Toward Energy Profiling of Connected Embedded Systems

Nadir Cherifi, Gilles Grimaud, Alexandre Boe, Thomas Vantroys

Toward Energy Profiling of Connected Embedded Systems

Nadir Cherifi
Worldline e-payment services
IRCICA/Cristal, CNRS UMR 9189,
Univ. Lille 1, France
Email: nadir.cherifi@worldline.com

Gilles Grimaud
Thomas Vantroys
IRCICA/Cristal, CNRS UMR 9189,
Univ. Lille 1, France
Email: gilles.grimaud@univ-lille1.fr
thomas.vantroys@univ-lille1.fr

Alexandre Boe
IRCICA/IEMN, CNRS UMR 8520,
Univ. Lille 1, France
Email: alexandre.boe@univ-lille1.fr

***Abstract*—A huge number of connected objects are expected to be deployed over the coming years in various areas of everyday life. Many of these objects are energy-constrained and depend on a battery. Thus, energy is a critical resource that limits a large scale deployment, and greatly complicates the development of the embedded software on these objects. Hence, the ability to measure and finely profile the power consumption of such devices, and correlate it with the on-board application is a big challenge to improve the software development. Furthermore, common energy patterns can be extracted from the collected energy figures in order to provide guidelines allowing a proactive energy-based development.**

In this paper, we present an ongoing work about a lightweight framework for energy profiling of embedded applications source code at a functional granularity. It is driven by an on-line hardware-based measurement technique permitting to gather accurate energy figures. The framework is integrated into an energy-centric iterative development cycle allowing fast revaluations of the energy consumed by the targeted functions after each source code modification. Afterwards, we describe our future works about overcoming an unlocked state of art issue relative to asynchronous energy consumption profiling.

I. Introduction

The Internet of Things (IoT) is a rising idea where the surrounding objects of the physical world are directly connected to the traditional Internet via wireless or wired connections. Todays life is more and more augmented with various connected objects in order to ease common tasks and provide extra informations to the users. To fully achieve this objective and allow the objects to perform their basic functions (sense, process, actuation, communication), a kind of complex intelligence has to be embed on the object in form of an onboard application.

Focusing on connected battery-operated devices, we can find today, multiple IoT applications where the replacement or the recharging of the battery is binding, difficult or even impossible regarding for example buried objects [1]. Moreover, the global energy consumption caused by a high concentration of connected objects in a building can be significant [2].

From the above statements, it appears that the design of energy efficient embedded software requires to have a fine grained energy consumption profile in order to detect software energy hot spots and fix them. we propose to help the embedded developer to achieve this goal. Taking for example the case of an energy-constrained embedded application (e.g. buried connected sensor, connected smart meter), the developer will be able to obtain accurate energy figures about the application source code functions. Through a visual feedback, this latter could make adequate inner modifications to the application and quickly re-evaluate the impacts on energy consumption. Our proposition make this process easier and totally transparent regarding the developer.

The contribution of this article is the proposition of an energy profiling framework for energy constrained embedded systems. The proposed framework is based on a lightweight source code instrumentation of the targeted embedded application, and a hardware-based measurement platform allowing to get real energy figures during the effective execution of the embedded application. An energy-centric development cycle is put forward to support the framework. It permits to ease and attenuate the cumbersome of the energy measurement and optimisation task.

This paper is organized as follows. In Section 2, we present some related works regarding energy profiling methods for embedded systems. We describe in Section 3, the composition and inner working of our framework. Finally, we conclude in Section 4 this article by enumerate various possible future improvements to our proposed method.

II. Related Works

Energy consumption of small connected objects is difficult to measure [3]. We can globally divide measurement methods into three distinct categories: hardware-based, software-based and simulation-based. We built our approach upon an on-line hardware-based technique which allows to acquire real energy figures of the embedded system on execution by taking in account real environment effects (e.g. radio noises). Furthermore, multiple studies describe and demonstrate that the gap in energy estimation between a hardware-based approach and a simulation-based one could be significant [4]. We can find in the literature, several related works that have chosen a similar way.

Acknowledgements: This work has been partially supported by Worldline e-payment services company.

