UDC 004.9:658.14.012.2, 004.051

A. Y. KRYVTSOV, S. V. HONTOVYI

Donbass state technical university, Ukraine

APPROACH TO IMPROVE ENERGY EFFICIENCY OF INFORMATION SYSTEMS

This article covers the problems of improving the energy efficiency of information systems. Two main directions - hardware and software - have been identified. The existing methods to reduce energy consumption at the software level have been analyzed. A new holistic approach to the development of energy efficient software that passes through its entire life cycle has been proposed. This approach consists of the five levels, including requirements, design, building, compile and testing. For each level algorithms, methods and techniques to improve the energy efficiency have been proposed.

Key words: *energy efficiency, software life cycle, green compiler, C-state, green software.*

Introduction

The problem of energy efficiency is one of the most important in the world today. The increasing hardware performance led to a powerful surge of energy consumption in workstations and servers. Especially it has affected the large data centers, which contain a lot of computer facilities and networking equipment.

Additionally, the problem of high energy consumption touched the scope of mobile information technology. The increase of hardware energy consumption was more rapid than the development of electricity storage technologies, in particular increasing the capacity of the batteries. This fact is expressed in a substantial reduction in battery life of mobile devices at the expansion of their functionality.

Thus the problem of increasing the energy efficiency of information technology is due not only to the necessity to save energy, but also with the need to increase the battery life of mobile devices.

The energy consumption of any device depends not only on the hardware, but also on its software [1]. Therefore, methods to improve the energy efficiency are created both for the hardware and the software.

The methods relating to the software in general optimize the upper levels of the system design hierarchy, thus eventually giving more substantial results than the hardware methods.

Problem statement

All information systems can be structured into three layers - hardware, operating system (OS), and application program. Even though the two software layers do not consume the power directly, they control the behavior of the hardware and have strong impact on the energy consumption of the hardware layer. However, most research has focused on the energy optimization of the hardware itself [2].

There are several other reasons to focus on the software:

– some progress has been made on the level of hardware, the network, and the data centre;

– focus on reducing energy loss in the power supply chain;

– hardware consumes energy on behalf of application of the software;

– the software design and construction are currently mostly energy-oblivious;

– focus on reducing the energy demand at the source.

The Software Improvement Group research shows that only 25% of the algorithms in the software provide optimal computational efficiency. The same studies show that only 65% of the technology features provide the solutions to the tasks, the remaining 35% provide the optional functionality [3]. On figure 1 we can see the power loss chain.

Thus we see that the hidden potential of the software in this area is huge. The given researches are only some of the many confirmations of weightiness of the software in the problem of the energy efficiency of the information technology.

The problems of energy efficient software have been widely discussed by home and foreign scholars. In some areas they achieve very significant results [2, 4-7], for example in the work [4] a series of recommendations for the development of green software is partly formulated. However, all these studies are somewhat fragmented.

Thus, currently the green optimization methods for the software are in the process of development. Therefore the main purpose of the article is to investigate the existing methods of reducing energy consumption of the software and to improve the energy efficiency of the

