Project Proposal for CS736: Advanced Operating Systems
E-MOS: Efficient Energy Management Policies in Operating Systems

Vinay Gangadhar
vinay@cs.wisc.edu

Clint Lestourgeon
clint@cs.wisc.edu

Jay Yang
jkyang2@wisc.edu

Varun Dattatreya
channagirida@wisc.edu

1. Problem

Current Linux Operating Systems' power governors do not provide a fine-grained control and management over the energy utilized by the applications running on a computing system. Linux power governors can either be set to statically scale the CPU frequency or they provide limited user capability to define the energy requirements of an application. These power governors operate on a generic energy management policy, which is based on the assumption that reducing frequency of CPU (using DVFS) or putting the system to low-power state would reduce the system energy always, without having necessary information about the application. In this project, we plan to enhance the Linux OS to have better energy management policies/mechanisms by first analyzing the resource consumption (CPU, Memory, I/O) of the applications and then using this application information to make better energy decisions in the kernel. Our E-MOS aims to provide improved energy efficiency to applications by having a fine-grained control over the energy (battery) resource.

2. Motivation

Modern multi-core computing systems (Laptops and hand-held devices) have major challenge of limiting the energy consumed by the system resources (CPU, Memory, Network) utilized when running applications. Most of the work on energy conservation in operating systems mainly rely on techniques which dynamically characterize power consumption and Quality of Service (QOS) of applications and provide a policy for users by using DVFS (Dynamic voltage frequency scaling) [3] or lower-CPU power states [5, 7].

Linux OS provide ACPI [1, 2] interfaces to extract information about batteries and other resources, but the power governors don't utilize this information efficiently to make proper energy management decisions. Current policies generally rely on user-specified static parameters to run at a specific lower CPU frequency or lower system power state without considering the applications' requirements. Many multi-core processors try to improve the applications execution time by running at higher frequencies and thus consume more power (May consume less energy, if the execution time is small). Some applications have more memory accesses, in which case running at higher frequencies always is not beneficial. Many applications are long running jobs which consume lot of energy if run at higher frequency. All these above scenarios suggest that there is 'NO' one-size fit all solution for better system energy efficiency. Thus, Linux power governors need more application information, to determine whether to boost the performance or run at lower-power state by continuously monitoring the available battery resource. Our project aims to solve this problem of energy management decision by deriving some of the principles from Liang et. al [4] implemented for Android systems.

3. Project phases

The project will be distributed into multiple phases, but some phases are orthogonal so that it can be implemented in parallel among the project members.

- **Phase 0:** Picking an application suite representative of typical Linux system (Single/Multi-threaded, I/O, Memory bound).
- **Phase 1:** Profiling the applications and categorizing them into different buckets – CPU intensive, Memory intensive, I/O intensive. Determine the energy needs for each bucket.
- **Phase 2:** Implement framework to collect dynamic information about battery resource, CPU utilization, memory frequency (ACPI and Smart battery interfaces are good place to start). This has to be fed to power governors at specific time intervals to make certain decisions to actually reduce the CPU frequency or go into a lower system power mode. A decision table data structure to be formulated here based on application requirement and actual energy resource available.
- **Phase 2:** Implementing a Linux power governor to support policies/mechanisms for the energy decisions to be taken based on information obtained in Phase 1 and Phase 2. User-space power governors could be a good place to start.
- **Phase 4:** Analyze and compare/reason-out the effects of the energy policies implemented in E-MOS with standard Linux platform.

4. Methodology and Evaluation

This also lists the resources we will be using for our project.

For our study, we will be modifying a stable version of Linux kernel with ACPI and Smart battery interface support. PinTool [6] will be used for profiling applications and categorize them into different buckets. To get initial estimates of power/energy consumption of applications, we plan to use two methods – Our first method uses the ACPI interface to measure the runtime energy (battery) consumption at specific time intervals (epochs) based on the average execution time of all the applications and second method is to use a simulator integrated with energy measurement tool (Just to see whether our measurements co-relate). We plan to develop a energy decision table, with category of application on one dimension and concerned energy rule/target on the other. This decision table will be used to develop policies in Linux power governor. Evaluation will be done by determining the performance and energy for all the applications with E-MOS implemented. Performance counters (RAPL tool) and again standard ACPI interface for energy measurements will be used. Finally, we will compare it with runs on standard Linux platform.

References

[1] [Online]. Available: https://www.kernel.org/doc/Documentation/power/apm-acpi.txt

[2] [Online]. Available: https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

[3] R. Z. Ayoub, U. Ogras, E. Gorbatov, Y. Jin, T. Kam, P. Diefenbaugh, and T. Rosing, "Os-level power minimization under tight performance constraints in general purpose systems," in *Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design*. IEEE Press, 2011, pp. 321–326.

[4] Y. Liang, P. Lai, and C. Chiou, "An energy conservation dvfs algorithm for the android operating system," *Journal of Convergence*, vol. 1, no. 1, 2010.

[5] Y. Liu, S. C. Draper, and N. S. Kim, "Sleepscale: runtime joint speed scaling and sleep states management for power efficient data centers," in *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*. IEEE, 2014, pp. 313–324.

[6] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: building customized program analysis tools with dynamic instrumentation," in *Acm Sigplan Notices*, vol. 40, no. 6. ACM, 2005, pp. 190–200.

[7] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat, "Ecosystem: Managing energy as a first class operating system resource," *ACM SIGPLAN Notices*, vol. 37, no. 10, pp. 123–132, 2002.