele originale vizate în cadrul lucrării sunt framework-ul de evaluare a eficienței, introducerea şi clasificarea metricilor de consum, definirea cazurilor de test pentru evaluare şi analiza rezultatelor experimentale obținute.

This paper tries to gather together all the aspects that are important to implement or evaluate energy efficiency techniques in mobile computing devices. The main goal of our work presented in this paper is to propose, implement and validate a set of high-level metrics for energy efficiency evaluation of computing systems or Dynamic Power Management (DPM) software frameworks. Using these efficiency metrics we intend to evaluate and identify the hardware system and DPM software which are the most efficient in terms of energy to run certain application. The original aspects of the current work are the proposed evaluation framework for energy efficiency, power consumption metrics definition and classification, new evaluation test cases for energy efficiency and analysis of experimental results.

Keywords: power consumption, energy efficiency, high-level metrics

1. Introduction

Recent developments in areas like mobile systems and wireless communications as well as the trend to incorporate a lot of new functionalities (such as WLAN, Bluetooth, GPS, multimedia, VoIP) have lead to their acceptance in almost all domains of activity. The element that has underlined the development of this domain was the reduction of the commercialization price of intelligent mobile systems (pocket PC, smartphone, PDA) together with their increase in performance [1]. The foresight of the immediate following years are propitious for the mentioned domain also, for example Portio Research group

[1] Assoc.Prof., Dept. of Computer Science, "Politehnica" University of Timisoara, Romania, mmarcu@cs.upt.ro
[2] Eng., Dept.of Computer Science, "Politehnica" University of Timisoara, Romania

estimates in its market study forecast for 2009-2013 [2] that the number of mobile subscribers will double in this period of time. Therefore we can say that we shall attend a pronounced development of the number of applications that run on mobile devices and this will lead to a growth of the request on the market of mobile application developers.

One of the most important evolution direction of mobile as well as traditional computing systems during the last years are oriented towards energy efficiency because of limited battery capacity of these devices. Different studies and high level discussions [3, 4] address actual power consumption problems of electronic and computing devices. The power consumption issue of computing systems is in general a very complex one [5] because every physical component in the system has its own power consumption profile depending especially on its execution workload, so that together with the physical components, the software applications has a big influence on the energy consumption [6, 7].

Power management and energy efficiency is a multidisciplinary field that involves many aspects (i.e., energy, temperature, reliability), each of which is complex enough to merit a survey of its own [8]. Unfortunately, despite considerable effort to prolong the battery lifetimes of mobile devices, there is no standard efficient solution established for all the mobile applications and their hosting devices [9]. Therefore in our work we addressed energy and power aspects in order to facilitate energy efficient mobile applications evaluation. Some of the current power management aspects are presented in the rest of this chapter.

Dynamic power management (DPM) strategies have been proposed and implemented in order to reduce the power consumption of the computing systems. This is a very large research domain, where significant work has been presented in [8-10]. The DPM algorithms minimize the energy consumption by selectively placing the system’s unused components in their specific low-power consuming states. Dynamic Voltage and Frequency Scaling (DVFS) is another well known power management mechanism which relies on dynamically reducing or increasing the processor voltage and frequency in order to control performance and power consumption [11]. DVFS techniques provide a way to reduce power consumption of microprocessors and other system components by altering the system or component performance.

Understanding applications characteristics is important for designing efficient power management (PM) systems [10] therefore the hardware level PM and operating systems and drivers level PM are not sufficient to obtain the maximum efficiency for a certain system. Different authors [10, 12, 13, 14] consider that a dynamic, adaptive PM infrastructure is needed to improve actual PM strategies. This infrastructure is to be implemented as a middleware or framework [13, 14] that continuously monitor workload characteristics and adapt the system or applications accordingly in order to obtain the best efficiency in respect to a set of requested constraints. In order to evaluate and implements such a energy efficiency framework for a mobile device a number of well defined metrics and test cases is needed.

An important aspect discussed also in [17] is the energy efficiency of different systems, processes and applications. Running the same application on different hardware or implementing the same functionality in an application with different algorithms may achieve distinct energy levels for the same functionality. The energy efficiency of computing systems or software applications is a complex concept [17] which has to be correctly defined in order to be used in such PM frameworks.

The main goal of our work presented in this paper is to propose, implement and validate a set of high-level metrics for energy efficiency evaluation of computing systems or DPM software frameworks. Using the efficiency metrics we intend to evaluate and identify the hardware system and DPM software which are the most efficient in terms of energy to run certain application. Next section describes the power profiling concepts used to extract energy efficiency metrics. Section 3 introduces and defines the energy efficiency metrics and Section 4 specifies the proposed test cases which can be used to compute the metrics. In section 5 we provide some energy efficiency results for some of the proposed tests and we conclude in Section 6.

