Algorithms for Energy Saving*

Susanne Albers**

Department of Computer Science, University of Freiburg
Georges Koehler Allee 79, 79110 Freiburg, Germany
salbers@informatik.uni-freiburg.de

Abstract. Energy has become a scarce and expensive resource. There is a growing awareness in society that energy saving is a critical issue. This paper surveys algorithmic solutions to reduce energy consumption in computing environments. We focus on the system and device level. More specifically, we study power-down mechanisms as well as dynamic speed scaling techniques in modern microprocessors.

Keywords: Dynamic speed scaling, power-down mechanisms, scheduling, competitive analysis, probabilistic analysis, approximation algorithms.

1 Introduction

With increasing CPU clock speeds and higher levels of integration in processors, memories and controllers, power consumption has become a major concern in computer system design over the past years. Power dissipation is critical in battery operated mobile computing devices that have proliferated in recent years. In these devices, obviously, the amount of available energy is severely limited. Moreover, power consumption is a major concern in desktop computers and servers. Electricity costs impose a substantial strain on the budget of data and computing centers, where servers and, in particular, CPUs account for 50–60% of the energy consumption. In fact, Google engineers, maintaining thousands of servers, recently warned that if power consumption continues to grow, power costs can easily overtake hardware costs by a large margin [11]. In addition to cost, energy dissipation causes thermal problems. Most of the consumed energy is converted into heat, resulting in wear and reduced reliability of hardware components.

For these reasons, there has recently been considerable research interest in the design and analysis of *energy-efficient algorithms* that reduce the energy consumption while minimizing compromise to service. This survey focuses on energy saving mechanisms on the system and device level. In this context, there are basically two techniques to save energy.

---

* An extended and modified version of this survey, aiming at a different audience, will appear in the *Communications of the ACM*.

** Work supported by a Gottfried Wilhelm Leibniz Award of the German Research Foundation.

S. Albers, H. Alt, and S. Näher (Eds.): Festschrift Mehlhorn, LNCS 5760, pp. 173–186, 2009.

(1) *Power-down mechanisms*: When a system is idle, it can be transitioned into low-power standby or sleep states. This technique is well-known and widely used to save energy. One has to find out when to shut down a system, taking into account that a transition back to the active mode requires extra energy.
(2) *Speed scaling*: Microprocessors currently sold by chip makers such as AMD and Intel are able to operate at variable speed. The higher the speed, the higher the power consumption is. The goal is to save energy by utilizing the full speed/frequency spectrum of a processor and applying low speeds whenever possible.

The power management problems described above are *online problems* in that a system is usually not aware of future events. A power-down mechanism, during an idle period, usually has no information when the period ends. Is it worthwhile to move to a lower-power state and benefit from the reduced energy consumption, given that the system must finally be powered up again at a cost to the active mode? A speed scaling algorithm typically does not know future jobs. Should lower speed levels be used at the expense of delaying the service of tasks that may arrive in the near future?

Despite the handicap of not knowing the future, an online strategy should achieve a provably good performance. Here we resort to *competitive analysis* [29], where an online algorithm ALG is compared to an optimal offline algorithm OPT that knows the entire future and can compute an optimal solution. Online algorithm ALG is called *c-competitive* if, for every input, the total energy consumption of ALG is at most c times that of OPT.

In this survey we first present the most important results known for power-down mechanisms. Then we address dynamic speed scaling algorithms.

2 Power-Down Mechanisms

Power-down mechanisms are a common technique to save energy. We encounter them on an every day basis. The display of our desktop turns off after some period of inactivity. Our laptop transitions to a standby or hibernate mode if it has been idle for a while. In these settings, there usually exist idleness thresholds that specify the length of time after which a system is powered down. From an algorithmic point of view, we would like to design strategies that determine such thresholds and perform well relative to the optimum.

Formally, we are given a device that always resides in one of several states. In addition to the active state, there can be several standby and sleep modes. These states have individual power consumption rates. The energy incurred in transitioning the system from a higher-power to a lower-power state is usually negligible. However, a power-up operation consumes a significant amount of energy. Over time the device experiences an alternating sequence of active and idle periods. During active periods, the system must reside in the active mode to perform the required tasks. During idle periods, the system may be moved to lower-power states. An algorithm has to decide when to perform the transitions

and to which states to move. The goal is to minimize the total energy consumption. As the energy consumption during the active periods is fixed, assuming that prescribed tasks have to be performed, we concentrate on energy minimization in the idle intervals. In fact, we focus on any idle period and optimize the energy consumption in any such time window.

In the following we will first study systems that consist of two states only. Then we will address systems with multiple states. We stress that we consider the minimization of energy. We ignore the delay that arises when a system is transitioned from a lower-power to a higher-power state.

2.1 Systems with Two States

Consider a two-state system that may reside in an active state or in a sleep state. We assume without loss of generality that the power consumption rate in the active state is 1, i.e. the system consumes one energy unit per time unit. The power consumption rate in the sleep mode is 0. The results we present in the following generalize to arbitrary consumption rates. Suppose that β, $\beta > 0$, energy units are required to transition the system from the sleep state to the active state. The energy of transitioning from the active to the sleep state is assumed to be 0. If this is not the case, we can simply fold the corresponding energy into the cost of β incurred in the next power-up operation. The system experiences an idle period whose length T is initially unknown.

We first observe that an optimal offline algorithm OPT, knowing T in advance, is simple to formulate. If the value of T, counted in time units, is smaller than the value of β, OPT remains in the active state throughout the idle period. If T is at least β, OPT transitions to the sleep state right at the beginning of the idle period and powers up to the active state at the end of the period.

The following deterministic online algorithm mimics the behavior of OPT.

Algorithm ALG-D: In an idle period, remain in the active state first. After β time units, if the period has not ended yet, transition to the sleep state.

Theorem 1. *ALG-D is 2-competitive and this is the smallest competitiveness a deterministic online algorithm can achieve.*

Proof. We first analyze *ALG-D* and consider two cases. If the value of T is smaller than the value of β, then *ALG-D* consumes T units of energy during the idle interval and this is in fact equal to the consumption of OPT. If T is at least β, then *ALG-D* first consumes β energy units to remain in the active state. An additional power-up cost of β is incurred at the end of the idle interval. Hence, *ALG-D*'s total cost is 2β, while OPT incurs a cost of β for the power-up operation at the end of the idle period.

We next verify that no deterministic online algorithm can achieve a competitive ratio smaller than 2. If an algorithm transitions to the sleep state after exactly t time units, then in idle period of length t it incurs a cost of $t + \beta$ while OPT pays $\min\{t, \beta\}$ only. □