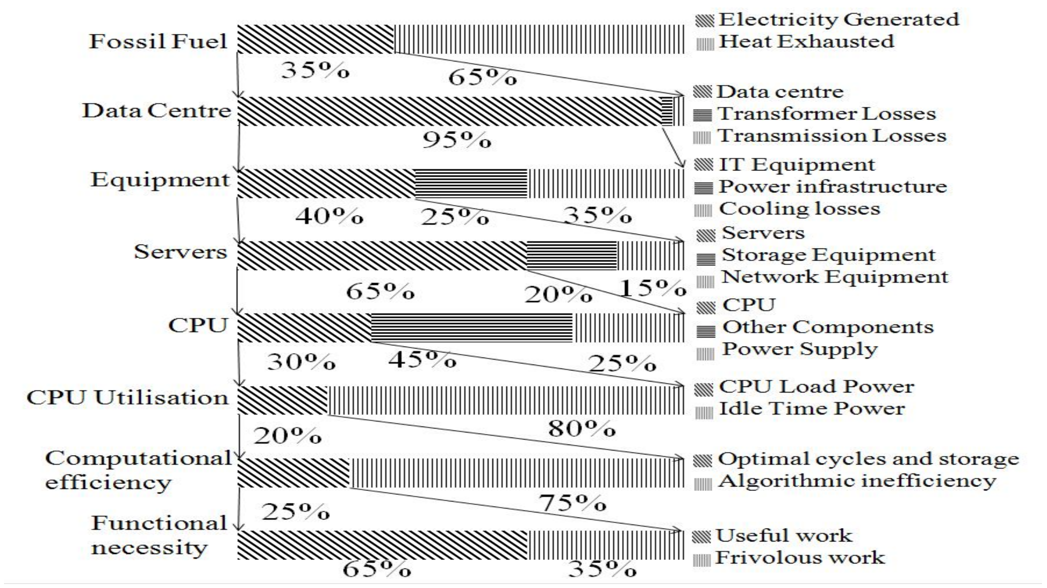

Figure 1. Power loss chain

information systems through the formation of a holistic approach to the creation of green software.

Main part

So, we want to offer a holistic approach to the development of green software that passes through its whole life cycle. Consider the life cycle of green software on figure 2. It consists of five stages.

At the first phase of the cycle we are developing the software requirements. They include the impact assessment requirements for the energy efficiency and eliminate frivolous requirements. The best tool for this is the "green" gap analysis [5].

A gap is sometimes called "the space between where we are and where we want to be."A gap analysis helps bridge that highlights which requirements are being met and which are not.

The next stage is the design stage. At this stage we assess the energy efficiency of the design decisions and consider a more profitable alternative.

An effective tool for the assessment of the software design is energy metrics. Their use makes it possible to predict the energy consumption in part at the design stage. In addition it is necessary to consider the software design methods. Article [6] shows the effectiveness of Agile methods at the design stage.

The building stage is the most complex and time-consuming.

It consists of four directions: computational efficiency, data efficiency, context awareness and idle efficiency.

The main idea of the computational efficiency is to complete the task quickly as possible. The sooner we complete the workload the faster we can return to the idle, and the more energy we will save. To obtain the necessary level of the computing performance it is necessary to use the software techniques such as multi-threading, vectorization, and of course efficient algorithms.

The algorithms and data structures present quite an extensive area of research in the field of energy efficiency. The selection of appropriate algorithms and data structures can lead to a huge difference in performance. And for each set of tasks such algorithms are unique. The use of algorithms largely depends on a thorough study of the problem, detailed examination of the application architecture. This choice depends on just increase productivity and decrease power.

Another approach used to achieve more computational efficiency is vectorizing the code. Instead of a scalar C-code, we use the advanced instructions such as SIMD (Single Instruction Multiple-Data) for implementation of instruction-level data parallelism. If the solution can be vectorized, we obtain the corresponding gains in productivity.

Multithreading will provide a better application performance and energy efficiency, respectively, as the work of a single thread takes much more time and energy than any of the multi-threaded runs.

Data efficiency reduces energy costs by minimizing data movement. Data efficiency can be achieved by using [7]:

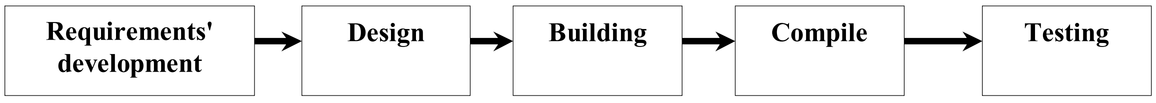

Figure 2. Green software life cycle

– software algorithms that minimize data movement;

– memory hierarchies that keep data close to processing elements;

– application software that efficiently uses cache memories.

The goal of context awareness is to create applications that can react or adapt to changes in the environment. The software can take advantage of the context-awareness to save energy for example by using power policies or AC/DC monitoring.

Power Policies provide a timely opportunity to change the power consumption depending on the software behavior, or change the software behavior, depending on the requirements for power consumption.

Similarly, monitoring of the AC/DC gives benefits in flexible adaptation of the software to the operating mode of the battery, thereby increasing the current battery life.

Another development of context-aware is monitoring the status of other components of the device, such as network cards, WI-FI, Bluetooth etc., and use this information for efficient software and save energy.

And the last direction - Idle efficiency It is based on the use of deep C-State residency (figure 3), timer resolutions and background activity.