2. Power consumption profiles

We consider that dynamic power management (DPM) mechanisms, when promoted at the higher layers (e.g. applications) can have an important impact on energy consumption reduction. The vast majority of existing literature deals with the physical and operating system's level, but in the last years a movement to higher levels DPM has been observed. We have considered this point of view for the obvious reason that applications may have knowledge or may estimate in a more accurate manner the necessary operations, their duration and importance. In addition, we see this focus change as a promising gap in power efficiency research.

We define a computing system as a number of physical components together with the running software applications using these components. Every application uses some components in order to finish its tasks and to provide the services it is executed for. The system components are considered as distinct power sources in the system which consume various power values function of their usage model. For each power consumption source in the system we can establish different power profiles (or power fingerprints) which denotes the power consumption of the component for a given utilization profile (e.g. applied stimuli or workload). Every power profile is identified by a set of power consumption levels corresponding to the different component usage models. A system component usage model assumes a certain workload level of the running applications using the component.

We understand by power profiles the variation in time of power consumption measurements related to the workload applied to the component. We define one profile per component and workload type in order to see how component power consumption changes with component's parameters when the workload is applied.

In order to run different evaluation test we designed and implemented a software framework for energy efficiency evaluation and energy usage optimization. This framework is not the subject of this paper but its design can be found in [15]. The energy efficiency evaluation framework presented in Fig. 1 continuously monitors the components and systems' parameters and computes energy efficiency metrics which are provided to the profiler components. Every power source in the system has a power profiler in order to estimate and predict its power consumption variation during its usage. We consider in our experiments only one component, the system CPU.

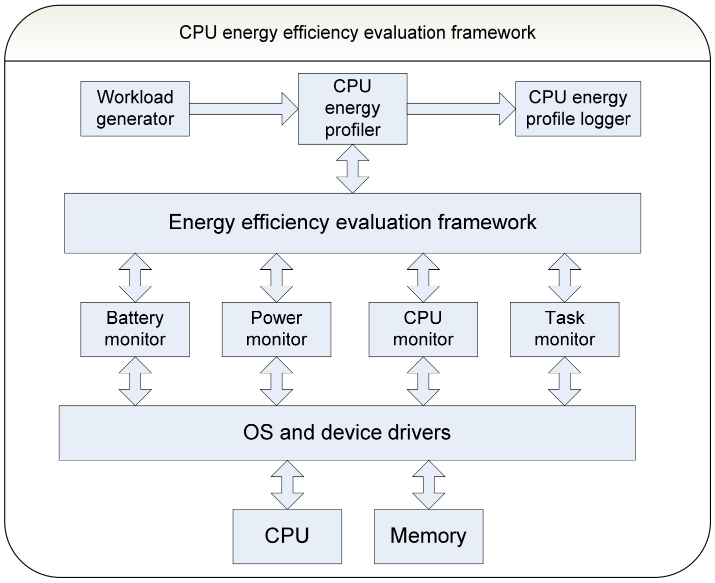

Fig. 1. Evaluation framework software architecture

In order to use power profiles we need to introduce a standard way to generate the profiles and in order to identify correctly power variation of the components. This standard method to generate power efficiency profiles we called power efficiency benchmarks that are described in detail in our previous work [16]. The power benchmark is defined as a software program that characterizes the power consumption of a system, component or application with respect to certain stimulus (workload). A power benchmark must by able to distinguish the way power consumption is increasing with workload related to idle state consumption and the type of workload. Therefore, we define a power benchmark to be composed by three intervals (Fig. 2): [0-t_1) idle mode power consumption, [t_1-t_2) the workload phase, when a certain stimulus is executed and [t_2-t_3) represents the releasing phase intended for the component to reach again the idle state power consumption.

Component measurements are the primary metrics the evaluation framework will use in order to characterize the current power consumption of the system. These measures could be obtained from different sources: battery driver, operating system, and other internal or external device drivers. Hardware component's power consumption depends on runtime usage profile which is dependent on the applications running on the device.