In [3] authors describes ICount, a small and inexpensive design able to add energy metering with no hardware overhead to the target board to profile. This provides in addition the board with the possibility of measuring directly *in-situ* its own energy consumption. As a continuation, the authors present the software energy profiler Quanto [5] which leverages the on-board energy meter ICount. It instruments the underlying embedded device drivers to track the hardware components behaviours and account for their states and the activities to which they contribute. Despite the obvious advantages of the ICount on-boards energy metering module, its low accuracy makes it unsuitable for fine-grained energy consumption profiling. Lim and all. propose in [6], FlockLab, a testbed designed for a network of connected objects to perform a distributed and a synchronized tracing. The solution proposes a hardware/software platform aiming to facilitate a distributed timestamped debugging by levreaging the availability of GPIO pins on most of existing platforms. Like FlockLab, Powerbench proposed in [7] is also a testbed mostly oriented toward the Wireless Sensor Network (WSN) domain. It consists in a bench which allow up to 24 objects to be energetically measured in parallel. The distributed energy measurement is a great feature, especially for the WSN case, allowing to detect some synchronous energy bugs. However, these approaches lack in providing any source code energy estimation at any level of granularity. Finally, Honig and all. introduce in [8] the PEEK framework, a *proactive energy-aware programming* system approach designed in a developer centric fashion. Its architecture is articulated around a snapshot-based workflow infrastructure which is supervised by a middle-end component where the Git revision control system is used. The hardware part is based on a low cost lightweight measurement platform which uses multiple capacitors driving a RS flip-flop. Finally, the proposed framework provide the developer with few source code energy optimization hints. Despite obvious benefits, The energy hints mechanisms proposed by PEEK are too rudimentary. In addition, the local function energy profiling used may represent a burden for a developer who want a global and a large profiling of his embedded application.

III. Energy Profiling Proposition

The ability of measuring precisely the energy consumption of a system, and relate it to particular parts of an embedded program is a critical issue in most of embedded development cases. As a solution, we propose an energy-centric development methodology exposing an energy profiling framework which leverages an on-line hardware-based measurement technique. This latter framework is capable to assist the embedded developer aiming to energetically profile its embedded application source code.

A. Energy-Centric Development Cycle

For enhancing the embedded development in an energy-constrained environment, we think strongly that not only a good measurement technique is needed, but the entire development have to be centred on the energetic issue. The proposed energy development cycle, illustrated in Figure 1, is focused on improving the energy-based development. For this purpose, the *software development phase* and the *measurement phase* are placed at the center of the cycle. They represent the main and critical steps upon which the entire cycle, and so the optimisation phases, is based on.

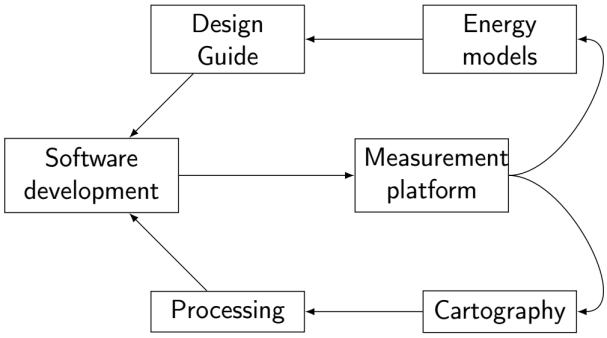

Fig. 1: Energy-based iterative development cycle, aiming to enhance and ease the embedded development for energy constrained applications.

Going deeper into the development cycle architecture, we can decompose it into two parts. The first (the bottom part), where an energy cartography of the source code is performed, and a visual feedback is presented to the developer. Following these processes, the developer will have in hand inputs permitting him to modify his software consequently, and then, replaying a new energy measurement cycle. These processes are totally illustrated in Figure 2 and will be explained with more details in the next subsections.

The second part of the development cycle (the upper part) is always in development, and refers to an energy model inferring process. These energy models reflect energy behaviours of multiple part of the targeted embedded system. Furthermore, by gathering and processing the produced models, we can theoretically design specific energy development guides regarding the embedded system hardware platform and the executed application on it. Energy model creation is a hard and long task. The creation phase itself requires a long period of energy samples acquisition, where the second phase represents the validation of the energy model by proving its accuracy in real environment conditions. In order to answer to these requirements, we leverage our energy framework to perform various energy measurement passes, and collect multiple energy samples of the embedded system in real execution. These samples are then processed using statistical tools and time series features to extract relevant informations about the energy behaviour of the embedded system. Knowing that I/O operations, and then peripherals, are the most energy consuming parts of an embedded system platform, we intend to use our model inferring process to build peripheral specific energy models. These latter must be able to resume the energy behaviour of each peripheral, and particularly, the energy states and states transitions highlighted by the acquired energy samples.