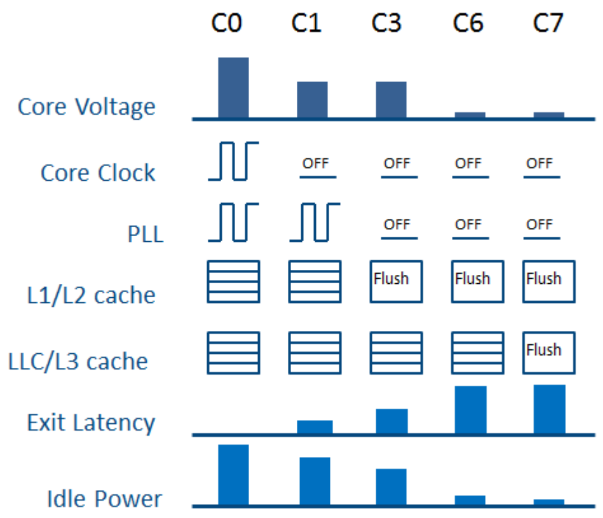

Figure 3. C-states

The power consumed when the system is in ACPI S0 state (S3- sleep or S4-Hibernate) with a running software, but with not active workloads, is called the Idle power. The background activity should be minimal in this state. The main objective is to reduce the idle floor by improving the software idle efficiency that will greatly increase the battery runtime. It will also give some benefits for various power scenarios even with the most demanding workloads.

To keep the platform in deep C-state as long as possible is one of the key requirements for idle efficiency. While the platform is in idle state, the level of deep C-state should compose at least 90%. The software should try to make the number of C-State transitions as low as possible. Frequent C-State transitions from deep to active state are not energy efficient. So to allow higher C-state residencies the activity should be coalesced where it is available.

This sort of frequent C-state transition can be implemented in two ways [8]:

– the energy requirements to enter/exit C-state are non-trivial. When the C0 (active) duration is very short, the latency to transition in and out of the C-states is appreciable and may result in net energy loss;

– the hardware policy may demote the C-state to a lower state based on heuristics. Even if the frequent C-state transition behavior occurs only for 2-3msec in a 15.6msec window, the hardware polices may either demote the core C-state or re-open the package level cache and this will impact the power for the remaining ~12-13msec of the idle period.

Reducing the C-state transitions in the software does not need division of the tasks among the processes / threads, if the parallel execution can occur. If there is a need to divide the task between the processes, the time schedule is constructed so that the amount of C-state transition may be reduced. The same can be mentioned in the connection with the increasing idle period residency, the software should coalesce the activity whenever possible.

Another approach to the energy efficiency is to change the system timer. On figure 4 we can see how the power consumption changes when the system timer is changed.

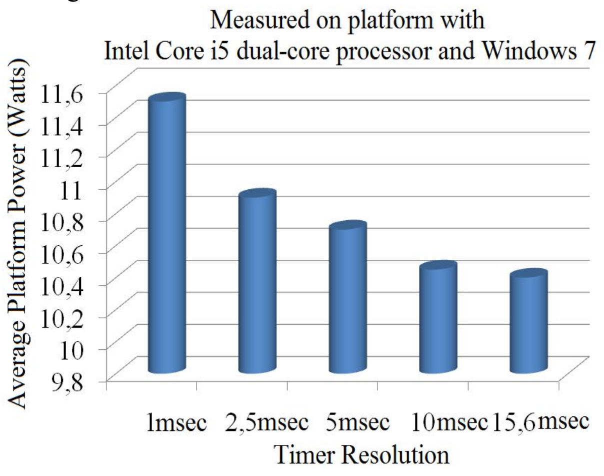

Figure 4. Power Impact of increasing Periodic Timer resolution

If the software must use a high-resolution periodic timer, one should use the periodic timer only while the required functionality is active. The same consider disabling use of the periodic timer and associated functionality when the system is running on battery power or when a Power Saver power plan is active.

Frequent changes of the background activity increases the power consumption. This loads both the processor and the chipset power. The system will also prevent idling to sleep by using long-term rare events. There are some ways to minimize frequent idle activity [8]:

– elimination of TCP DPC timer on every system timer interruption;

– reduction in frequency of USB driver maintenance timers;

– intelligent timer tick distribution;

– timer coalescing.