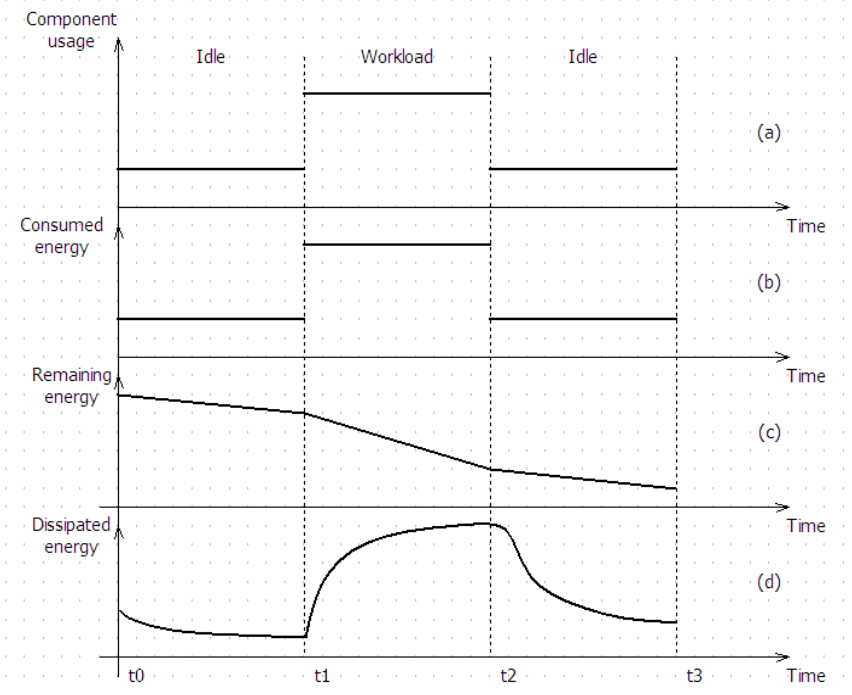

Fig. 2. Theoretical power consumption and system usage profiles

Conceptually, these profiles can be grouped in power consumption classes based on two kinds of parameters: system (usage) parameters and battery (consumption) parameters. Fig. 2 describes three types of power consumption profiles for the consumption parameters: consumed energy, remaining energy and dissipated energy. Each of these profiles is dependent on the applied workload. The workload profile determines the power states the device can switch while running (Fig. 2.a). Power consumption is shown in Fig. 2.b where there are different power consumption values for a component at different usage levels. Power consumption profiles are usually described as power consumption levels. Remaining energy, which is shown in Fig. 3.c, is the energy available in the device energy supply (usually the battery) that can be further used by the device. The remaining energy is decreasing with time and function of usage profile. The available energy is decreasing faster when higher usage levels are applied on the component. Dissipated energy is the fraction of the consumed energy loosed by the system as heat (Fig. 3.d).

To conclude, the power efficiency framework is based on following concepts:

- Workload components – are considered the system's physical or virtual modules that consume power and contribute to the overall power consumption of the system. For a mobile device we identified the following components: CPU and memory, WLAN, Flash file system, Audio chipset, Display, Video chipset, Bluetooth, GPS receiver, GSM chipset, etc. In this paper we address the first two components: the CPU and memory as a single component because for the higher levels they are tight related.
- Component measurements – define the measured parameters available for every addressed component. For every component in system there are a set of measurable parameters which describes how it is used and its power consumption state. Every component parameter is included in one of the four classes presented in Fig.2.
- Power efficiency metrics – are defined as high-level metrics computed from the measurements. High-level metrics are domain specific values computed for every component. These values are domain specific in terms of power consumption for component or application workload operations.
- Power profiles – specify how power efficiency metrics varies in time. We define one profile per component in order to see how component power consumption changes with component's parameters and workload type.

3. CPU energy efficiency metrics

CPU is the most important component in the system and it has a substantial contribution to the overall systems' power consumption. CPU parameters could be measured globally for all applications running in the system or specifically for every application registered and monitored by the framework. There are two types of CPU parameters: static and dynamic. Static CPU parameters are known a priori for a certain system and could be configured or achieved from OS. Dynamic CPU parameters could be measured in real time (online) for the running applications and they are variable with the CPU utilization by these applications. Dynamic parameters could be achieved for the whole system or per application, but static CPU parameters are usually global parameters. Dynamic parameters are computed as an average on a fixed amount of time which is a settable parameter for the framework, and we call it framework sample time period (FSTP).

We consider in our model that one application is composed, in terms of CPU, from one or more threads. The application threads could be started when the application is launched (as a pool of threads) and they are alive during the whole application lifetime or the application could start dynamically a number of threads, when they are needed. Every application thread has a unique ID, the processor core it is allocated to, the priority and CPU and memory usage by the thread during the current time period. The CPU power consumption due to a certain application is to be computed from CPU parameters computed for every thread in the observed application.

