Methods for Energy Modelling of Embedded Operating System

Jari Hyvärinen
Helsinki University of Technology
jahyvari@cc.hut.fi

Abstract

Energy efficiency of mobile devices becomes more and more important with the increase in their capabilities. Since the operating system of these devices is big energy user, knowing more about it's power usage is important.

This paper compares the energy modelling methods created by Tan et al. and Li et al. I present the summaries of their methods and discuss their suitability for other operating systems than those presented in their papers.

KEYWORDS: energy modelling, operating system, embedded system

1 Introduction

With the increase in computing power, mobile devices are no longer designed to perform only one task. Mobile phones used to be plain phones, but now they have digital cameras, GPS-location services and similar features built in. This increase in number of mobile device's functions means that their embedded operating systems have replaced other software architectures in mobile devices, since embedded operating systems can handle addition of new features most easily.

While the computing power of mobile devices has been rising rapidly, the battery capacity hasn't increased as fast. This means that energy efficiency has become more and more important and a big part of the energy usage is now the embedded operating system.

Companies need to get their products to the market place fast and it is always much better to build system right first time than to try correct it later. Knowing the energy model of the operating system that will be used gives the developers valuable information. With it they can make big decisions early in software development. Thus, having a model of operating system's energy usage available during the design phase is important.

Another way of improving the energy efficiency of a device is managing the system's power consumption at runtime. Knowing the energy model of the operating system helps in finding the right balance between performance and energy usage.

This paper compares two recent energy modelling methods made by Tan et al. [12] and Li et al. [9]. Both groups use a similar approach to the problem: measure the energy usage of the operating system's basic routines with test programs and then create a model from the results. However, while Tan et al. focused on producing an energy model that would be helpful in design phase, Li et al. wanted to make an energy model that could be used to manage the system's energy usage in run-time.

In Section 2 I present the energy modelling methods created by Tan et al. and Li et al and in section 3 compare them to each other. Section 4 considers the methods suitability to be used on other operating systems. Section 5 discusses possible improvements to methods and section 6 presents the conclusions.

2 Energy Modelling Methods

2.1 Macromodel Method

2.1.1 Method

The method developed by Tan et al. [12] gives an energy model that is useful for comparing the effect of different design decisions on the energy usage of the ready product. In this method operating system is considered to be one big box. Through the box go multiple paths and each path corresponds to different work done by the operating system. Paths can be divided into two different groups.

First group of paths that are those generated by system calls from applications. This means that analysing the energy usage of these paths gives valuable insight into development of an application. The starting point for this analysis is operating system's source code. From the source code developers should pick those system calls that they believe will be heavily used or significant for deciding between different software architectures. After this, a test program should be coded for each selected path and run through a simulator. For simulator Tan et al. used both Sparcsim [6] and EMSIM [11]. Both of these simulators report the used energy for each function call. This method uses the results from simulators to calculate average energy usage for selected paths. The resulting energy model can then be used to choose between different architectural solutions or as part of operating system's total energy usage model.

The second group is paths that are generated by interruptions and other operating systems' internal workings, like context switching. Energy usage of these paths can also be measured in energy usage simulators by using suitable testing programs. When grouped together, the results from both paths form the model of the operating system's energy usage.

2.1.2 Experiments

Tan et al. tested their method by creating energy models for two different operating systems, μC/OS[8] and Linux

OS(arm-linux v2.2.2)[1]. They used Sparsim to collect the energy data required for characterization for μC/OS and EMSIM for Linux OS. After the group had completed the models, they used the same energy simulators to measure the energy usage of different test programs and compared them to predicted values.

The model for μC/OS gave values that were all within 2% of the measured values. In the Linux OS's case the results were more mixed, with most of the results staying within 5% of the measured values. However there were also cases where the difference was bigger, from 11.2% all the way to 27.5%. The only case where the error was over 17% was freeing allocated memory, which was among the smallest in actual energy usage. This level of accuracy is still usable for choosing between different software architectures.

2.1.3 Limitations

The biggest limitation of the method created by Tan et al. is that it is designed with only certain types of operating systems in mind. The operating system needs to be either monolithic or microkernel-based with synchronous message passing mechanism. Monolithic operating systems run all their services in same memory area with the main kernel thread and both Linux and μC/OS belong to this group. Microkernel-based operating systems have very minimal set of system calls to access hardware and all the services are run in user-space. L4[3] is an example of microkernel-based operating system with synchronous message passing mechanism. This means that the method doesn't work as such for other OS types.

