System Energy Consumption is a Multi-player Game

[Extended Abstract]

Mian Dong[†], Tian Lan[‡], and Lin Zhong[†]

[†]Rice University, Houston, TX [‡]George Washington University, Washington, DC
{dongmian, lzhong}@rice.edu tlan@gwu.edu

1. INTRODUCTION

The ability to account system resource usage by software is the key to the design and optimization of modern computer systems. For example, scheduling and memory management are two classic operating system (OS) functions based on the ability to account the CPU and memory usage by process. Energy has become an important system resource due to electricity and thermal concerns. This is particularly true for mobile systems that are battery-powered and require compact form factors. Knowing the energy contribution by a process, or *per-process energy accounting*, is the foundation for OS energy management and optimization [10, 8], incentive mechanisms for emerging applications in participatory sensing and cooperative communication, detecting rogue applications [7], and software optimization for energy [5].

Per-process energy accounting has been a long-standing, hard problem for multiprocessing systems where multiple processes can be active at the same time. Yet modern mobile systems are multiprocessing. Mobile System-on-Chip (SoC) provides multiple, heterogeneous processing units that can serve several software processes simultaneously within a scheduler period. The multiprocessing capabilities have been capitalized by both the mobile OSes and third-party applications developers, e.g., to enable background applications in mobile systems.

In the past decade, many researchers have attempted to solve the problem with a two-step approach, exemplified recently by [11, 7, 5, 9]. They first employ a model for system energy consumption that estimates system energy consumption from system resource usage statistics obtainable in software. By estimating the contributions by a process to the usage statistics and plugging them into the linear model, the authors infer the energy contribution by the process. *This approach is fundamentally limited because energy estimation in the first step is significantly hindered by how the estimation is distributed to processes in the second step.* That is, the model must employ linear aggregation and employ predictors that can be counted per process. As a result, existing solutions only work when there is a single running application and where system energy consumption is determined by hardware visible to software such as CPU, memory, and disk usage. Unfortunately, modern mobile systems are multiprocessing and have rich collection of I/O and heterogeneous resources, many of which contribute significantly to system energy consumption but remain invisible to the OS [6]. None of the existing solutions provide accurate per-process energy accounting when multiple processes are active. Even worse, there is no known ground truth with which one could evaluate any policy for allocating system energy consumption to running processing, e.g., policies based on per-process resource usage.

2. MULTI-PLAYER GAME

Our key insight is that per-process energy accounting can be formulated as a problem that has been extensively studied in game theory: when multiple players participate in a game and the game produces a surplus, how to divide the surplus among the players? Shapley value [1] is a well-known single value solution to this problem. For any coalition of players $S \subseteq N = 1, 2, \ldots, n$, we denote $v(S)$ as the game surplus if played by coalition S. Let $\phi_i(v)$ denote the share of surplus player i receives. Shapley value defines the *only way* to distribute the surplus among the n players that satisfies four simple axioms [1]. That is, the share received by player i, $phi_i(v)$, is

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n|S|1)!}{n!}(v(S \cup \{i\})v(S)),$$

where $|S|$ is the cardinality of S, or the number of members of S. To calculate Shapley value $\phi_i(v)$, one need not only the surplus when all n players play, $v(N)$, but also the surplus when a subset of the n players play, or $v(S)$, for all $S \subseteq N$. Shapley value has been widely applied for solving benefit and cost sharing problems in diverse fields including computer science [4, 3].

We apply multi-player game theory to per-process energy accounting by treating any time interval in which the system under question consumes energy as a game; the processes active in the interval the players; and the system energy consumption, $E(N)$, the game surplus. Naturally, Shapley value provides a powerful framework to allocate the system energy consumption to the processes. We note the

four axioms required by Shapley value are natural for treating system energy consumption as a multi-player game. (*i*) *Efficiency* requires the sum of the energy contributions by all processes be equal to the total system energy consumption. (*ii*) *Symmetry* requires that the energy contributions by two processes are identical if replacing one with the other in a game does not change the system energy consumption of every possible coalition. (*iii*) *Dummy* requires that if adding a process to every possible coalition will not change the system energy consumption, the energy contribution by this process should be zero. (*iv*) *Additivity* requires that the same energy attribution policy should work for all the time intervals of the same length.