Compiling is the next stage of software life cycle. It includes two directions - to compile interpreted languages and using green compiler.

Most of the developed software using programming languages that are translated into machine code by interpreter at run-time, cannot be compiled into efficient machine code before deployment. Such interpreted code accordingly requires more processing power, therefore, requires more energy to do the same job. However, for some of these interpreted languages, such as PHP, compilation is still possible. If small changes are included to the code, it may be suitable for compilation, then it can be started more energy efficiently [9].

Compiler is a power source to optimize energy on the software level. The most known green compiler are:

– encc an energy aware C compiler;

– Coffee compiler for C language is combination of software and customized hardware to achieve energy conservation at compile time;

– mrcc is a distributed open source C compiler using Map Reduce on Hadoop platform;

– DGC is a hardware independent compiler that does not require any special hardware.

Energy aware compilers analyze software programs at run time and reshape the software source code by applying several green aspects during the code transformation. The following green techniques that can make the software more energy aware [10]:

– cache skipping;

– use of register operands;

– instruction clustering;

– instruction re-ordering and memory addressing;

– use of energy cost database;

– loop optimization;

– dynamic power management;

– resource hibernation;

– cloud aware task mapping;

– eliminate recursion.

And the last stage is testing. During this stage we monitor the energy consumption and provide feedback to the development. To determine the energy efficiency of the resulting software, we should first define a set of metrics for analysis. The paper [11] proposed the set of metrics characterizing energy consumption. To determine the metric data, we can use the existing software tools. We will review a few of them, because at the moment there are quite a lot of such software.

Intel Power Checker - perhaps the easiest and fastest way to evaluate the energy efficiency of the program [12].

Intel Battery Life analyzer - more complicated, but at the same time more informative tool used to track various hardware and software activities that affect the battery life [13].

Microsoft Joulemeter - also quite interesting tool that determines the power consumption of various components of the platform [14]. It can work in conjunction with a power meter WattsUp [15].

Summary

In the given paper the problem of improving the energy efficiency of the information systems has been considered. Despite the progress obtained in the field of hardware, the energy consumption of information systems continues to grow. To achieve the required level of energy savings one must look for new approaches. One of these approaches is the creation of a green software.

In the given work the existing methods of reducing the energy consumption by optimizing the software have been analyzed. A holistic approach to creating the energy efficiency software passing through the entire life cycle has been formulated. At each stage modern green methods, from the requirement development with green gap analysis to the use of the existing software tools for determining the energy efficiency of the final product, are used.

The proposed approach could be the basis for a new model of green software.

The software optimization techniques used in the approach affect the upper levels of the system design hierarchy. Therefore it can be concluded that the use of this approach will help achieve the desired level of energy efficiency of information systems.

Literature

1. Power estimation of embedded systems: A hardware/software codesign approach [Text] / W. Fornaciari, P. Gubian, D. Sciuto, C. Silvano // IEEE Trans. on VLSI Systems. - 1998. - Vol. 6/2. - P. 266-275.

2. Chung, E. Software Approaches for Energy-efficient System Design: Focused on Dynamic Power Management and Program Specialization [Text] /

E. Chung. - VDM Publishing, 2009. - 128 p.

3. Visser, J. Green Software [Electronic resource] / J. Visser. - Access mode: http://staff.science. uva.nl/~delaat/news/2012-03-23/slides_visser.pdf. – 25.11.2013.

4. Сидоров, Н. А. Зеленые информационные технологии и системы [Текст] / Н. А. Сидоров // Інженерія програмного забезпечення. – 2011. – № 3. – С. 5-12.

5. McDonald, M. F. Integrating green into an existing management system: Performing a "green" gap analysis [Text] / M. F. McDonald // Annual Quality Congress. - 2002. - Vol. 56, № 1. - P. 451-458.

6. Green software engineering with agile methods [Text] / M. Dick, J. Drangmeister, E. Kern, S. Naumann // 2nd International Workshop on Green and Sustainable Software (GREENS 2013), May 20, 2013 San Francisco, CA, USA. – 2013. – P. 78–85.

