OS-Directed Power Management for Mobile Electronic Systems

Qinru Qiu, Qing Wu and Massoud Pedram
Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089
{qinru, qwu, pedram}@usc.edu

Abstract

We present methods for operating system directed dynamic power management. We model a power-managed system using a continuous-time Markov decision process and solve for the optimal power management policy using a mathematical programming technique. Next we extend the model by using the controllable Generalized Stochastic Petri Nets with cost to handle systems with complex behavioral characteristics such as concurrency, synchronization, mutual exclusion and conflict. Experimental results demonstrate the effectiveness of the proposed modeling framework and solution techniques.

Introduction

Military systems have to be low power yet high performance. This is because they are battery-operated portable (wearable) systems that require significant computation and communication capabilities. Sophisticated dynamic power management policies, power-aware operating systems and C compilers, and hardware/software co-synthesis tools are required to satisfy these often conflicting design requirements. We propose to investigate a number of problems at the system, software, operating system, and behavioral levels, which are critical to effective power-performance tradeoff and optimization in complex embedded systems.

Dynamic power management (DPM) [2]– which refers to selective shut-off or slow-down of system components that are idle or underutilized – has proven to be a particularly effective technique for power saving. The goal of a dynamic power management policy is to reduce the power consumption of an electronic system by putting system components into different states, each representing certain performance and power consumption level. The policy determines the type and timing of these transitions based on the system history, workload and performance constraints.

We present a simple example to illustrate the basic idea behind dynamic power management and the resulting trade-off space. Consider an I/O device (service provider) that is controlled by the OS-directed power management software and that can be put into one of two modes: "ON" and "OFF". The device consumes 10 Watts of power when it is in the "ON" state and 0 Watt when it is "OFF". It takes 2 seconds and 40 Joules of energy to turn this device from "OFF" to "ON" whereas it takes 1 second and 10 Joules of energy to turn it from "ON" to "OFF". The service request pattern in a time period of 25 seconds is shown in Figure 1(a).

Without a power management policy, this device is always "ON" as depicted in Figure 1(b). The shaded areas in the figure identify the time periods when the device is processing the requests. Under a "greedy power management policy", the OS turns on the device whenever there is a service request for it and turns off the device whenever it has nothing to do. The working pattern of the device under this policy is shown in Figure 1(c). If, however, the average delay for processing a request can be increased, then the power management policy shown in Figure 1(d) will result in minimum power dissipation (total energy in 25 seconds) for the device under the specified delay constraint. The comparison between the three policies in terms of total energy consumption and average delay is presented in Table 1.

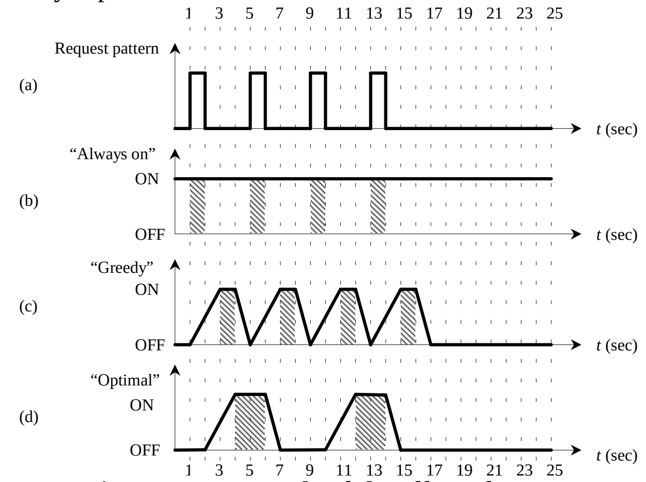


Figure 1 An example of the effect of power management.

Table 1 Comparison of total energy consumption and average delay.

	Total energy consumption in 25 sec	Average latency per request
"Always on" policy	250 J	1 sec
"Greedy" policy	240 J	3 sec
"Optimal" policy	140 J	2.5 sec

This simple example illustrates the fact that there is a fundamental tradeoff between power dissipation and latency per request (while maintaining the same throughput) and that a "good" power management policy is one that can exploit this trade-off dynamically.

DPM Background

Portable electronic devices tend to be much more complex than a single VLSI chip; they contain many components, ranging from digital and analog to electro-mechanical and opto-electronics. Digital components often consume 20-50% of the total power. Hence, reducing power consumption in the non-digital electronic parts of the system is as important. System designers have started to respond to the requirement of power-constrained system designs by a combination of technological advances and architectural improvements.

