An Energy-Efficient Scheduling Algorithm for Real-Time Tasks

Youlin Ruan^{1,2}, Gan Liu³, Jianjun Han³, and Qinghua Li³

¹ School of Information Engineering, Wuhan University of Technology,
430070 Wuhan, P.R. China

² State Key Laboratory for Novel Software Technology, Nanjing University,
210093 Nanjing, P.R. China

³ Department of Computer Science and Technology, Huazhong University of Science and
Technology, 430074 Wuhan, P.R. China
ruanyl@126.com

Abstract. Based on maximal slack first, this paper proposes a novel energy-efficient scheduling algorithm for periodic real time tasks. The scheduling solution combines static priority and dynamic speed adjustment mechanism to save energy. Simulation results show that the proposed algorithm outperforms other major scheduling schemes.

Keywords: energy-efficient, maximal slack first, energy consumption.

1 Introduction

In recent years, several software techniques have been proposed to adjust the supply voltage. Krishna and Lee propose a power-aware scheduling technique using slack reclamation, but only in the context of systems with two voltage levels^[1]. Mosse *et al.* propose and analyze several techniques to dynamically adjust processor speed with slack reclamation^[2]. Aydin proposes a power-aware scheduling of periodic tasks to reduce CPU energy consumption in hard real-time systems through a static solution to compute optimal speed and an online speed reduction mechanism to reclaim energy^[3].

By dynamic voltage scaling, Chen proposes an optimal real-time task scheduling algorithm for multiprocessor environments with the allowance of task migration^[4]. Zhu *et al.* present two algorithms GSSR based on the concept of slack sharing for single task sets with and without precedence constraints. These scheduling techniques are based on longest task first, which reclaim the time unused by a task to reduce the execution speed of future tasks, and thus reduce the total energy consumption of the system^[5]. However, Han proposed the opposite strategy STFBA1 that is based on the policy of shortest task first and combine with other efficient techniques^[6], which believe same slack used by longer tasks can save more energy.

We aim to the strategy of tasks assignment, and how to dynamic adjust speed to save energy consumption. The rest of the paper is organized as follows. Section 2 describes the task model, energy model and power management schemes. A dynamic power aware scheduling with maximal slack first for periodic tasks is addressed in section 3. Simulation and comparison are given and analyzed in section 4.

2 System Models and Schemes

In this paper, we assume that $E = C \cdot C_{ef} \cdot \frac{S^2}{k^2}$ holds as defined in literature for simplicity[10]. We assume a frame based real-time system in which a frame of length D is executed repeatedly. A set of tasks $\Gamma = \{T_1, \dots, T_n\}$ to execute within each frame and is to complete before the end of the frame. Because of the schedule's periodicity, we consider only the problem of scheduling Γ in a single frame with deadline D .

In specifying the execution of a task T_i , we use the three tuple $(WCET_i, ACET_i, AET_i)$, where $WCET_i$ is the estimated worst case execution time (WCET), $ACET_i$ is the estimated average case execution time (ACET) and AET_i is the actual execution time (AET), which are all based on maximal processor speed. We assume that for a task T_i , the value of $WCET_i$ and $ACET_i$ are known before execution, while AET_i is determined at run time. To get maximal energy savings, we combine static priority assignment and dynamic voltage/speed adjustment. Thus, we assume that canonical execution is first checked to see whether a task set can finish before D or not. If not, the task set is rejected; otherwise, our algorithm then apply static priority assignment and dynamic voltage/speed adjustment to save energy.

3 Maximal Slack First-Based Algorithm

Zhu propose global scheduling with longer tasks high priority, which believes longer tasks can generate more dynamic slack during execution that can be used by shorter tasks, thereby, it will reduce the energy consumption. However, it is not always true. Thus, Han propose an opposite strategy that is based on shortest task first, which believe same slack used by longer tasks can save more energy. In fact, such two strategies are not always efficient under all conditions, which fit for different aspects respectively. In the following, we will point out that both strategies are not always the most efficient heuristic to save energy consumption.

Lemma 1. Assuming that there exist two tasks T_i and T_j , the $WCET$ and $ACET$ of T_i are x and $x-a$ respectively, the $WCET$ and $ACET$ of T_j are y and $x-a$ respectively, and $x > y$. The energy consumption generated by the execution order in which T_j is prior to T_i is higher than that generated by the reverse execution order.

Proof. We assume that the energy generated by small slack first heuristic is no longer than that generated by large slack first heuristic, then

$$(x-a)C_{ef} \frac{1}{k^2} + (x-a)C_{ef} \left(\frac{x}{y+a}\right)^2 \frac{1}{k^2} \leq (x-a)C_{ef} \frac{1}{k^2} + (x-a)C_{ef} \left(\frac{y}{y+a}\right)^2 \frac{1}{k^2}$$

$$\Leftrightarrow (x-a) + (x-a)\left(\frac{x}{y+a}\right)^2 \leq (x-a) + (x-a)\left(\frac{y}{y+a}\right)^2$$

Since $x > y > a > 0$ hold, then $\left(\frac{x}{y+a}\right)^2 > \left(\frac{y}{y+a}\right)^2$, the left part of above inequality is

not true. Hence it leads to contradiction.