7. Steigerwald, B. Developing Green Software [Electronic resource] / B. Steigerwald, A. Agrawal. - Access mode: http://software.intel.com/sites/default/ files/m/0/6/7/8/1/37258-developing_green_software.pdf. – 25.11.2013.

8. Murugesan, S. Harnessing Green IT: Principles and Practices [Text] / S. Murugesan, G. R. Gangadharan. - Wiley, 2012. - 432 p.

9. Bicknell, D. 8 ways to make your software applications more energy efficient [Electronic resource] / D. Bicknell. - Access mode: http://www. computerweekly.com/blogs/greentech/. – 25.11.2013.

10. Software level green computing for large scale systems [Text] / F. Fakhar, B. Javed, R. Rasool, O. Malik, K. Zulfiqar// Journal of Cloud Computing: Advances, Systems and Applications. – 2012. – Vol. 1, № 4. – P. 1-17.

11. Юрченко, А. В. Проектирование и анализ программного обеспечения с низким энергопотреблением с помощью программных метрик энергоэффективности [Текст] / А. В. Юрченко // Инженерное образование. - Т. 1. - 2013. - С. 215-234.

12. Intel Power Checker [Electronic resource]. – Access mode: http://software.intel.com/en-us/blogs/ 2011/06/27/intel-power-checker. – 25.11.2013.

13. Intel Battery Life analyzer [Electronic resource]. – Access mode: https://downloadcenter.intel. com/Detail_Desc.aspx?agr=Y&DwnldID=19351. – 25.11.2013.

14. Microsoft Joulemeter [Electronic resource]. – Access mode: http://research.microsoft.com/en-us /projects/joulemeter/. – 25.11.2013.

15. WattsUp [Electronic resource]. – Access mode: https://www.wattsupmeters.com/secure/ index.php. – 25.11.2013.

Поступила в редакцию 25.11.2013, рассмотрена на редколлегии 12.02.2014

Рецензент: д-р техн. наук, професор, завідувач кафедри комп'ютерних систем та мереж В. С. Харченко, Національний аерокосмічний університет ім. М. Є. Жуковського «ХАІ», м. Харків, Україна.

ПОДХОД К ПОВЫШЕНИЮ ЭНЕРГОЭФФЕКТИВНОСТИ ИНФОРМАЦИОННЫХ СИСТЕМ

А. Ю. Кривцов, С. В. Гонтовой

В данной статье освещен вопрос повышения энергоэффективности информационных систем. Определены два основных направления - аппаратное и программное. Проанализированы существующие методы уменьшения энергопотребления на программном уровне. Предложен новый целостный подход к разработке энергоэффективного программного обеспечения, проходящий через весь его жизненный цикл. Предложенный подход состоит из пяти этапов, включающих требования, проектирование, построение, компилирование и тестирование. Для каждого этапа предложены алгоритмы, способы и методы повышения энергоэффективности программного обеспечения.

Ключевые слова: энергоэффективность, жизненный цикл программного обеспечения, зеленый компилятор, С-состояние.

ПІДХІД К ПІДВИЩЕННЮ ЕНЕРГОЕФЕКТИВНОСТІ ІНФОРМАЦІЙНИХ СИСТЕМ

А. Ю. Кривцов, С. В. Гонтовий

У даній статті висвітлено питання підвищення енергоефективності інформаційних систем. Визначено два основні напрями - апаратний і програмний. Проаналізовано існуючі методи зменшення енергоспоживання на програмному рівні. Запропоновано новий цілісний підхід до розробки енергоефективного програмного забезпечення, що проходить через весь його життєвий цикл. Запропонований підхід складається з п'яти етапів, що включають вимоги, проектування, побудову, компілювання і тестування. Для кожного етапу запропоновано алгоритми, способи і методи підвищення енергоефективності програмного забезпечення.

Ключові слова: енергоефективність, життєвий цикл програмного забезпечення, зелений компілятор, С-стан.

Кривцов Андрей Юрьевич – аспирант кафедры "Специализированные компьютерные системы", Донбасский государственный технический университет, г. Алчевск, Украина, e-mail: andrewdgtu@mail.ru.

Гонтовой Сергей Викторович – канд. техн. наук, заведующий кафедрой "Специализированные компьютерные системы", Донбасский государственный технический университет, г. Алчевск, Украина, e-mail: gsv@dmmi.edu.ua.