CPU core parameters are used to characterize statically and dynamically the CPU configuration of the system:

– *CPU cores number* (cpu_{no}) - specify the number of cores available in the system. This parameter is static and is well defined globally for a certain system. [cpu_core_number]

– *CPU core available power states* (cpu_{ps}) - every CPU core has a well defined number of power states: active, low power, sleep, inactive. For every power state the power consumption level has to be known and statically configured. A CPU power state is defined by its name, power consumption and the activation time needed by the CPU core to enter this power state. [cpu_power_states_number, cpu_power_state_name, cpu_power_state_value, cpu_power_state_uptime]

– *CPU core runtime power states* - specify in every period in time the current power states the CPU cores are used in. For every core, the runtime power state is one of the available power states for the CPU presented before. [cpu_runtime_power_state]

CPU usage parameters specify how the CPU is used by all running applications:

– *Global CPU time* (cpu_{gt}) - specify the time the CPU or its cores are used by the running programs’ threads in the current evaluation time period. CPU time is expressed in clock tick counts and defined as the amount of time the CPU is actually executing instructions of the running applications. Global CPU time is the difference between the total amount of time the CPU executes instructions counted from the system begin time to the FSTP end time and the amount of CPU time from the system begin time and the start time of the FSTP [global_cpu_time].

Thread usage parameters specify how the CPU cores are shared by the running threads:

- *Thread CPU time* (cpu_{tt}) - specify the amount of time a thread has actively used the CPU core during the framework sample time period. This parameter is computed for every combination of thread and CPU core in order to show how a thread was executed and migrated on different cores. The same with global CPU time, it is expressed in clock tick counts and can be computed by difference between the total amount of time the thread used the CPU from it's begin to the stop time of the current FSTP and the amount of CPU time from the thread begin and the start time of the FSTP [thread_cpu_time].

The evaluation framework is responsible to provide to the higher layers energy efficiency metrics such as information on power consumption (average, instantaneous), application or component power efficiency. Energy efficiency metrics specify how much energy is needed to complete an application specific job.

Global metrics are the parameters that characterize the power consumption and energy efficiency of the whole system:

– *Global power consumption* (P_s) is the average power consumption of the system during the FSTPs (framework sample time period). This metric is the absolute power consumption expressed in Watts and is obtained from the battery measurements. In case the battery driver does not provide any information which could be used to estimate absolute system power consumption, relative values for this metric are accepted. [global_power_consumption]

– *Global energy efficiency* (E_s) specifies which the costs in terms of energy are in order the system finishes its work. The efficiency is measured in operations per consumed energy and is computed as the useful work executed by the system per quantity of energy consumed to finish the work. The system's operations could be defined in terms of users' operations, component operations or applications operations. [global_energy_efficiency]

– *Total available energy* (E_t) is the energy that the system's power sources can supply within specific parameters in order the system works properly. The energy is expressed in joules or in terms of battery capacity (Wh). This metric is available from battery measurements. [total_available_energy]

CPU metrics are the framework computed values that characterize the power consumption and its efficiency of a hardware component:

– *CPU power consumption* (P_{cpu}) is the average power consumption of the CPU during every FSTP. This metric specifies the part in the global power consumption due to the CPU. This power consumption is a component specific metric which is computed based on the low level measurements for the component. The metric is expressed either in absolute values (recommended) or in relative values (when poor measurements are available). [component_power_consumption (CPU)].

– *CPU energy efficiency* (E_{cpu}) specifies which the costs in terms of energy are in order the CPU execute a specific amount of work. CPU energy efficiency is the report between CPU times and energy consumed by the component for the time period the CPU time is computed. [component_energy_efficiency]

Application metrics are the framework provided parameters that characterize the power consumption and power efficiency of a software application:

– *Application global power consumption* (P_{app}) specifies how much of the system power consumption is due to the specific application. This metric provided by the framework core is computed based on the application's threads measurements and the used components' parameters. The metric is expressed either in absolute values (recommended) or in relative values (when poor measurements are available). [application_power_consumption (application)]

– *Application energy efficiency* (E_{app}) specifies which the costs in terms of energy are in order the specific application executes its work. The efficiency is measured in operations per consumed energy and is computed as the useful work executed by the application per quantity of energy consumed to finish the work. [application_energy_efficiency]

– *Application CPU power consumption* ($P_{app\text{-}cpu}$) is the power consumption of a hardware component induced by a specific application. This metric is computed by the framework function of the application's threads measurements and specific component parameters. The metric is expressed either in absolute values (recommended) or in relative values (when poor measurements are available). [application_power_consumption (application,CPU)]

– *Application CPU energy efficiency* ($E_{app\text{-}cpu}$) specifies which the costs in terms of energy are in order the CPU execute a specific amount of work for a certain application. CPU energy efficiency is the report between CPU times and energy consumed by the component for an application for the time period the CPU time is computed. [application_energy_efficiency]

4. CPU energy efficiency test cases

The process of extracting power profiles for a computing system is called system power profiling or characterization. We selected some tests in order to extract power profiles of system's CPU and its cores. Once the test are established, we run these test many times on the same machine with different environment conditions, we test different types of workloads and we try to emphasis the influence of different components or applications on the overall power consumption of the system.

