NANOENERGY LETTERS

Whole Systems Energy Transparency

Kerstin Eder[1], Steve Kerrison[1], John Gallagher[23], Pedro López-García[34]
[1] University of Bristol, Department of Computer Science, Bristol, UK
[2] Roskilde University, Roskilde, Denmark
[3] IMDEA Software Institute, Madrid, Spain
[4] Spanish Council for Scientific Research (CSIC), Spain

***Abstract*—Energy efficiency is now a major (if not the major) concern in electronic systems engineering. While hardware can be designed to save a modest amount of energy, the potential for savings are far greater at the higher levels of abstraction in the system stack. The greatest savings are expected from energy consumption-aware software. This article emphasizes the importance of energy transparency from hardware to software as a foundation for energy-aware system design. Energy transparency enables a deeper understanding of how algorithms and coding impact on the energy consumption of a computation when executed on hardware. It is a key prerequisite for informed design space exploration and helps system designers to find the optimal tradeoff between performance, accuracy and energy consumption of a computation. Promoting energy efficiency to a first class software design goal is therefore an urgent research challenge. This article outlines our vision and first steps towards addressing this challenge.**

I. INTRODUCTION

ENERGY efficiency is now a major (if not the major) concern in electronic systems engineering. Research initiatives worldwide have a strong focus on making systems energy efficient. The EC's FET MINECC programme aims to "lay the foundations for radically new technologies for computation that strive for the theoretical limits in energy consumption while maintaining or even enhancing functionality and performance." The research objectives targeted in this context range from "new elementary devices and inter-device-communication mechanisms operating at the limits of minimum energy consumption", "novel computing paradigms with radically improved energy efficiency" such as "approaches inspired by biology, post-Boolean logics and computing under uncertainty, randomness and unreliability as a result of low-energy device properties", to "software models and programming methodologies supporting the strive for the energetic limit (e.g. energy cost awareness or exploiting the trade-off between energy and performance/precision)." This article addresses the last of these objectives; it is focused on energy transparency, which we regard to be a key prerequisite for new energy-efficient software models and software development methodologies.

II. ENERGY-AWARE COMPUTING

Energy-aware computing is a research challenge that requires investigating the entire "system stack" from application software and algorithms, via programming languages, compilers, instruction sets and micro architectures, down to the design and manufacture of the hardware, or alternatively bottom up. This is because energy is consumed by the hardware performing computations, but the control over the computation ultimately lies within the software and algorithms, i.e. the applications running on the hardware. It has recently become clear that, while hardware can be designed to save a modest amount of energy, the potential for savings are far greater at the higher levels of abstraction in the system stack. The greatest savings are expected from energy-consumption-aware software. Experts from Intel [1] estimate that energy-efficient software can realize savings that are three to five times of what conventional software is achieving. These savings can be made purely by software taking better control of the energy-saving features of hardware. Our research aims to realize exactly these savings - closing the gap that currently exists between software and hardware.

The importance of closing this gap has long been recognized as evidenced by the closing statements in an article published in 1997 on "software design for low power" [2], where the authors list five key objectives that "help making software design decisions consistent with the objectives of power minimization" as follows: "Choose the best algorithm for the problem at hand and make sure it fits well with the computational hardware. Failure to do this can lead to costs far exceeding the benefit of more localized power optimizations. Minimize memory size and expensive memory accesses through algorithm transformations, efficient mapping of data into memory, and optimal use of memory bandwidth, registers and cache. Optimize the performance of the application, making maximum use of available parallelism. Take advantage of hardware support for power management. Finally, select instructions, sequence them, and order operations in a way that minimizes switching in the CPU and datapath."

Addressing the challenge of energy-aware computing clearly requires collaboration between researchers vertically over several layers in the system stack, so that a deeper understanding of how energy is consumed during computation can be gained. The trend over the last decades has been to de-couple software engineering from the operation of the hardware; this now hinders progress when designing software for low-energy systems, for which a good fit of the algorithm to the hardware is critically important. As a consequence, much of the potential energy saving available from power-efficient hardware is wasted.

III. ENERGY TRANSPARENCY

Energy transparency from hardware to software is the central concept that lays the foundation for energy-aware system design. It aims to provide information on energy usage of programs without executing them, and at different levels of abstraction, from machine code to high-level application code. Energy transparency is fundamentally at odds with the natural trend in modern software engineering - the desire to abstract away machine-level details using high-level languages, abstract data types and classes, and sophisticated libraries that require layers of interpretation or compilation, in the interests of portability,

understandability and software reuse across hardware platforms. By contrast, energy transparency requires making visible how algorithms and their encoding in software impact on the energy consumption of a computation when executed on hardware. Availability of this information enables early design space exploration. It helps system designers to find the optimal tradeoff between performance, accuracy and energy usage of a computation. Energy transparency enables the promotion of energy efficiency to a first class software design goal.

To achieve energy transparency, models of how energy is consumed during a computation are required. Such models can in principle be established at different levels of abstraction, ranging from models that characterize individual functional hardware blocks (e.g. [3]), via Instruction Set Architecture (ISA) specific characterization models (e.g. [4], [5] and [6]), to models that aggregate hardware energy consumption values at intermediate representations used by the compiler tool chain (e.g. [7]). The process of establishing these models involves developing a representative set of test cases, executing these on a hardware test harness that includes a power consumption monitoring system, and attributing the measurements obtained to the respective model elements.