The problem of finding a power management scheme (or policy) that minimizes the system power dissipation under performance constraints is of great interest to system designers. Aside from finding an effective policy for a given electronic system, the design and implementation of such a policy in the context of the system itself is a complicated and error-prone process. Realizing this difficulty, a number of standardization efforts have started, chief among them is the Advanced Configuration and Power Interface (ACPI) [1].

Under OS-directed power management (OSPM), the operating system directs all system and device power state transitions. Employing user preferences and knowledge of how devices are being used by applications, the OS puts devices in and out of low-power states. Devices that are not being used can be turned off. Similarly, the OS uses information from applications and user settings to put the system as a whole into a low-power state. The OS uses ACPI to control power state transitions in hardware. The functional areas covered by the ACPI specification are:

1. System power management – ACPI defines mechanisms for putting the computer as a whole in and out of system sleeping states. It also provides a general mechanism for any device to wake the computer up.
2. Device power management – ACPI tables describe motherboard devices, their power states, the power planes that the devices are connected to, and controls for putting devices into different power states. This enables the OS to put devices into low-power states based on application usage.
3. Processor power management – While the OS is idle but not sleeping, it will use commands described by ACPI to put processors in low-power states.

ACPI does not, however, specify the power management policy. It is the objective of the proposed research to provide a framework and supporting tools for constructing optimal power management policies based on modeling the power-managed system as a continuous-time Markov decision process. The OS must be modified to provide the stochastic parameters of this stochastic model by profiling the activities of the system components when an application program is run on the system.

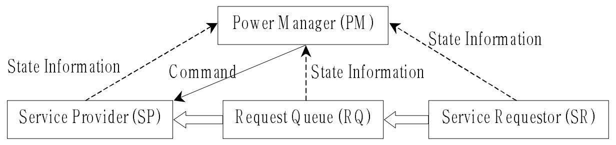


Figure 2 An abstract model of a power-managed system.

The choice of the policy that minimizes power dissipation under a performance constraint (or maximizes performance under a power constraint) is a new kind of constrained optimization problem that is of great relevance for low-power electronic systems. This problem is often referred to as the *policy optimization* (PO) problem. A number of heuristic policies have been proposed and adopted in some practical systems [3][4]. In [5], Benini et al. proposed a stochastic model for a rigorous mathematical formulation of the problem and presented a procedure for its exact solution. The solution is computed in polynomial time by solving a linear optimization problem. Their approach is based on a stochastic model of power-managed devices and workloads and leverages well-known stochastic optimization techniques based on the theory of *discrete-time Markov decision chains*. In the model of [5], time is divided into small intervals of length L. It is assumed that the system can only change its state at the beginning of a time interval. During interval (jL, $(j+1)L$), the transition probability of the system depends only on the state of the system at time jL (hence, the Markovian property) and the command issued by the power manager. The system model consists of four components: a *power manager* (PM), a *service provider* (SP), a *service requestor* (SR) and a *service request queue* (SQ). (See Figure 2.) The objective function is the expected performance (which is related to the expected waiting time and the number of jobs in the queue) and the constraint is the maximum expected power consumption (which is related to the power cost of staying in some server state and the energy consumption required for the transfer from one server state to the next).

Modeling Framework

In our stochastic model [6][7], instead of dividing the time into discrete intervals, we treat time as a continuous variable.

In the discrete-time model, the stochastic character of the system is represented by the probability that a certain event will occur in a given time interval, e.g., the probability that a service request comes during some interval, the probability that the server finishes serving a given task, and so on. In the continuous-time model, the stochastic character of the system is represented by 1) the probability distribution function for the time that the system requires to perform a certain task such as the time it needs to provide service for a given request, the time it needs to change a server state, *etc.*, and 2) the probability distribution function for the inter-arrival time of the task requests. We have shown that a continuous-time system model is more efficient for doing transient analysis of the system and is more accurate compared to the discrete-time model. Furthermore, in the continuous-time model, the behavior of the system does not need to be synchronized to each time interval; therefore, the resulting power management can be asynchronous (or event-driven). This means that the power manager issues a command only when the state of the system changes. Consequently, in a system with a low incoming request rate, an asynchronous power manager will consume much less power than a synchronous power manager.