CPU power profiles present a description of the CPU power consumption variation over the time when it executes different workloads with well known parameters. CPU power profiles are considered as a characteristic feature of the CPU because when it executes a specific workload a number of times with the same parameters, the same power consumption profiles are obtained. In order to produce the CPU power profiles a number of tests are further introduced. CPU power profiling test are based in the power benchmarks introduced before. During the profiling phase the tests are running for a certain amount of time when the measurements and efficiency metrics are collected and recording by the framework.

– *CPU idle power consumption* - The CPU idle state power profile is the power consumption variation over the time when the CPU is idle and do not execute any application, except the default operating system services. When the idle test is executed the CPU configuration parameters are set to default values, no workload is applied and no other user application or interaction is allowed. CPU default parameters mean that only one core is active and this core is set in the active power state, not in any of the available low power states.

– *CPU instruction set power consumption* - The CPU consumes distinct power values for any instruction it executes. Some complex instructions consume more power to complete than the simple CPU instructions. However, at the higher levels the system cannot seize the differences between the power consumption levels of two instructions, but we need in the framework an idea of power consumption of different instruction classes: integer, memory, floating point, etc. In case we know what kind of operations an application thread uses we can estimate its power consumption for a specific workload.

– *CPU usage power consumption* - The same workload the CPU can execute at different usage levels which may imply different power consumption levels for the same type of workload. Using this profiling test we want to emphasize the relation between power consumption and CPU usage for certain workloads. Inside this test the same workload is repeatedly executed with different sleeping times in order to achieve different values for CPU usage.

– *Multithreading power consumption* - Another proposed CPU profile test is to launch the same algorithm workload on different thread counts using one single core. Using this test the OS task scheduling and switching operations along with workload operations are observed in order to get their power consumption. In a multithreading test case it creates a number of threads running the same workload in order to see how thread count influence the overall power consumption, and further to understand how much power consumes the OS task scheduling and switching mechanisms.

– *Multicore power consumption* - For multi-core processors two new tests are needed to establish the relation between the number of active cores and their individual and cumulated power consumption. First, the test is used to activate every core one at the time to run the same workload, in order to see how much power consume every active core. Next, the test activates successively step by step one more core while cheeping the previous active and run the same workload on every active core, in order to emphasis the increasing power consumption for every new active core.

– *Multhithreading and multicore power efficiency* - A much more general power profiling test is to launch an arbitrary number of distinct workload threads on an arbitrary number of active cores. Using this test we can obtain the efficiency of multi-core CPU for certain types of workloads in order to obtain the optimum number of threads and activated cores.

– *Synchronization power consumption* - Synchronization mechanisms are an important aspect in multithreading and multi-core programming. In order to observe the power consumption of synchronization we propose another test. The test creates a number of threads running on the specified cores and the threads use a number of synchronization objects to access shared data.

– *CPU states power consumption* - CPU cores recently implement more than the active/inactive power states. Also modern technologies like DFVS (Dynamic Frequency and/or Voltage Scaling) are also implemented. In order to detect power consumption of CPU core's states we introduce another power profile. With this profile the same workload is executed on the same CPU core selecting in each test step the next available power state or DFVS step.

5. CPU energy efficiency results

In this section we show some of the results we obtained after execution of the proposed CPU energy efficiency test cases. The test we executed on three hardware systems: Fujitsu Siemens laptop E series with Intel Core Duo CPU at 2 GHz and 1.5 GB memory; Dell Inspiron laptop with Intel CPU at 1.8 GHz and 512 GB memory; Dell Optiplex desktop with Intel Core Duo at 2.4 GHz and 2Gb memory.

We run the same tests on every system a number of three times and we logged all measurements and computed metrics for further analysis. One test last 15 minutes: 5 minutes idle, 5 minutes workload and 5 minutes again idle. The measurements are sampled every one second and based on them the efficiency metrics are computed.

After we execute a CPU energy profiling test with integer workload we obtained the drawing in Fig. 3. The power consumed by the system we measured with an external device we build which measures the AC power socket where the system was plugged in. CPU temperature was measured for every core in the processor through the build in temperature sensors. The CPU usage was computed using the operating system API.