The final energy models, irrespective of their level of abstraction, provide information that feeds into static resource usage analysis algorithms, such as [8], where they represent the resource usage of elementary parts of the computation, e.g. program operations including library procedures, or usage of hardware blocks. Static resource usage analysis techniques can then be performed on a representation of a program that matches the one used in the energy model. The analysis traverses the representation and infers upper and lower bounds on the resource usage of the computation, expressed in terms of a function on several parameters of the input data (e.g., size), and the hardware platform where they are intended to be executed (e.g., clock frequency and voltage).

It is important that a good trade-off is found between the simplicity of the models, which improves the efficiency of the analysis, and the accuracy of the models, which improves the accuracy of the global analysis. Today, a large number of fundamental research questions need to be investigated, and numerous practical challenges need to be overcome, before energy transparency can become an integrated part in electronic system design. A future, where software developers have an intuitive understanding of how much energy is consumed by the programs they write, where instant feedback can be given to developers on how “hot” or “cold” their code is, is a vision that, in our opinion, is realizable. It forms the basis for energy-efficient computing from hardware to software, over the entire system stack, and enables system designers to “strive for the energetic limit.”

IV. ESTABLISHING THE LIMIT

The question of what constitutes the limits of energy consumption of computation is an important one generally, and, in the context of optimizing software for energy efficiency, particularly. Considerable research effort is being invested into finding the theoretical limits of energy consumption of computation, thus establishing the basic physical properties of information processing. To realize the considerable savings that are expected from energy-efficient software, determining the practical limits is no less important.

This necessitates investigating the minimum energy consumption that can be achieved for a given computation by optimal utilization of a target hardware platform. Finding this limit may require manual tuning of application code and manual configuration of the hardware; the remaining challenge is then to develop techniques that optimize and configure the system to reach this limit fully automatically. In practice, there are vast numbers of optimizations, different computer architectures, hardware configurations, e.g. in terms of memory/cache size, supply voltage and clock frequency, power management systems and manufacturing technologies with different physical properties, all influence the energy consumption during computation. In modern multi-core and distributed systems, the major cost of a computation is caused by data communication, not the computation itself. Fundamental research is needed to fully understand the impact of communication on the energy consumption of a computation, and to determine how to exploit this knowledge to save energy. For instance, in some cases, moving the program to the data is more energy efficient than moving the data to the program. Energy transparency aims to make all these effects visible; it is therefore a critical step to establish the practical limit and crucial to enable automatic optimizations for energy-efficient software, as well as dynamic scheduling of resources, so that energy usage, performance and accuracy are optimally balanced.

It is the vision of the Whole Systems ENergy TRAnsparency (ENTRA) project (http://entraproject.eu) to enable energy-efficient system design, especially energy-efficient software engineering, through resource usage analysis, verification and optimization, both during code development and at runtime, based on whole-system energy transparency.

ACKNOWLEDGEMENTS

The research leading to this article has received funding from the European Union 7th Framework Programme under grant agreement no 318337, ENTRA - Whole Systems Energy Transparency.

REFERENCES

[1] C. Edwards. Lack of Software Support marks the Low Power Scorecard at DAC. ElectronicsWeekly (http://www.electronicsweekly.com), No. 2472, 15-21 June 2011.

[2] K. Roy and M.C. Johnson. Software Design for Low Power. Chapter in *Low power design in deep submicron electronics*, W. Nebel and J. Mermet (Eds.). Kluwer Nato Advanced Science Institutes Series, Vol. 337. Kluwer Academic Publishers, Norwell, MA, USA, pp 433-460. 1997.

[3] D. Brooks, V. Tiwari, M. Martonosi. Wattch: a framework for architecture-level power analysis and optimizations. Proc. 27th International Symp. on Computer Architecture (ISCA), pp. 83-94, 2000.

[4] V. Tiwari, S. Malik and A. Wolfe. Instruction Level Power Analysis and Optimization of Software. Journal of VLSI Signal Processing Systems, 13, pp 223-238, 1996.

[5] S. Steinke, M. Knauer, L. Wehmeyer, and P. Marwedel. 2001. An accurate and fine grain instruction-level energy model supporting software optimizations. In Proc. of the Int. Workshop on Power & Timing Modeling, Optimization & Simulation (PATMOS), 2001.

[6] K. Eder and S. Kerrison. Energy modelling and optimisation of software for a hardware multi-threaded embedded microprocessor. Technical Report, University of Bristol, June 2013.

[7] C. Brandolese, W. Fornaciari. Software Energy Optimization Through Fine-Grained Function-Level Voltage and Frequency Scaling. International Conference on Hardware/Software Co-design and System Synthesis, pp 539-546, New York, USA, 2012.

[8] J. Navas, E. Mera, P. López-García, and M. Hermenegildo. User-Definable Resource Bounds Analysis for Logic Programs. In International Conference on Logic Programming (ICLP'07), LNCS. Springer, 2007.