ePRO-MP: energy PRofiler and Optimizer for MultiProcessors

Wonil Choi, Hyunhee Kim, Wook Song, Jiseok Song, and Jihong Kim

{choi11, hh0726, answer03, danielsong, jihong}@davinci.snu.ac.kr

School of Computer Science and Engineering, Seoul National University

http://davinci.snu.ac.kr/

Abstract

We present a software development tool, ePRO-MP, for profiling and optimizing energy and performance of mobile multithreaded applications. We demonstrate ePRO-MP using two optimization problems.

1. Introduction

For mobile multiprocessor applications, achieving both high performance and low energy consumption becomes a challenging task. In order to meet these design requirements, programmers should understand the performance and the energy consumption of their applications, thus making system development tools play an important role. Furthermore, automatic optimization support is becoming more important in multiprocessor-based embedded systems because an efficient implementation often requires exploring a large design space as the number of cores increases and the co-running threads share the limited system resources.

In this paper, we describe ePRO-MP which profiles and optimizes the performance and energy consumption of multi-threaded mobile applications. One unique feature of ePRO-MP is that it profiles the energy consumption of a target application without using an extra power measurement equipment. The proposed ePRO-MP also helps programmers to improve the performance and energy consumption of embedded design problems without programmers' intervention. In the experiments described later, we show that ePRO-MP can improve the performance and energy consumption by 6.1% and 4.1%, respectively, by optimizing the number of threads assigned for two co-running applications. We also demonstrate that ePRO-MP can improve the performance and energy consumption by 60.5% and 43.3%, respectively, over a baseline version for the producer-consumer application.

2. Overview of ePRO-MP

Figure 1 shows an overall architecture of ePRO-MP which consists of a target system and a host system. In this paper, we use ARM11 MPCore [3] where four ARM11 cores are integrated on a single chip as a target system and each of the cores supports various types of hardware performance counters. We also use a Pentium-4 desktop PC as a host system. For performance monitoring, ePRO-MP estimates various performance metrics such as the cache miss rate and IPC by using the hardware performance counters of each core.

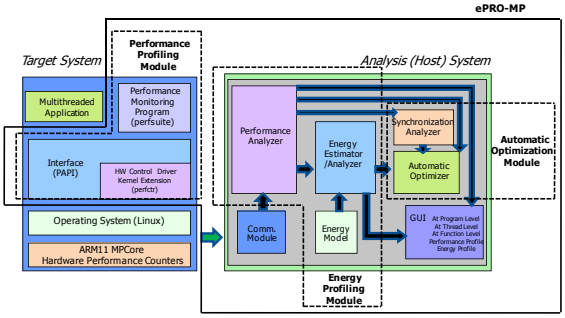


Figure 1: An Overview of ePRO-MP

In the target system, three logical modules, a target application, an operating system, and a performance profiling module, are running. The target application which will be profiled becomes multi-threaded parallel programs using the POSIX thread library. For an OS, we use Linux 2.6 for ARM11 MPCore. The performance profiling module collects performance data during runtime.

On the other hand, the host system consists of five main modules, communication module, performance and energy analyzer modules, graphical user interface module, and automatic optimization module. The communication module transfers performance profiling data from the target system to the host system. The collected performance data is analyzed by two analyzers, performance analyzer and energy analyzer. The performance analyzer classifies and arranges the performance profiling results while the energy analyzer applies the energy model to the performance profiling data to estimate the energy consumption. The analysis results are presented in multiple levels. The current version of ePRO-MP adopts the GUI of the Eclipse platform. Finally, the automatic optimization module uses the feedback from the profiling results for various optimizations.

3. Energy Profiling Module

ePRO-MP employs the regression-based modeling approach for energy profiling. Our methodology for developing the energy model consists of four main steps. In *Random Program Generation* step, we build an energy model that can accurately predict the energy consumption of an arbitrary program based on hardware performance counters and generate various random test programs with different execution characteristics. In *Automatic Run of Random Programs* step, training data for regression analysis are produced by executing the random programs

generated by the random program generator. At the same time, the energy consumption value is gathered from power measurement equipment. Analysis is applied to the training data gathered in the previous step in *Model Generation Using Regression Analysis Regression* step. Finally, in *Verification* step, we verify our power model using several benchmark programs as well as the random test programs used in the model generation step. We showed that an average error was about 3%.