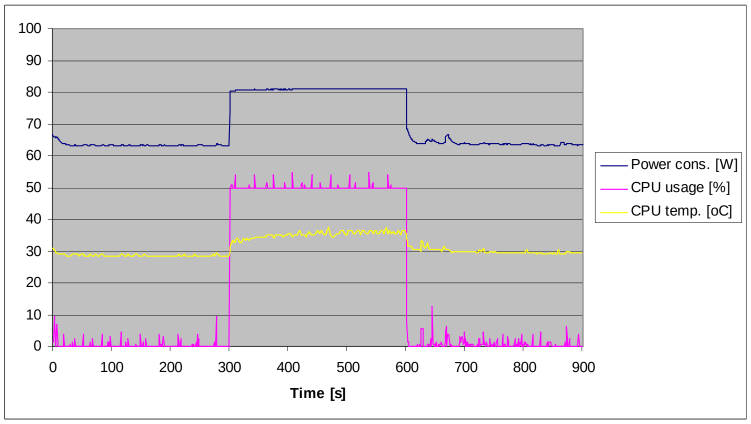

Fig. 3. CPU measurements

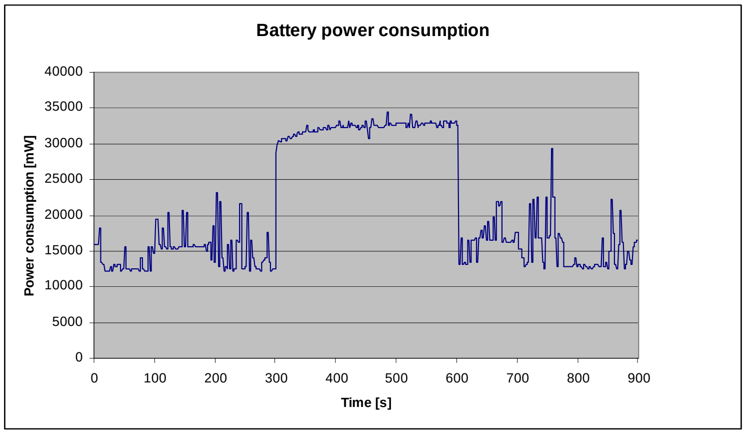

Fig. 4. Heat dissipation and power consumption

Part of the battery energy of a mobile device is transformed into heat. The increase in temperature enforces more energy to be consumed. In Fig. 4, power consumption profile for the previous memory workload is presented. During the second phase of the benchmark, when the workload is applied, the temperature of the processor and also the temperature of the entire mobile device increase (in our example the temperature increases from 60 to 90^{o}C). This increase in temperature of has an effect on power consumption, and a smooth increase (from 30W to 34W) during phase 2 of the benchmark can be observed in the current profile. This increase of approx. 4W during the workload execution is due to the heating of the device.

Various computing and mobile systems consume different power values depending on a lot of factors like: enabled components, component parameters, usage pattern, running applications, etc. Using the proposed high level metrics we can estimate which are the most efficient systems in terms of energy for certain workload execution. For example we run the same workload (a CPU integer benchmark) on three systems and we obtained different performance and power consumption values (Fig. 5). The energy efficiency is shown in Fig. 6 where it can be observed that not the most performing system is the most energy efficient one.

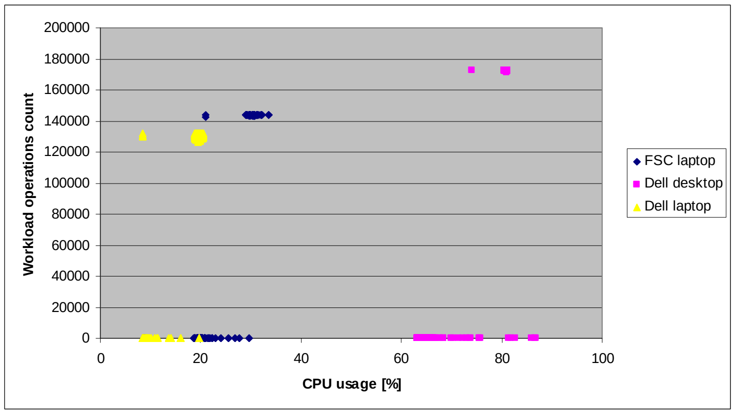

Fig. 5. Integer workload performance and energy results

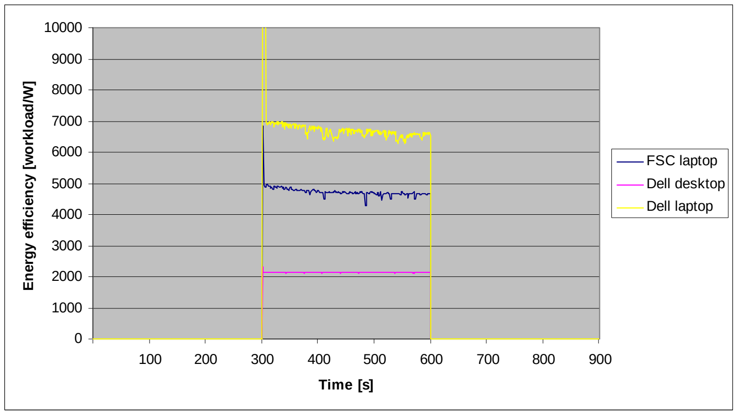

Fig. 6. Energy efficiency of workload execution

The same workload can be executed on the same system within different system parameters or DPM parameters. For example if we run the same integer benchmark with various CPU usage values we obtain different energy efficiency values (Fig. 7). In this test, run on the FSC laptop, the energy efficiency of the workload has better values for high CPU usage levels, but there are some experiments where better efficiency is achieved for lower CPU usage levels because at the higher levels the cooling energy increase when the CPU temperature increases.

