EProf: An Energy Profiler for the iPAQ

Kelly Koskelin, Kenneth Barr, Krste Asanović

Introduction: As the use of portable electronic devices has become increasingly widespread, efforts to extend battery life have become more crucial. Much work has gone toward improving the efficiency of hardware components and recently more emphasis has been placed on software techniques that help conserve energy. Energy profilers help programmers determine what parts of their code are most energy-intensive by correlating energy use with functions or instructions within a program. Being able to see where energy is consumed allows programmers to concentrate on the software hotspots.

Approach: EProf is an energy profiler for the iPAQ. It will gather data using statistical sampling and is designed to be unobtrusive to the user. EProf differs from previous energy profilers in that it is intended for use outside of a laboratory environment. First we describe previous work on statistical sampling for energy profiling and then we describe our system. Next we discuss how EProf differs from other profilers and the design decisions involved in building a profiler that a user can easily carry.

The use of statistical sampling techniques for energy profiling borrows from work done on cycle profilers [1]. In a cycle profiler the profiling system periodically interrupts the processor to record the executing instruction. The system keeps track of how many times each instruction is interrupted, thus building a statistical profile of where the processor spends its time. In an energy profiler the instantaneous power is recorded along with information about the instruction that was interrupted. This might include an identification of the process running, the software module it belonged to, and the address of the actual instruction [2]. This data is later processed to create a statistical profile of the amount of energy used by various processes.

PowerScope takes a time-driven approach to sampling. When sampling is enabled, the system periodically samples the current drawn by the system while simultaneously recording the program counter, process identifier, and other information [3]. F. Chang et al. use energy counts to decide when to take a sample [2]. When a given quanta of energy is consumed, an interrupt is issued and the instruction in progress is recorded and time-stamped. One advantage of the energy-driven approach is that when a lot of energy is being used, a series of interrupts will be triggered in quick succession. This allows analysis tools to pinpoint the expensive processes and catch energy peaks that might be missed in the time driven scheme. Also, when the system is in an idle state using little power, fewer samples will be taken, saving storage space and reducing the amount by which the system is disturbed.

Progress and Future Work:

The hardware component of the EProf system is used to measure power and trigger sampling in the iPAQ. The software portion consists of a profiler that records samples and offline tools for data analysis.

EProf is implemented on a Compaq iPAQ H3600. The power-sampling subsystem consists of a sense resistor, two analog to digital converters (ADC), and a Field Programmable Gate Array (FPGA) as shown in Figure 1. The sense resistor is placed between the battery and the iPAQ. One of the ADCs samples the voltage drop across the resistor, from which we can determine the current being drawn by the iPAQ. The other ADC is used to measure the voltage being supplied by the iPAQ's battery. Because instantaneous power is equal to current times voltage, the above information is sufficient to determine the power being drawn at the time of the sample. The FPGA controls the ADCs and passes their samples to the iPAQ for processing. The FPGA communicates with the iPAQ via a PCMCIA port on the iPAQ's backPAQ.

The daemon will allow the user to set the sampling rate of the system and start or stop sampling based on system events. The user can also set the sampling rate so that it is randomly jittered around a given rate. This helps guard against samples that are synchronized with system events.

The ability to start or stop sampling based on system events means that we can profile the energy associated with particular processes, functions, or other events that happen throughout the day. We can also sample continuously all day long. Offline analysis tools subsequently put the gathered data into an easily readable format.

The major design issues we face are related to the fact that EProf is intended for use outside a lab. This means that the system must be unobtrusive to the user, low power, and able to handle a large quantity of data.

EProf will be used on iPAQs running the Familiar distribution of Linux. The software subsystem will consist of a kernel module that records sample data, a user-level daemon to control sampling, and offline tools to analyze the data. The kernel module is a slight modification of a cycle profiler that already exists for the iPAQ. It provides the API that the user-level daemon uses to start sampling, stop sampling, and control sampling parameters.

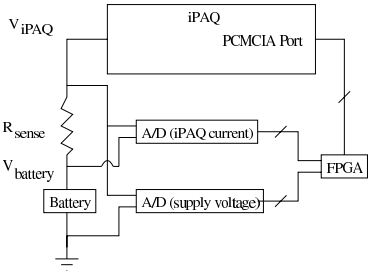

Figure 1: Hardware System Overview

The hardware is designed with size in mind. The system takes up about 3 square inches on a prototyping board and the additional hardware for the energy-driven profiler will at most double that space. The final design should fit within the confines of a Type II PCMCIA card so the user should not have to change her usage patterns.

We minimize the processing and energy overhead of the profiler in a number of ways. Most data processing will be done offline and we keep the number of transactions on the PCMCIA bus to a minimum. We keep the hardware portion low power by carefully choosing low-power components. Currently the power source for the hardware subsystem is the PCMCIA socket's power lines, but eventually the hardware subsystem will have a separate power supply.

We are taking a number of steps to address the large quantity of data gathered throughout the day. We are investigating the tradeoffs between storage space and processing time involved in converting the voltage samples to power samples offline. We make careful choice of the triggers that start and stop sampling to avoid filling space with redundant or unwanted samples. We will use compression and perhaps add memory to the hardware subsystem to hold more data. We may periodically upload the data to a central server.

We are currently working on measuring the energy that the battery supplies to the main system. Soon we will measure the energy drawn by the screen and backPAQ separately to see how the power drain varies throughout the system. This will require very little extra hardware.

The EProf prototype we are currently working on uses time-driven statistical sampling. In the future we will add energy-driven sampling to the system in order to compare the two approaches. The type of sampling will be selectable by the user at run-time. This will allow us to compare the accuracy of the two approaches for our application.

Soon, EProf will be distribute to a variety of users and we will map out energy use over a day under the varied conditions. Real-time processing can be added to allow users to see energy consumption as it happens. We will explore whether this allows users to adjust their use according to how much energy is drawn. With the large quantity of data gathered we will evaluate the energy needs of various applications.

Research Support: This work is supported by MIT Project Oxygen, NSF CAREER award CCR-0093354, and an equipment grant from Intel.

References:

[1] J. Anderson, L. Berc, J. Dean, S. Ghemawat, M. Henzinger, S. Leung, D. M. Vandevoorde, C. Waldspurger, and W. Weihl, "Continuous profiling: Where have all the cycles gone," in *Proceedings of the 16th Symposium on Operating Systems Principles*, Oct. 1997.

[2] F. Chang, K. Farkas, and P. Ranganathan, "Energy-driven statistical profiling: Detecting software hotspots," in *Proceedings of the Workshop on Power-Aware Computer Systems*, Feb. 2002.

[3] J. Flinn and M. Satyanarayanan, "Powerscope: A tool for profiling the energy usage of mobile applications," in *Proceedings of the Workshop on Mobile Computing Systems and Applications*, Feb. 1999.