Second limitation is that this method uses the operating system's source code in the beginning of the process. The source code is used in creating of test programs that trie to measure the operating system's interrupts' energy costs. Without knowledge of the operating system's internal mechanics measuring the energy usage of these interrupts is harder.

2.2 Method for Run-Time Model

2.2.1 Method

Li et al. present another method for estimating energy usage in their paper[9]. Their goal was to create a model that is usable at run-time for power management. The method uses system power simulator called SoftWatt [7] to measure the system's energy usage and the model is built based on the results.

This method doesn't separate the operating system's interrupts and system calls into different group, instead they are all handled in the same way as one group and called operating system's routines. The SoftWatt is used to measure the energy usage of each routine. Initial testing showed that simply calculating the average cost of operating system's basic routines wasn't accurate enough. However, measuring each routine's energy usage separately gave good results.

One important finding of their research was that even the energy usage of one single routine could change dramatically. Upon closer examination they found that the power usage of the routines correlated with the number of instructions the processor completed every cycle while processing the routines. So, if two routines are handled simultaneously, it will consume more energy than completing them one after another. Since the correlation of instructions per cycle and energy usage is linear, including this correlation number in energy model means that the model can give results that are more accurate without noticeable increase in the model's complexity.

The energy model that this method gives doesn't require much memory, only 2 numbers are required for each system call and interrupt: base energy usage and number for correlation. Since the energy usage is in linear correlation with the instructions completed per cycle, it means that there isn't big overhead in calculating the estimated power usage from the base energy usage and the number describing how routine's power usage changes with increased processor load. These two facts make this energy model well suited for run-time managing of devices power consumption.

2.2.2 Experiments

Li et al. tested their method on a simulated version of SGI IRIX 5.3[2]. They used operating system simulator SimOS[10] to run simulated operating system and SoftWatt to measure the energy usage. This made possible for them to measure the energy usage of each routine of the operating system.

For the testing Li et al. used two different sets of test programs. The first had programs that were used for collecting the data of energy usage. This data was then used to make the energy model. The second group of programs was used to test how well the numbers given by the energy model would match the measured values. The results showed that the estimates of single routine's power consumption were all within 6% from measured value.

2.2.3 Limitations

Since Li et al. use the same basic method as Tan et al., all the same limitations in the operating systems apply to their method. However, since this method requires more detailed information from the simulator, finding suitable simulator will be harder than with the method of Tan et al.

3 Comparison

3.1 General Comparison

The basic idea of both modelling methods is the same: use a simulator to measure energy usage of operating system's basic routines. Despite this the different goals of the groups resulted in slightly different methods for energy modelling.

Tan et al. concentrated on giving designers new tools for comparing different architecture solutions and operating systems. Because of this there isn't need for their method to analyse whole operating system. Instead, it concentrates on modelling only those of the operating system's routines that are either heavily used or that can affect the architectural design of the application. This means that the number of things

that need to be measured in the energy usage simulator is much smaller than in the method proposed by Li et al.

Li et al. wanted to create method that creates an energy model for run-time power management. Because they included the number of instructions the processor completes per cycle, their model gives accurate results even when the processor's load varies. Since the model is meant for power management, it measures energy usage of every routine of operating system. This means that the time needed to run the simulation is longer than in method created by Tan et al. On the other hand, the resulting energy model can also be used for the same purposes that energy model of Tan et al. is used for.

The symmetry of these two methods works also in the other direction. When using the method created by Tan et al., if you measure the power usage of every system call and interrupt of the operating system, you get an energy model quite similar to that created by the method of Li et al. The only difference would be, that this energy model wouldn't include any information on how the energy consumption of operating system's routines change in relation to the processor's load(instructions per cycle). This means that the energy model could be used for run-time management of energy usage, but the result wouldn't be as good as those you would get by using the method of Li et al, since it wouldn't account for the changes in the energy usage caused by the processor's varying load.