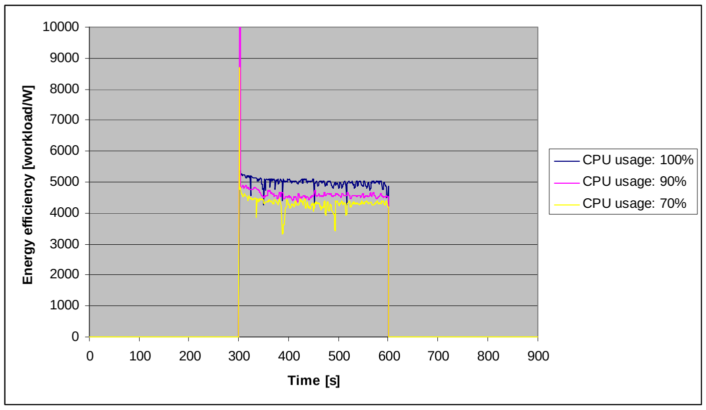

Fig. 7. Energy efficiency of CPU usage levels

In the test shown in Fig. 8 and 9 we run the same integer workload on the same system but we used different CPU cores and one or two workload threads. We run one thread scheduled by the operating system on two CPU cores and the same thread we run on one single core by setting the thread CPU affinity to core 0. We also run two threads on one core by setting the affinity of both threads to the core 0. These three tests show similar energy efficiency values (Fig. 8) but if we take a closer look at the obtained values (Fig. 9) we observe some differences. Scheduling one workload thread on two cores has higher efficiency because of reduced temperature, switching the thread execution from one core to another allows unused cores to cool. When executing one thread by one single core, the efficiency is the same in the beginning but decreases in time while core temperature increases. The lower efficiency has the execution of two threads on one core because of both increased temperature and thread context switching. In case we run two threads on both CPU cores an increase of energy efficiency is observed (Fig. 8) but with a multiplied factor of 1.7.

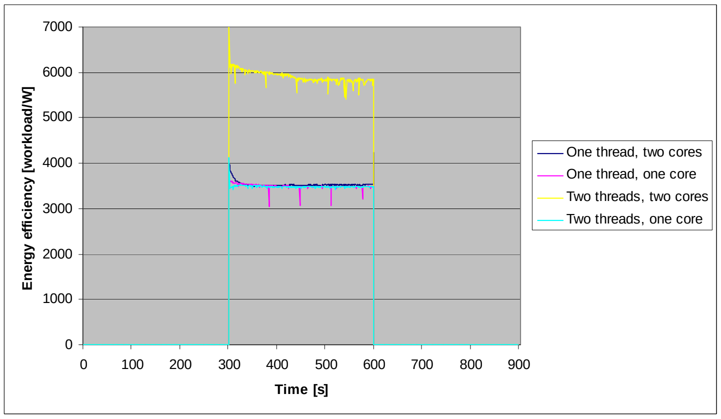

Fig. 8. Energy efficiency of CPU cores and running threads

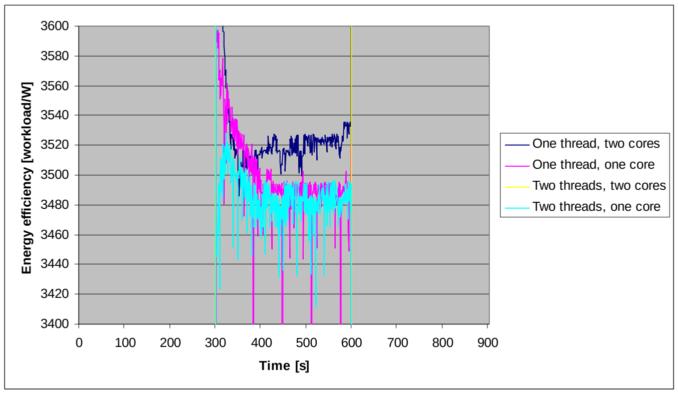

Fig. 9. Energy efficiency of CPU cores and running threads (zoom in)

6. Conclusions

In this paper we presented the aspects that are important to implement or evaluate energy efficiency techniques in mobile computing devices. The main contribution of the work presented in this paper is the definition, implementation and validation the set of high-level metrics for energy efficiency evaluation of DPM mechanisms. We proposed also a set of test cases which can be used to profile CPU efficiency. Using these efficiency metrics we intend to evaluate and identify the hardware system and DPM software which are the most efficient in terms of energy to run certain application. The original aspects of the current work are the proposed evaluation framework for energy efficiency, power consumption metrics definition and classification, new evaluation test cases for energy efficiency and analysis of experimental results.