Further more, our model explicitly distinguishes the busy state and the idle state of the SP so that the system characterization becomes accurate. The model considers the correlation between the state transition of the SQ and that of the SP. The service queue model consists of a normal queue and a high priority queue. This is important since some service requests are "urgent" and need immediate response from the server.

Further more, for example, if we want to model a power-managed system as shown in Figure 3. The example system is from a typical multi-server (distributed computing) system. Note that we are only interested in the system behavior that is related to the power management. The system contains two SPs and their own Local SQs (LSQ). There is a SR that independently generates the tasks (requests) that need to be serviced. The Request Dispatcher (RD) makes decisions about which SP should service which request. We assume that different SPs may have different power/performance parameters. In real applications, the RD and LSQs can be part of the operating system, while SPs can be multiple devices or sub-systems in our target system or the number of networked computers of a distributed computing system.

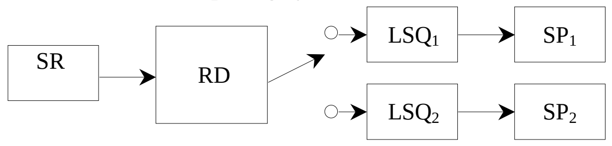


Figure 3 A multi-server/distributed-computing system.

The complexity of the modeling problem for the above system is high not only because of the increased number of components, but also because of the complex system behaviors that are present. As examples of these complex behaviors, we need to consider the synchronization of LSQs and SPs, the synchronization of the SR and LSQs, the concurrent working behavior of SPs, the dispatch behavior of the RD, and so on. In this situation when complex system behaviors is a major part of the system model, the modeling techniques mentioned earlier [5][6][7] become powerless because they only offer stochastic models for individual components and dictate that system behaviors be captured manually. Obviously, we need new dynamic power management modeling techniques for large systems with complex behaviors.

We propose a methodology [11] based on **Generalized Stochastic Petri Nets** (GSPN) to model systems with complex behaviors and calculate the optimal power management policy under given performance constraints. By using GSPN, we can easily and accurately model the system behaviors as well as model the individual components.

Industrial systems consist of multiple requesters and multiple servers; they may use a centralized or a distributed power management scheme; System components may interact in complicated manner; the requests may have different service priorities and/or be non-stationary; and so on. These features increase the complexity of the PO problem. To manage this complexity, we develop a hierarchical power management scheme where we merge states with similar characteristics into “macro-states” and then find an optimal power management policy for the coarse-grain stochastic model of the power-managed system. We also study techniques based on dropping redundant states or lumping similar states to achieve efficiency when dealing with systems with a very large number of states. In addition, we allow multiple requesters with different service request priorities and non-stationary service requests. Finally, constrained by the coarse-grain power management policy, we can search for a refined policy for the states inside each macro-state.

Solution Technique

We developed four formal optimization techniques to find the optimal DPM policy based on the above mentioned system model, which consumes minimum power while satisfies the given delay constraint.

The most widely used constraint optimization technique is linear programming based approach. In our work, we formulate the policy optimization problem as a linear programming problem. The resulting policy consumes minimum power while exactly satisfies the given performance constraint. However, the output policies are randomized policy. It means that each time, the action is taken with certain probability. Such a policy may has some difficulty in real implementation if the power manager is implemented using hardware.

On the other hand, a deterministic policy takes a deterministic policy at each time. To calculate the deterministic policy, we use a non-linear programming based approach. In addition to the constraints in the linear program formulation, the requirement of deterministic policy is formulated as a non-linear constraint.

A branch bound based algorithm is also proposed to solve the deterministic policy. In this algorithm, we search for the optimal deterministic decision using a decision tree. Associated with each leaf of the decision tree, there is a deterministic policy; associate with each branch of this tree, there is a partial decision problem and a predictor.

The predictor gives the lower bound of the power consumption of the policies in this branch. If this power consumption is higher than the best solution obtained, the branch is pruned; otherwise, we search for the optimal policy in this branch.

The policy iteration based approach is a heuristic approach, which finds optimal deterministic policy that satisfied certain conditions. The kernel of this approach is a conventional policy optimization algorithm, which finds the unconstraint optimal deterministic policy. We modified the algorithm so that it can be applied in our constraint optimization problem.