3. CHALLENGES AND SOLUTIONS

In theory, given $E(S)$ for all $S \subseteq N$, calculating the energy contribution by a process based on Shapley value is simple. The key problem, however, is to obtain $E(S)$ for all $S \subseteq N$, which poses three system challenges.

First, there are 2^n subsets for n processes. In order to apply Shapley value, one must obtain $E(S)$, the system energy consumption, for 2^n different coalitions, i.e., S, which can be practically very difficult, if possible at all. Second, $E(S)$ depends on not only which processes are running, but also the dynamics of process execution such as the CPU time of each process and the execution order. Therefore, $E(S)$ is not a fixed number but a random variable. The variance of $E(S)$ introduces uncertainty in energy accounting by Shapley value. Finally, $E(S)$ is further affected by the hardware state in a mobile system such as CPU frequency, display brightness, and GPS on/off mode. This will introduce even more variance in $E(S)$.

We have been investigating three methods to tackle the above challenges. The first two challenges can be tackled by estimating system energy consumption, E, in situ for very short time intervals, down to the OS scheduler period, which is 10 ms in most mobile systems. Fewer processes can run simultaneously in a shorter time interval. In a time interval of 10 ms, n is not many more than the number of processing units. Moreover, in a shorter time interval, the process execution dynamics have less impact on energy consumption. By monitoring the system energy consumption in situ for an extended period of time, one can potentially acquire $E(S)$ for many different S. In [2], we report promising results from estimating system energy consumption for 10 ms time intervals in situ using power readings from the battery interface and a suite of machine learning techniques. The first challenge can be further tackled by extending the Shapley value theory to work without $E(S)$ for all $S \subseteq N$, or a partially defined multi-player game. Finally, to address the third challenge, one can incorporate hardware states as a condition into $E(S)$, or estimate system energy consumption for a coalition S given the hardware state s, $E(S|s)$. As a result, one can apply Shapley value separately to time intervals with a given hardware state.

4. REFERENCES

[1] K. Arrow and H. Kuhn. *Contributions to the Theory of Games*, volume 2. Princeton University Press, 1953.

[2] M. Dong and L. Zhong. Self-constructive high-rate system energy modeling for battery-powered mobile systems. In *Proc. ACM Int. Conf. Mobile Systems, Applications, and Services (MobiSys)*, June 2011.

[3] J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63(1), 2001.

[4] V. Misra, S. Ioannidis, A. Chaintreau, and L. Massoulié. Incentivizing peer-assisted services: a fluid Shapley value approach. In *ACM SIGMETRICS Performance Evaluation Review*, volume 38, 2010.

[5] R. Mittal, A. Kansal, and R. Chandra. Empowering developers to estimate app energy consumption. In *Proc. ACM Int. Conf. Mobile Computing and Networking (MobiCom)*, August 2012.

[6] R. Muralidhar, H. Seshadri, K. Paul, and S. Karumuri. Linux-based ultra mobile pcs. In *Proc. Linux Symposium*, 2007.

[7] A. Pathak, Y. Hu, and M. Zhang. Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. In *Proc. ACM European Conference on Computer Systems (EuroSys)*, 2012.

[8] A. Roy, S. Rumble, R. Stutsman, P. Levis, D. Mazieres, and N. Zeldovich. Energy management in mobile devices with the cinder operating system. In *Proc. ACM European Conference on Computer Systems (EuroSys)*, 2011.

[9] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha. Appscope: Application energy metering framework for android smartphone using kernel activity monitoring.

[10] H. Zeng, C. Ellis, A. Lebeck, and A. Vahdat. ECOSystem: Managing energy as a first class operating system resource. In *ACM SIGPLAN Notices*, volume 37, 2002.

[11] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. Dick, Z. Mao, and L. Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proc. IEEE/ACM/IFIP Int. Conf. Hardware/Software Codesign and System Synthesis*, 2010.