An Integrated Approach to Energy Consumption Optimization in Embedded Systems

Maciej Borzęcki, Technical University of Łódź

Abstract
The paper presents an approach to energy consumption optimization by use of a framework, which integrates development tools with the execution environment. A general concept of such approach is presented, as well as the plans form implementation in a real-world embedded system. The method involves enhancements to build time process, which is provided with energetic system model, as well as execution environment. The binary files are augmented with energy cost information, associating abstract cost value with blocks of code. The information is read by the operating system kernel at run time, upon startup of particular application. The application execution is monitored, and the statistical information about the execution is made available to user space. The collected data is analyzed and used for adjustments of both process and I/O scheduling. The framework is based on open source software, such as GNU compiler collection and the Linux kernel. The work is part of a project carried out at the Technical University of Łódź within Embedded Power interest group.

1. Introduction
Energy consumed by an embedded system, as a result of its components being active for particular period, is tightly coupled with the running software. The energy consumption optimization process, as referred to in this paper, is aimed at managing the software in such a way, that the total energy spent on executing given tasks is reduced.

Contemporary processors, available as discrete units or in form of tightly integrated System-on-Chip provide mechanisms for extensive control, and even a selective shutdown, of its internal blocks thus allowing for a fine tuned energy consumption optimization. Although, the well known relation between energy, chip clock frequency and supply voltage provided below is still valid.

\[ E \approx V^2 \times f \]

The effects of sole voltage or frequency reduction are suboptimal. A complete, system wide, energy consumption optimization mechanism should exploit the features of modern processors. At the same time, it should be possible for the executed applications to meet their deadlines.

The presented work is a part of a project carried out at Technical University of Łódź, aimed at building a framework to aid the software development process with tools for optimizing energy consumption.

The main work packages in the project are:
- define an energetic model of an embedded system
- develop algorithms for estimating energy consumption based on source code analysis
- develop scheduling and resource acquisition algorithm with focus on energy consumption and meeting task deadlines

The paper describes the general approach with focus on methods of extending an embedded Linux system with mechanisms allowing for distributing, loading and monitoring programs with associated energy-consumption-metadata.

Section 2 provides a brief overview of state of the art. Section 3 describes the methodology and the aspects of implementation, considering distribution of energy-metadata and run-time use of this information. Concluding remarks are provided in section 4.

2. Related work
There has been a number of contributions in the area, with most of the work focused on efficient energy and voltage control [8] or task scheduling [12]. Works such as [2] carry out optimization only on the CPU, not considering peripherals. The GNU gcc compiler was modified for the purpose of energy estimation in [2]. While another custom compiler was fed with the benchmark-based energetic model of the target processor in [3]. The results obtained in [3] show that use of the processor's energetic model does not only provide reasonably accurate results, but it also provides invaluable insight into energy consumption.
distribution. Attempt to build a closely tied framework such as described in this paper was undertaken in [9].

3. Methodology

The main goal is to develop a coherent framework for optimizing energy consumption, spanning from development tools to the runtime environment. The overall concept is presented in Fig. 1. Due to the availability of source code, and support for a significant number of architecture, Linux 2.6 kernel and gcc compiler, are use throughout the project.

Fig. 1. Overview of the proposed approach

The framework shall integrate estimation and monitoring of energy consumption into the development and execution processes. The inputs are application source code, energetic model of the target platform and run-time collected execution information. Based on the system model, the compiler shall estimate the energy consumption of the compiled code and create an energy map corresponding to particular program. The system model carries energy consumption information of both the main processor and accompanying peripherals. The characterization data is obtained through a benchmarking process, where basic code blocks, are executed a significant number of times, allowing the average energy consumption of particular operation to be estimated. The flow of the components is shown in Fig. 2.

Fig. 2. Flow of the proposed approach

The resulting energy map, as well as additional metadata containing application class (real-time, multimedia, system, I/O intensive, numerical, interactive) and scheduling hints shall be distributed with the program and used for run-time optimization purpose in the target system. The monitoring functionality shall collect the run-time information provided by the kernel for further analysis, and provide input to the system-wide scheduling unit, responsible for coordinating both process and I/O scheduling.

3.1 Energy map and program binary augmentation

The information about estimated energy consumption resulting from execution of a given program needs to be provided for the decision framework to carry out its task. The information can provided in a form of energy map. The concept is shown in Fig. 3. The map consists of elements which describe code blocks in form of a tuple: (block ID, start address, size, cost). The cost metric is an abstract value, specifying the expected energy cost of executing a given block of code, with higher values corresponding to higher cost.

Fig. 3. Energy map concept

For ease of distribution, the information should be attached to the program binary. The author proposes a mechanism of extending ELF[10], and augmenting the binary files with relevant energy consumption and additional application specific information. The approach is visualized in Fig. 4, where the ELF file is used as a container. In this case, the energy map information is stored within one of the user sections in the binary file.

Fig. 4. Storing energy map inside ELF binary

The Linux kernel needs to be modified in order to make use of this. The process of loading a binary file in the kernel discards all the information read from the file after the memory image of the process has been prepared for execution. The energy map information needs to be accessed at run-time, hence the kernel needs to store the information about the energy map and associate it with the kernel process descriptor as shown in Fig. 5.
For ease of execution monitoring described in Section 3.2, the section-relative addresses stored in the map should be translated to the actual virtual memory addresses.

The binary may contain additional meta-data, such as application class or extra scheduling hints, which is stored in specific data structures and made available for use within the kernel.

3.2 Execution sampling and scheduler enhancements

The use of energy map is closely tied to program execution. The general concept of the interaction of the kernel with energy map is shown in Fig. 6. The kernel shall supervise the program execution, and sample the executing code at context switch, thus identifying the currently executed instruction and it's location within the energy map.

3.3 Feedback loop

The statistical information obtained during run-time process execution monitoring shall be exported to user space. The `debugfs` file system was positively evaluated for use in implementation. The kernel drivers shall make the statistical data available for user access in a special location within the host file system. The user-space application for collecting the information can be run periodically to fetch a snapshot of current state, frequently enough for the temporal behavior to be observed.

The statistical information is provided in a format which directly matches the energy map description. Additional tools can be used for offline analysis.

This information combined with the energy consumption measurements will result in adjustments, primarily to current schedule, as the system wide scheduler will need to incorporate the information on I/O access and execution patterns.

It is expected that a the development environment shall allow for offline testing of the proposed schedule, by use of the snapshots of collected runtime information.
4. Conclusions

The author has presented a general concept of a framework for system wide energy consumption optimization. The proposed framework is composed of a customized compiler suite, extended Linux kernel and a number of minor tools, all used for estimating the energy consumption, monitoring of application execution, and collecting statistical information about the process behavior, resulting in an on-line scheduling adjustments.

The described work will be used for identification of important parameters of the energetic model of an embedded system, as well as testing of new energy cost estimation and scheduling algorithms.

The paper presents a snapshot of current work carried out at Technical University of Łódź within Embedded Power interest group.

References

2. D. Zhurikhin, A. Belevantsev, A. Avetisyan, K. Batuzov, S. Lee, Evaluating power-aware optimizations within GCC compiler, GCC Research Opportunities workshop, GROW’09, 2009

Author:
Mgr inż. Maciej Borzęcki
Technical University of Łódź
ul. Wólczańska 221/223 B18
90-924 Łódź
tel. (042) 631 26 49
email: mborzecki@dmcs.pl