Lemma 2. Assuming that there exist two tasks T_i and T_j , the $WCET$ and $ACET$ of T_i are x and $y-a$ respectively, the $WCET$ and $ACET$ of T_j are y and $x-a$ respectively, and

$x > y$. The energy consumption generated by the execution order in which T_j is prior to T_i is higher than that generated by the reverse execution order.

Proof. We assume that the energy generated by small slack first heuristic is no longer than that generated by large slack first heuristic, then

$$(x-a)C_{ef} \frac{1}{k^2} + (y-a)C_{ef} \left(\frac{x}{y+a}\right)^2 \frac{1}{k^2} \leq (y-a)C_{ef} \frac{1}{k^2} + (x-a)C_{ef} \left(\frac{y}{x+a}\right)^2 \frac{1}{k^2}$$

$$\Leftrightarrow (x-a) + (y-a)\left(\frac{x}{y+a}\right)^2 \leq (y-a) + (x-a)\left(\frac{y}{x+a}\right)^2$$

$$\Leftrightarrow (x-a)\left[1 - \left(\frac{y}{x+a}\right)^2\right] \leq (y-a)\left[1 - \left(\frac{x}{y+a}\right)^2\right]$$

Since $x > y > a > 0$ hold, then $(x-a) > (y-a)$ and $\left(\frac{x}{y+a}\right)^2 > \left(\frac{y}{x+a}\right)^2$,

$\left[1 - \left(\frac{y}{x+a}\right)^2\right] > \left[1 - \left(\frac{x}{y+a}\right)^2\right]$, the left part of above inequality is not true. Hence it leads to contradiction

Thus, we can see that both longest task first and shortest slack first are not always efficient strategies. The key problem to save energy is the size of slack used by the following tasks. Therefore, we can draw a conclusion that the strategy based on maximal slack first. The longer slack of task generates, the higher priority of the task has. If the two slacks generated by two tasks are equal, shortest task first can save more energy. Moreover, longer tasks may generate more dynamic slack during execution. Thus, we use the longest task first heuristic to determine task's priority when the two slacks are 0, which may save more energy consumption. Thus, we propose the MSF algorithm, which combine the strategy of maximal slack first and the concept of dynamic adjusting of speed of tasks. The MSF algorithm comprises of two parts: static algorithm and dynamic algorithm. Static algorithm computes the static slack of each task offline and determines the priorities of tasks. Dynamic algorithm do not modify priorities of tasks, but adjust speed of tasks according to the actual execution time and dynamic slack, which is shown in following.

Dynamic Algorithm of MSF

```
{ slack=0; k=1;
  while Ready-Q <> ∅ Do //Tasks Γ = {T1, ..., Ts}
    Tk = Dequeue(Ready-Q);
    Sk = Smax *  $\frac{WCET_k}{WCET_k + slack}$ ; Execute Tk at speed Sk;
    ETk = AETk / Sk; slack = slack + WCETk - ETk; k=k+1;
  endwhile }
```

4 Experiment Results

In this section, we present results of simulations performed to compare three algorithms MSF, GSSR and STFBA1. We define α_i as average/worst case ratio for T_i^* s

execution time, the actual execution time Z_i will be generated as a normal distribution around $\alpha_i \cdot WCET_i$. Let $U(i, j)$ be a uniformly distributed integer in the range of $[i, j]$, the number of tasks is $U(40, 400)$, the WCET of each task is $U(20, 200)$. The value of α is 0.1-1.0. Regardless of α_i of all tasks are same and different, we can see that MSF consume less energy than GSSR and STFBA1 from Fig.1.

Fig. 1. Comparison of energy savings

Acknowledgments. This work has been supported by the national natural science foundation of china under grant no. 60503048 and 60672059, the 863 program no. 2006AA01Z233.

References

1. Krishna C.M., Lee Y.H.: Voltage Clock Scaling Adaptive Scheduling Techniques or Low Power in Hard Real-Time Systems. Proc.6th IEEE Real-Time Technology and Applications Symp.(2000)
2. Mosse D., et al.: Compiler-Assisted Dynamic Power-Aware Scheduling for Real-Time Applications. Proc.Workshop Compiler and OS for Low Power (2000)
3. Hakan Aydin, Rami Melhem, Daniel Mosse.: Power-Aware Scheduling for Periodic Real-Time Tasks, IEEE Transaction on Computer. 5(2004)584–600
4. Chen J.J, Kuo T.W. :Multiprocessor Energy-Efficient Scheduling for Real-Time Tasks with Different Power Characteristics. Proc. 2005 International Conference on parallel Processing (2005)
5. Zhu D, Rami Melhem, Bruce Childers. :Scheduling with Dynamic Voltage/Speed Adjustment Using Slack Reclamation in Multiprocessor Real-Time Systems. IEEE Transaction on Parallel and Distributed Systems, 7 (2003)686–699.
6. Han J.J, Li Q.H. :Dynamic Power-Aware Scheduling Algorithms for Real-Time Task Sets with Fault-Tolerance in Parallel and Distributed Computing Environment, Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (2005)