Dynamic Voltage and Cache Reconfiguration for Low Power

Andre C. Nacul and Tony Givargis
University of California, Irvine
{nacul, givargis}@ics.uci.edu

Introduction

In this work, we propose a combined *Dynamic Voltage Scaling* (DVS) and *Dynamic Cache Reconfiguration* (DCR) online algorithm that dynamically adapts the processor speed (i.e., voltage) and the cache subsystem to the workload requirements for the purposes of saving energy. The workload is considered to be a set of tasks with real-time deadlines. Our online algorithm is invoked as part of the OS scheduler, which performs standard earliest deadline first (EDF) task scheduling first. Then, our online algorithm, determines an ideal voltage/cache configuration for the current executing task.

Related Work

An extensive analysis of related work can be found in [1].

Dynamic Voltage Scaling (DVS) is an approach for power reduction that has gained much attention in the recent years. Power reduction is often accomplished by appropriately scheduling tasks and selecting voltage settings that eliminate the slack [2].

A great amount of previous work has also shown that statically tuning the cache subsystem to the running task can result in significant energy savings. Malik et al. [3] have shown that the best cache configuration depends heavily on the particular running task. Likewise, Zhang et al. [4] analysis shows that having a dynamically configurable line size architecture can have a significant (up to 50%) energy saving potential in embedded systems.

Problem Formulation

We assume a system composed of *N non-preemptive* tasks, T_1, T_2 … T_n. Each task T_i has a deadline D_i and a period P_i. One of the tasks is the scheduler T_s. The scheduler selects the next task T_j to be executed based on EDF. Then, our online algorithm, running as part of the scheduler, selects a system configuration that maintains the timing of the task T_j while saving as much energy as possible.

The platform's cache subsystem has a finite number of possible configurations C_1, C_2 … C_n, each one different than any other by at least one of the configurable parameters: *cache size*, *line size* or *cache associativity*. Among all valid configurations, one of them is the so-called reference configuration C_r. The voltage of the platform can also be set to one of a finite set of voltages V_1, V_2 … V_n. A reduction in voltage directly affects the operating frequency of the system.

We assume a time penalty as well as a power penalty for cache reconfiguration. These penalties are for writing dirty data back to memory, and are a function of the current configuration C_i and the new configuration C_j. The functions can be either hard coded statically, or learned by our online algorithm during run time. In a similar fashion, we assume time and power penalties for selecting a new processor operating point (i.e., voltage/speed).

Proposed Solution

Any feasible solution in this context must address a multi-objective problem: minimize power while still meeting task deadlines. In a multi-objective problem, it is usually the case that one solution is good for one objective, but not so good for the others. In the universe of different configurations, we can identify some configurations that are better than all the other ones for at least one of the objectives. These are the so-called Pareto-optimal solutions.

Assuming the exact set of Pareto-optimal voltage and cache configurations for each task are known, our online algorithm, after performing EDF, picks the Pareto-optimal configuration that best fills the slack given the next task to be executed.

The challenge, thus, is to compute the voltage and cache Pareto-optimal configurations for each task. Extensive simulations are needed in order to compute the exact Pareto-optimal set. For practical reasons, we have considered computing the Pareto-optimal sets online. However, due to computation overhead, it is not feasible to compute the exact Pareto-optimal sets. Instead, an approximation of the Pareto-optimal set is sufficient.

Given the Pareto-optimal sets (or an approximation in the online case), the system can trade-off power consumption and execution time by selecting the configuration that is best suited to fill the excess processing time (i.e., slack). The configuration selection is based on the utilization rate of the processor. The utilization rate of the processor is calculated every time a task finishes execution, or whenever a task is added or removed to and from the system.

$$util = \max_{\forall i} \left(\frac{\sum_{j=1}^{i} exec\_time_j}{deadline_i - current\_time} \right) \quad \text{Eq. (1)}$$

For the utilization calculation, the best case execution time (but not necessarily most energy efficient) of each task is used. Given this utilization rate, we calculate the target execution time for the next task T_j as shown in Equation 2.

$$target\_exec\_time_j = exec\_time_j / util \quad \text{Eq. (2)}$$

Given the target execution time, the scheduler is able to select the Pareto-optimal configuration that has a time less, but closest to the target time.

Experimental Setup

In order to evaluate the effects and benefits of our online algorithm, we have performed several simulations. We have considered different task timings and have experimented with only DVS, only DCR, and the combination of DVS and DCR.

Our simulations were performed on a target platform that is composed of a MIPS processor, unified L1 reconfigurable cache, on-chip memory, and the associated busses between the cache and the processor, as well as cache and on-chip memory. In addition, our platform includes a hardware power monitor for real-time power measurements.

The largest cache size is 32Kb, and the fastest speed setting for the platform is 400Mhz. The total number of possible system configurations is the cross-product of the parameters, resulting in 820 different valid platform configurations.

1530-1591/04 $20.00 (c) 2004 IEEE

Combining DVS and DCR

In the first set of experiments we attempted to estimate the possible savings by the combination of DVS and DCR. In order to get a more accurate estimate of the largest savings possible, the OS was fed with the actual Pareto-optimal sets. We call this an Oracle solution, since it knows the behavior of the applications beforehand.

We observed that the combination of DVS and DCR had a potential for larger savings than either of the two techniques alone. However, the effective energy savings was highly dependable on the deadlines (and so on the slack available), as expected. Table 1 summarizes the results for the different scenarios and platform configurations.