The proposed metrics are implemented and further used in an energy efficiency software framework for energy efficient mobile applications development. The main goals of the energy efficiency framework is to provide energy efficiency monitoring services and a high level feedback loop to applications and the operating system or drivers in order to improve the both applications' and system's power efficiency. In our system, we define a series of power components which are specialized components that are accountable for power consumption (e.g. CPU). The proposed metrics are used in the framework to estimate the energy efficiency of an application when executed with certain parameters.

A C K N O W L E D G M E N T

This work was supported by Romanian Ministry of Education CNCSIS grant 680/19.01.2009 and the eMuCo FP7 project (www.emuco.eu).

R E F E R E N C E S

[1] *David Haskin*, Top Mobile and Wireless Trends for 2007, PCWorld, Dec. 2006

[2] Portio Research, "Worldwide Mobile Market Forecasts 2009-2013", Market Research, http://www.portioresearch.com/WMMF09-13_brochure.pdf , 2009

[3] Gartner Inc, Top 10 Strategic Technologies for 2008, http://www.gartner.com/, 2008

[4] European Parliament, Action Plan for Energy Efficiency: Realising the Potential, INI/2007/2106, Ian. 2008

[5] *J. Sorber, N. Banerjee, M. Corner, S. Rollins*, Turduken: Hierarchical Power Management for Mobile Devices, The Third International Conference on Mobile Systems, Applications and Services, Mobisys2005, Jun. 6-8, 2005, USA

[6] *L. Zhong, N. Jha*, Energy Efficiency of Handheld Computer Interfaces: Limits, Characterization and Practice, The Third International Conference on Mobile Systems, Applications and Services, Mobisys2005, Jun. 6-8, 2005, USA

[7] *M. Marcu, D. Tudor, H. Moldovan, M. Micea*, Power Profile Evaluation of Battery-Powered Mobile Applications, 14th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2007, Dec. 11-14, 2007 Marrakech, Morocco, pp. 1015-1018, ISBN 1-4244-1378-8

[8] *V. Venkatachalam, M. Franz*, Power Reduction Techniques For Microprocessor Systems, ACM Computing Surveys, **Vol. 37**, No. 3, Sep. 2005, pp. 195–237

[9] *J. Flinn, M. Satyanarayanan*, Managing battery lifetimewith energy-aware adaptation. ACM Transactions on Compute Systems (TOCS) 22, 2 (May 2004)

[10] *K. Rajamani, H. Hanson, J. Rubio, S. Ghiasi, F. Rawson,* Application-Aware Power Management, IEEE International Symposium on Workload Characterization, IISWC, San Jose, (2006)

[11] *Venkatesh Pallipadi*, Enhanced Intel SpeedStep Technology and Demand-Based Switching on Linux, Intel Software Network (2008)

[12] *Shivajit Mohapatra and Nalini Venkatasubramanian*, A Game Theoretic Approach for Power Aware Middleware, Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, Toronto, Canada, 2004

[13] *Bo Chen, William Pak Tu Ma, Yan Tan, Alexandra Fedorova, Greg Mori*, GreenRT: A Framework for the Design of Power-Aware Soft Real-Time Applications, Workshop on the Interaction between Operating Systems and Computer Architecture, WIOSCA 2008, Beijing, China (2008)

[14] *Ashwin Iyenggar, Ambudhar Tripathi, Ajit Basarur and Indranil Roy*, Unified Power Management Framework for Portable Media Devices, IEEE International Conference on Portable Information Devices, PORTABLE07, (2007)

[15] *Dacian Tudor and Marius Marcu*, Designing a Power Efficiency Framework for Battery Powered Systems, ACM Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference, Israel, 2009

[16] *Marcu Marius, Vladutiu Mircea, Moldovan Horatiu, and Popa Mirce*a, Thermal Benchmark and Power Benchmark Software, Proceedings of the 12th IEEE International Workshops on THERMal Investigations of ICs and Systems, THERMINIC 2006, France, Sep. 2006.

[17] *Michel Feidt*, Energy efficiency and environment, U.P.B. Scientific Bulletin, Series C, **Vol. 72**, Issue 1, 2010.