3.2 Methodology

Since the methods created by Tan et al. and Li et al. are so similar, it is possible to present generalised methodology for both of them. The steps of the methodology are:

1. Analyse
2. Select test programs
3. Measure energy usage
4. Model-fitting
5. Verify results

In the first step the operating system is analysed. The purpose of this analyse is to identify those system calls and interrupts that will be inspected in later steps. Those who only want to use the resulting energy model for architectural considerations can use the method of Tan et al. and inspect only few routines, but if the energy model will be used for run-time power management, then every system call and interrupt should be included.

The selecting of test programs is heavily influenced by capabilities of the testing system. Li et al. had access to a system that was capable of measuring the energy usage of individual system calls and interruptions. Because of this they could use general testing programs and still get accurate numbers. The simulators used by Tan et al. couldn't separate the energy usage of different system calls and interrupts from each other, so the group made their own testing programs that only used specific system calls. This way they could separate the energy consumption of different interrupts from the actual energy usage of each system call. After the test programs have been selected, measuring the energy usage of system calls and interrupts is straightforward.

In the fourth step the measured energy usages are fitted into macromodel templates. Macromodel templates are equations in the form:

$$E = c_1 R_1 + c_2 R_2 + ... + c_n R_n$$

The E is energy usage of the system call or interrupt, R_js are the parameters and c_js are the variables that need to be determined. The number of the variables is determined by the inspected properties of the system call. If the system call is tested with with just one input,then there is only one variable. Testing with varying lengths of input or with different processor loads would raise the number to two, and testing with both different inputs and processor loads would raise it to three.

Last step is verifying that the energy model corresponds to actual energy usage. This is done by using the energy model to predict the energy usage of program and then comparing the prediction with measured values from a test program.

4 Other Operating Systems

In this section I present some popular embedded operating systems consider the presented methods suitability for analysing them.

4.1 Symbian OS

Symbian OS[4] is currently the most used operating system in smartphones. It is designed from the start to be used in mobile devices and minimizing the use of resources has been an important design goal. This combined with the fact that phone manufacturers are very supportive for developers makes Symbian popular choice for developers. This means that an energy model for Symbian would be useful for many people.

Nokia has provided a program for monitoring energy usage of applications on phones based on s60. This means that at least on that platform there is way to measure the energy usage of Symbian. The accuracy of the monitoring program isn't too good, but could be enough if the test programs are made to run for longer time periods. However, differentiating operating system's interrupts from the tested system calls won't be easy at this resolution.

Symbian has a hybrid kernel. From microkernel side it uses the idea of keeping the kernel itself small and running the operating system's services in user mode. The exception to this is that device drivers are included inside the kernel. However, because the kernel uses asynchronous message passing, the proposed methods don't work as they are.

4.2 Windows CE

Windows CE[5] is version of Microsoft's Windows that is designed to be used in embedded systems. For this reason it is designed to work on limited system resources.

Microsoft has made available the kernels source code to make it easier to develop applications for it. This makes Windows CE more suited for energy modelling than other versions of Windows.

Even though Windows CE's kernel is also hybrid between microkernel and monolithic kernel like other versions of Windows, it isn't just simplified version of regular Windows versions. Instead Windows CE is more modular so that manufacturers can use only the needed parts of it in their devices. From the energy modelling perspective the bad side of this is that the energy model of the operating system isn't as useful since the operating system isn't same for everyone.

All in all Windows CE looks like it would be good target for energy modelling because of the availability of source code and the kernel architecture. The problem is in measuring the power usage of the system calls and interrupts. Since Windows CE supports so many different hardware platforms, finding simulator for measuring the energy usage on specific hardware is hard.

5 Challenges & Improvements

5.1 Challenges

One big open question is how well does the method of Li et al. work with different operating systems or hardware? Does the instructions per cycle affect the energy usage in just as straight forward way with different hardware and operating system? The experiments done by Tan et al. showed that although one operating system and hardware pair gave really good results, the other was significantly different. Clearly there is need for some testing to make sure that the principle is widely usable.

Both methods also share the same problem: they need a simulator that can measure the energy usage when test programs are run in the operating system. Simulators are usually designed to simulate only a couple of different operating systems on one hardware configuration. Because of this, it is difficult to compare two different operating systems on same hardware, unless one simulator supports both of them. Of course, if the intention is just trying to make design decision on one particular operating system(intended use of method from Tan et al.) or trying to manage the operating system's energy usage run-time(intended use of method from Li et al.), you don't need to compare different operating systems. Still, considering that there are tens of embedded operating systems in use, lack of simulators does limit the methods' usefulness.