Experimental results

Experiments have been designed to examine the performance of our system model and optimization method. We present the comparison of our DPM policy and the heuristic policies including greedy policy, time-out policy and show that our policy consumes less power than the heuristic policies.

The system model used in this part includes:

A SP model that has three different power modes: active, waiting and sleeping.

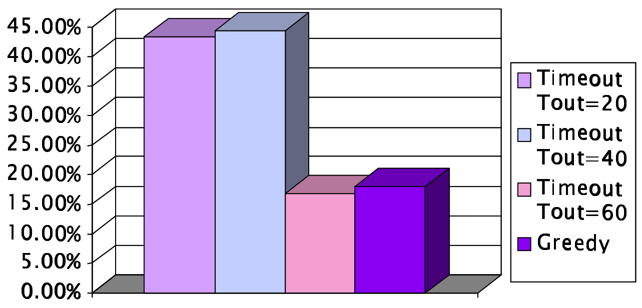
A SR model based on real applications.

A SQ model based on real implementation in the OS.

We compare the new DPM policy with two heuristic power management methods:

1. Greedy policy: turn on the SP whenever a request comes and turn off the SP whenever the SP is idle and there is no request in the queue.

2. Timeout policy: turn on the SP whenever a request comes and turn off the SP whenever the SP has been idle for tout and there is no request in the queue.



Power Improvement of Our approach vs. Heuristics

Figure 4 Experimental results.

We simulated the heuristic policies to get the average delay of the low priority and high priority requests. We then use these delay values as the delay constraint and searched for a randomized DPM policy using linear programming. Finally, We simulated the DPM policies and compare the power and delay value with that of heuristic policies. From Figure 4 we can see that the DPM policy saves more than 15% power than the heuristic policy while having the same performance level.

Conclusions

We presented methods for operating system directed dynamic power management. We model a power-managed system using a continuous-time Markov decision process and solve for the optimal power management policy using a mathematical programming technique. We also extended the model by using the controllable Generalized Stochastic Petri Nets with cost to handle systems with complex behavioral characteristics such as many realistic mobile systems. Experimental results demonstrated the effectiveness of the proposed modeling framework and solution techniques.

Reference

[1] Intel, Microsoft and Toshiba, "Advanced Configuration and Power Interface specification", URL: http://www.intel.com/ial/powermgm/specs.html, 1996

[2] L. Benini and G. De Micheli, Dynamic Power Management: Design Techniques and CAD Tools, Kluwer Academic Publishers, 1997.

[3] M. Srivastava, A. Chandrakasan. R. Brodersen, "Predictive system shutdown and other architectural techniques for energy efficient programmable computation," IEEE Transactions on VLSI Systems, Vol. 4, No. 1 (1996), pages 42-55.

[4] C.-H. Hwang and A. Wu, "A Predictive System Shutdown Method for Energy Saving of Event-Driven Computation," Proc. of the Intl. Conference on Computer Aided Design, pages 28-32, November 1997.

[5] G. A. Paleologo, L. Benini, et.al, "Policy Optimization for Dynamic Power Management", Proceedings of Design Automation Conference, pp.182-187, Jun. 1998.

[6] Q. Qiu, M. Pedram, "Dynamic Power Management Based on Continuous-Time Markov Decision Processes", Proceedings of the Design Automation Conference, pp. 555-561, Jun. 1999.

[7] Q. Qiu, Q. Wu, M. Pedram, "Stochastic Modeling of a Power-Managed System: Construction and Optimization", Proceedings of the International Symposium on Low Power Electronics and Design, 1999.

[8] L. Benini, A. Bogliolo, S. Cavallucci, B. Ricco, "Monitoring System Activity For OS-Directed Dynamic Power Management", Proceedings of International Symposium of Low Power Electronics and Design Conference, pp. 185-190, Aug. 1998.

[9] E. Chung, L. Benini and G. De Micheli, "Dynamic Power Management for Non-Stationary Service Requests", Proceedings of DATE, pp. 77-81, 1999.

[10] L. Benini, R. Hodgson, P. Siegel, "System-level Estimation And Optimization", Proceedings of International Symposium of Low Power Electronics and Design Conference, pp. 173-178, Aug. 1998.

[11] Q. Qiu, Q. Wu, M. Pedram, "Dynamic Power Management of Complex Systems Using Generalized Stochastic Petri Nets", to appear, *Proceedings of the Design Automation Conference*, Jun. 2000