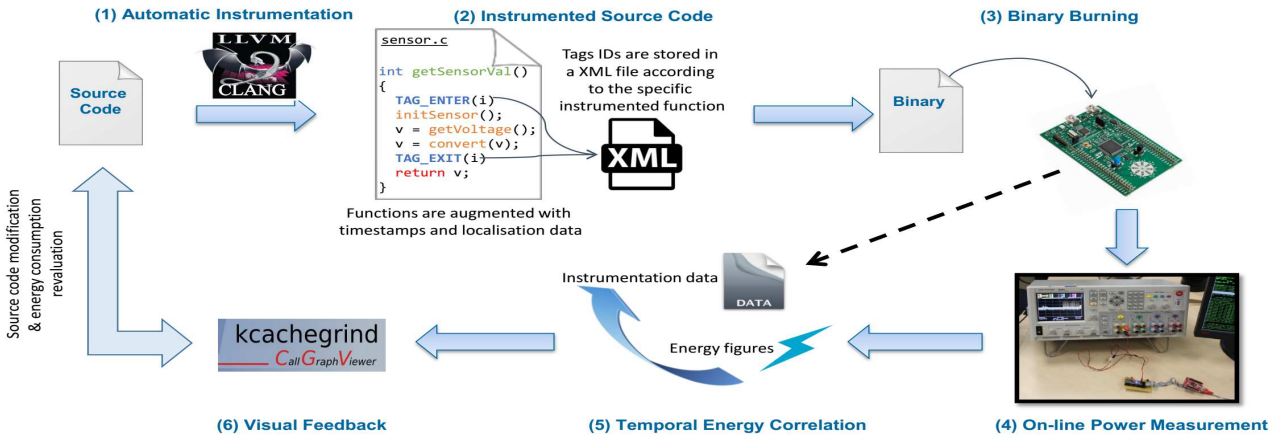

Fig. 2: The energy profiling cycle.

On a technical side, the major part of the framework is developed in Python, the instrumentation process is developed in C/C++, and the statistical processes are implemented using the R environment which communicate seamlessly with the rest of the framework using the Python rpy2 package.

B. Automatic Source Code Instrumentation

The source code instrumentation is a critical block of our development cycle and framework. Having only energy traces of the embedded application is not sufficient to deduce the functions that cause the energy consumption. The code instrumentation is then introduced and used to get logical information from the application on execution. One of the major goal of our framework is to ease and optimise the energy profiling phase. In this aim, we decide to automatise the entire instrumentation procedure. We leverage the LLVM-CLANG framework to build a third-party application responsible of automatically instrumenting the embedded application source code (Fig. 2.1). It inserts specific source tags on functions entry and exit. The functions to instrument are specified by the developer, which can also use the framework to perform a more global and spread instrumentation. Hence, the framework is able to recursively search and instrument all the called functions, by the ones specified by the developer, on the control flow graph (CFG).

When a function is instrumented, two tags are inserted at the entry and at the exit (Fig. 2.2). Every tag are only responsible of getting: the current time (timestamps) from a hardware counter, the address of the current function, and the address of the calling function. The instrumentation data resulting from the inserted tags are saved directly in the embedded system memory (i.e. RAM), and are sent out via a serial interface at a high baud rate. Among these data, The hardware counter timestamps represent a critical part. Indeed, we use the saved timestamps to locate on the output energy traces, the instrumented functions entries and exits (Fig. 2.5). The measurement task, and the hardware counter on the embedded board are synchronized at the beginning of the experiment by a simple digital GPIO. Finally, using the gathered instrumentation data, we can easily compute the energy consumption of every instrumented function by retrieving at the same time a partial dynamic CFG.

Every additional source code induces an overhead. However, in our case, the inserted tags lead to a really low perturbation of the original behaviour of the embedded application. In fact, every tag execution consume less than 15 cycles. The only issue concerns the serial buffer transfer which consumes a little amount of overhead energy. Nevertheless, knowing the start time and duration of every transfer, we can simply deduct the overhead energy from the measured energy consumption.

C. Hardware Measurement Board

On a measurement side, the proposed framework is supported by a hardware measurement approach, as quoted in Section 2. Until now, we use as a measurement platform a laboratory power analyser, the Agilent N6705A (Fig. 2.4). It features a high accuracy, a high resolution and a maximum frequency sampling of 50KHz. It supports a large range of voltage and current configurations which are sufficient to cover most of embedded devices power requirements. The Agilent device exposes 4 digital outputs allowing to measure up to 4 distinct power loads. This permit to deeply profile energy consumption of an embedded system by measuring separately its external peripherals. These benefits contrast unfortunately with a high cost neighbouring 6K€ which can represent a barrier for many developers and organisations. To remove this fence, we are working, in an advanced stage, on a low cost measurement platform based on the STM32F746NG Discovery board. We take profit of the several multichannel DMA controllers integrated on the board to link in an effective manner the High frequency ADCs with the Ethernet interface. We enhance the designed measurement software by adding the LwIP network stack to control easily the Ethernet interface. The FreeRTOS real time operating system is also used to address with a higher flexibility, in a real time scheme, the tasks of the measurement board software.