Configuration	D=19.0ms	D=20.3ms	D=22.5ms
(A) 32K,64,8,400Mhz	2.26W	2.29W	2.33W
	13.4ms	13.4ms	13.4ms
(C1) 32K,64,8,300Mhz	1.14W	1.16W	1.19W
	18ms	18ms	18ms
(D) 32K,64,8,DVS	1.04W	1.04W	0.93W
	18.9ms	19.6ms	21.9ms
(B1) 16K,16,2,400Mhz	0.65W	0.67W	0.69W
	16.3ms	16.3ms	16.3ms
(B2) 16K,16,2,330Mhz	n/a	0.51W	0.54W
		19.7ms	19.7ms
(E) 16K,16,2,DVS	0.53W	0.50W	0.44W
	18.5ms	19.2ms	22.3ms
(F1) Dynamic,400Mhz	0.55W	0.52W	0.47W
	18.7ms	20.2ms	21.7ms
(F2) Dynamic,330Mhz	0.56W	0.51W	0.45W
	18.4ms	20.2ms	22.4ms
(G) Dynamic,DVS	0.52W	0.45W	0.32W
	18.9ms	20.1ms	22.4ms

n/a = not possible to meet time constrains with the respective configuration

Table 1. Summary of experimental results.

Based on Table 1 and on the results presented in [1], we conclude that a DVS-only system performs slightly better than a DCR-only system. We also observe that the savings provided by the combination of DVS and DCR increase with larger slacks. In the 19 ms deadline scenario, there is almost no gain in adding DCR to the system (i.e., changing from E to G). However, in the 20.3 ms and 22.5 ms, there is an extra saving of 10% and 27%, respectively.

Table 1 also shows that combining DVS and DCR allows a better usage of the slack. For example, in the 20.3 ms scenario, DVS only (experiment E) can slowdown execution to 19.2 ms, leaving 1.1 ms unused. When DCR is combined to DVS (experiment G), the execution time is stretched to 20.1 ms.

Online Reconfiguration

As an alternative to pre-computing the exact Pareto-optimal sets, we introduce an online algorithm for calculating an approximation of the Pareto-optimal sets. This online algorithm uses the same scheduling algorithm discussed before. Here, the OS scheduler additionally interleaves the configuration selection with the configuration discovery algorithm. After each invocation of the scheduler, a new point may be added to the Pareto-optimal sets.

The main objective of the Pareto discovery algorithm is to converge on to a reasonable approximation of the actual Pareto-optimal set for each task. The discovery procedure starts with the reference configuration as the only member of the Pareto-optimal sets. Gradually, each of the cache size, line size, and associativity parameters are varied, individually (i.e., one change per scheduler invocation) in a greedy search, until the Pareto-optimal set converges.

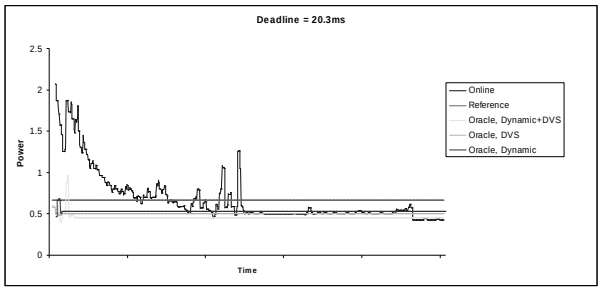

Figure 1. Online execution behavior.

Simulation Results

The power consumption results for the online approach are partially shown in Figure 1. For comparison purposes, Figure 1 includes the plots for the online system as well as the Oracle system and the reference configuration.

As expected, the online performance is slightly worse than the Oracle DCR+DVS implementation. The worst increase happened in the case when the deadline is 22.5 ms, where the power consumption increased by as much as 20%. However, savings are still higher when DVS and DCR are combined when compared to either technique alone.

The online discovery behavior can also be seen in Figure 1. Initially, the power consumption oscillates quickly, as the system discover new Pareto-optimal points. As the discovery converges, the power profile stabilizes.

Furthermore, while the system is testing new cache configurations, some deadlines are eventually lost. With deadline set to 19 ms, 10% of the deadlines are lost during discovery. On the other hand, when deadline is 22.5 ms, only 4% of the deadlines are lost during the discovery process. Clearly, the online approach is not suitable for real-time applications with strict deadlines. In the hard real-time instances, the static approach to discovering Pareto-optimal points should be utilized

As a final remark, we observed that the discovery process requires to analyze about 60-70 platform configurations in order to converge. This is less than 10% of the 820 possible configurations when cache and voltage are combined.

Acknowledgments

This work was supported in part by a National Science Foundation Award (#0205712) and by a CAPES Foundation, Brazil scholarship (#1054015).

References

[1] A. Nacul, T. Givargis. Dynamic Voltage and Cache Reconfiguration for Low Power Systems. Technical Report CECS-03-34, November 2003.

[2] T.D. Burd, T.A. Pering, A.J. Stratakos, R.W. Brodersen. A Dynamic Voltage Scaled Microprocessor System. IEEE International Solid-State Circuits Conference, Nov. 2000.

[3] A. Malik, B. Moyer, D. Cermak. A Lower Power Unified Cache Architecture Providing Power and Performance Flexibility.

[4] C. Zhang, F. Vahid, W. Najjar. A Highly Configurable Cache Architecture for Embedded Systems. In Proceedings of International Symposium on Computer Architecture. 2003.