5.2 Improvements

The energy modelling methods presented by Tan et al. and Li et al. don't leave much room for improvement. The simulator side of methods is where the problems are. Because both methods need to know the energy usage of the operating system's routines, a general level simulator doesn't suffice. Since the amounts of energy used per one use of a routine are measured in less than microjoules [12], measuring them without a simulator doesn't seem practical.

The other direction where improvements can be found is trying to make the methods work with Operating systems with other kernel architectures than monolithic and microkernels with synchronous message passing mechanism. Microkernels with asynchronous message passing mechanism seem to next logical step, since microkernels with synchronous message passing mechanism already work. The basic idea of measuring the energy cost of operating system's routines would still be the same. Only getting the correct value would be harder if the simulator can't measure operating system's interrupts' energy usage, since with asynchronous message passing triggering the interrupts at predictable time is harder.

6 Conclusions

The importance of energy usage is growing because computation power of modern mobile devices, along with their power consumption, increases faster than battery capacity. The operating system is a big part of total energy use of device, therefore analysing the characteristics of it can guide the development of new devices.

Both Tan et al. and Li et al. have developed methods that create an energy model for an operating system and even though their basic idea is same, intended use of the energy models resulted in different methods. This paper presented these two methods and compared their similarities and differences. Tan et al. created a method that needs less simulation in energy usage simulator and gives numbers only for those parts of operating system that user is interested in. By contrast, Li et al. created method that requires more simulation time but that results in model which is usable for run-time power management.

Since the methods are quite similar to each other, they also share same weaknesses. Both require simulator that can measure the energy usage of the operating system's routines. This can be problem because there are many embedded operating systems in use today and not all of them have simulator for energy usage. Both methods are also limited in the types of operating systems they can analyse.

Symbian and Windows CE are both popular embedded operating systems and I considered the suitability of the energy modelling methods on them. Common problem for them is the lack of suitable energy simulators. Also, since Symbian's kernel is microkernel with asynchronous message passing, creating it's energy model is harder.

References

[1] Arm linux. URL: `http://www.arm.linux.org.uk/`.

[2] Irix. Silicon Graphics, Inc. URL: `http://www.sgi.com/products/software/irix/`.

[3] L4. URL: `http://os.inf.tu-dresden.de/L4/`.

[4] Symbian os. Symbian Ltd. URL: `http://www.symbian.com/`,.

[5] Windows ce. Microsoft Corporation. URL: `http://msdn.microsoft.com/embedded/windowsce/default.aspx`,.

[6] R. P. Dick, G. Lakshminarayana, A. Raghunathan, and N. K. Jha. Power analysis of embedded operating systems. In *DAC '00: Proceedings of the 37th conference on Design automation*, pages 312–315, New York, NY, USA, 2000. ACM.

[7] S. Gurumurthi, A. Sivasubramaniam, M. J. Irwin, N. Vijaykrishnan, M. Kandemir, T. Li, and L. K. John. Using complete machine simulation for software power estimation: The softwatt approach. In *HPCA '02: Proceedings of the 8th International Symposium on High-Performance Computer Architecture*, page 141, Washington, DC, USA, 2002. IEEE Computer Society.

[8] J. J. Labrosse. *Microc/OS-II*. R & D Books, 1998.

[9] T. Li and L. K. John. Run-time modeling and estimation of operating system power consumption. *SIGMETRICS Perform. Eval. Rev.*, 31(1):160–171, 2003.

[10] M. Rosenblum, S. Herrod, E. Witchel, and A. Gupta. Complete computer system simulation: the simos approach. *Parallel & Distributed Technology: Systems & Applications, IEEE [see also IEEE Concurrency]*, 3(4):34–43, Winter 1995.

[11] T. Tan, A. Raghunathan, and N. Jha. A simulation framework for energy-consumption analysis of os-driven embedded applications. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 22(9):1284–1294, Sept. 2003.

[12] T. K. Tan, A. Raghunathan, and N. K. Jha. Energy macromodeling of embedded operating systems. *Trans. on Embedded Computing Sys.*, 4(1):231–254, 2005.