D. Energy Visual Feedback

After correlating synchronously the energy traces with the instrumentation data generated by the embedded platform, the framework is able to present energy figures results in a raw fashion. It attaches to each instrumented function entity all its occurrences, and for each occurrence, the energy consumption value and the caller function. This allows, in a certain way, to compute a dynamic energy control flow graph (eCFG). For helping the developer to better understand the energy consumption of his embedded application, we choose to add two graphical feedbacks. The first represents simply the power curve of the experimentation dynamically and graphically timestamped with the instrumented functions entries and exits. The second feedback refers to a desire of presenting a more concise form of the energy raw results cited before. Thus, we get advantage of the open-source profile visualizer, KCachegrind (Fig. 2.6), to get in shape the raw figures and show to the developer a clear and schematic vision of the eCFG. The initial capabilities of KCacheGrind allow us to assign different color captions according to the energy value of a function, thereby, permitting the developer to easily and quickly find energy hotspots or bugs in its application code.

E. Incremental Energy Instrumentation

The instrumentation performed by the framework is a valuable proposition to get logical data allowing to correlate the energy consumption with the executed embedded application source code. However, the more the number of instrumented functions is high, the greater the overhead caused by this instrumentation is sizeable. To resolve this issue, we introduce an automatic incremental instrumentation feature. The basic idea is to claim that functions which are associated with a highly variable power behaviour on the output energy trace, are functions that possibly perform multiple different treatments. Thus, instrumenting the called sub-functions of these latter is more profitable for learning more about the energy behaviour of the application than deeply instrumenting function characterised by a stable power behaviour.

To achieve this entire process, we use a multiple changepoints detection algorithm implemented on the statistical R framework within the changepoint package. The algorithm is able to detect accurately, and in a small time (i.e. 3 seconds for a 700.000 points time series) all the changepoints happened inside the time series. If an instrumented function power curve features a number of changepoints higher than a certain threshold, the called sub-functions are instrumented in its turns and a new measurement phase is engaged.

IV. Conclusion and Future Works

In this paper we present a novel embedded development cycle centred on the energy consumption of the embedded application. We describe a hardware/software energy profiling framework based on an automatic source code instrumentation, and a real-time hardware measurement scheme. Globally, the work done enable to ease, speed up, and optimize the energy profiling phase of an embedded development. It was applied and experimented on multiple embedded application giving accurate results. However, our energy profiling proposition is, until now, only accurate with synchronous energy consumption profiling. In other words, when the energy is consumed synchronously to the effective execution of the function responsible of it. Unfortunately, regarding several external peripherals, the energy consumption can be asynchronous unlike that of the processor for example.

Our main future work is then to overcome the asynchronous energy profiling lock. We intend to take advantage of the model inferring process of our framework (Fig. 1), and the fact that every embedded peripheral proposes a specific lightweight function API. Using multiple API function combinations and energetically profiling their executions, we can infer an energy automaton which summarize each peripheral energy behaviour according to the executed functions. To lighten and ease this process we'll base on the statical features of our framework to automatically treat the energy output figures. It uses a time series changepoint detection algorithm to locate the existing energy patterns. Afterwards, a hierarchical clustering based on a Bayesian classifier is performed to group the same energy patterns in different clusters forming then energy states.

The asynchronous profiling problem can be resoled using the previously cited energy models. When energetically profile the embedded system, the external peripherals activities are tracked thanks to the code instrumentation process. A correlation and a correspondence is done with the according energy model (i.e. automaton) to retrieve the supposed produced energy state. After ensuring the presence of this state on the current measurement energy output, the function responsible of the asynchronous consumption is accounted by the specific energy amount.

References

[1] K. M. Z. Shams and M. Ali, "Wireless power transmission to a buried sensor in concrete," *IEEE Sensors Journal*, Dec 2007.

[2] I. E. Agency, "More data, less energy: Making network standby more efficient in billions of connected devices," 2014.

[3] X. Jiang, P. Dutta, D. Culler, and I. Stoica, "Micro power meter for energy monitoring of wireless sensor networks at scale," in *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*. New York, NY, USA: ACM, 2007.

[4] N. Cherifi, G. Grimaud, T. Vantroys, and A. Boe, "Energy consumption of networked embedded systems," in *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, Aug 2015.

[5] R. Fonseca, P. Dutta, P. Levis, and I. Stoica, "Quanto: Tracking energy in networked embedded systems," in *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2008.

[6] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel, "Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems," in *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*. New York, NY, USA: ACM, 2013.

[7] I. Haratcherev, G. Halkes, T. Parker, O. Visser, and K. Langendoen, "PowerBench: A scalable testbed infrastructure for benchmarking power consumption," in *Int. Workshop on Sensor Network Engineering (IWSNE)*, Santorini Island, Greece, jun 2008.

[8] T. Hnig, H. Janker, C. Eibel, W. Schrder-Preikschat, O. Mihelic, and R. Kapitza, "Proactive Energy-aware Programming with PEEK," in *Proceedings of the 2014 International Conference on Timely Results in Operating Systems*. Berkeley, CA, USA: USENIX Association, 2014.