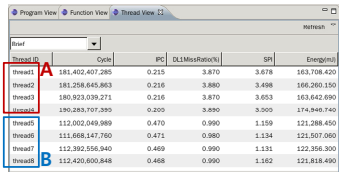
For the current energy model, five performance events, the number of instructions (*Instr*), the number of L1 data cache accesses (*DL1Access*), the number of L2 cache accesses (*L2Access*), the number of stall cycles due to data dependency (*DataDep*), and the number of coherence transactions (*cohTrans*), are selected. The power model for ARM11 MPCore is given as follows:

$$Power = A \times (Instr / time) + B \times (DL1Access / time) + C \times (L2Access / time) + D \times (DataDep / time) + E \times (CohTrans / time) + F_{const}$$

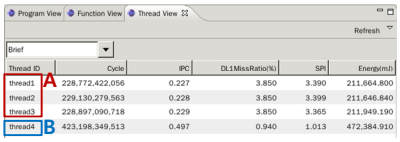
4. Profile-Based Automatic Optimizer

Based on performance and energy profile results, we propose profile-based automatic optimizer. In the current implementation, ePRO-MP's automatic optimization function optimizer tries to find the optimal number of threads for *co-running applications* and *producer-consumer applications* using a simple heuristic.

Figure 2 shows the result of the automatic optimization of co-running applications, Matrix-Multiplication (MM) and Insert-Sort (IS). Over the baseline (4, 4) configuration as shown in Figure 2(a), the (3, 1) thread configuration for MM and IS improves the total execution time by 6.1% and reduces the total energy consumption by 4.1% as shown in Figure 2(b). The performance and the energy consumption results of producer-consumer applications are shown Figure 3 where Matrix-Multiplication (MM) is used for producer and Matrix-Transpose (MT) is used for consumer. The results are normalized to the baseline (2, 2) thread allocation. Exploring thread allocation problem space with varying tile size, the (4, 1) thread configuration for MM and MT improves the total execution time by 60.5% and reduces the energy consumption by 43.3%.



(a) Profiling Result of (4, 4) Thread Allocation for (MM, IS)



(b) Profiling Result of (3, 1) Thread Allocation for (MM, IS)

Figure 2: Optimization Results of MM-IS Application

Figure 3: Optimization Results of MM-MT Application

5. Conclusion

We described ePRO-MP, an energy and performance profiler and optimizer for embedded multiprocessors. ePRO-MP provides both energy profiling information and performance profiling information that can be important in developing high-performance and low-energy embedded multi-threaded applications without power measurement equipment. Experimental results show that we can improve the performance and the energy consumption over the baseline thread allocation by 6.1% and 4.1%, respectively. For producer-consumer application, the execution time and the energy consumption were reduced by 60.5% and 43.3%, respectively.

6. References

[1] W. Baek, Y. Kim, and J. Kim, "ePRO: A Tool for Energy and Performance Profiler for Embedded Applications," In *International SoC Design Conference*, 2004.

[2] G. Contreras and M. Martonosi, "Power Prediction for Intel XScale Processors Using Performance Monitoring Unit Events," In *International Symposium on Low Power Electronics and Design*, 2005.

[3] ARM11 MPCore, http://www.arm.com/products/CPUs/ARM11MPCoreMultiprocessor.html

[4] R. Kufrin, "PerfSuite: An Accessible, Open Source Performance Analysis Environment for Linux," In *International Conference on Linux Clusters*, 2005.

[5] "Linux x86 Performance Monitoring Counters Driver," http://www.csd.uu.se/mikpe/linux/perfctr/

[6] J. Dongarra, K. London, S. Moore, P. Mucci, and D. Terpstra, "Using PAPI for hardware performance monitoring on Linux systems," *Proceedings of International Conference on Linux Clusters*, 2001.

Acknowledgement

This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MEST) (No. R0A-2007-000-20116-0). This work was supported by World Class University (WCU) program through the Korea Science and Engineering Foundation funded by the Ministry of Education, Science and Technology (R33-2008-000-10095-0). This work was also supported in part by the Brain Korea 21 Project in 2008 and Samsung Electronics Inc.. The ICT at Seoul National University provides research facilities for this study.