Energy-Aware Scheduling for Real-Time Multiprocessor Systems with Uncertain Task Execution Time

Changjiu Xian
Department of Computer Science
Purdue University
West Lafayette, Indiana
cjx@cs.purdue.edu

Yung-Hsiang Lu
School of Electrical and Computer Engineering
Purdue University
West Lafayette, Indiana
yunglu@purdue.edu

Zhiyuan Li
Department of Computer Science
Purdue University
West Lafayette, Indiana
li@cs.purdue.edu

ABSTRACT

This paper presents an energy-aware method to schedule multiple real-time tasks in multiprocessor systems that support dynamic voltage scaling (DVS). The key difference from existing approaches is that we consider the probabilistic distributions of the tasks' execution time to partition the workload for better energy reduction. We analyze the problem of energy-aware scheduling for multiprocessor with probabilistic workload information and derive its mathematical formulation. As the problem is NP-hard, we present a polynomial-time heuristic method to transform the problem into a probability-based load balancing problem that is then solved with worst-fit decreasing bin-packing heuristic. Simulation results with synthetic, multimedia, and stereo-vision tasks show that our method saves significantly more energy than existing methods.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: design studies

General Terms

Design, Performance

Keywords

Dynamic Voltage Scaling, Multiprocessor, probability

1. INTRODUCTION

Energy consumption is an important design issue for battery-operated embedded systems. In these systems, the processor is a major energy consumer. Embedded systems often run tasks with real-time constraints. Since dynamic voltage scaling (DVS) can achieve quadratic energy savings with only linear decrease of the processor's speed, combining DVS with real-time scheduling has been extensively studied [9] [16] [18] [20] [21]. For real-time tasks with uncertain execution time, the worst-case execution time (WCET) must be considered for meeting the tasks' deadlines. Since this paper considers frequency and voltage scaling, we use "execution cycles" instead of execution time to express the workload of a task. The task's execution time is the execution cycles divided by the processor's frequency.

While most studies on DVS and real-time scheduling are for a single processor, today's embedded systems are increasingly based on multiprocessors for higher performance and lower power consumption. Energy-efficient task scheduling on multiprocessors therefore becomes an important issue. Existing studies on multiprocessor real-time scheduling take two major approaches: (a) partitioning— each task is assigned to a particular processor permanently, and (b) dynamic scheduling— a global scheduler selects tasks from a single ready queue to execute and tasks can migrate among processors. Either approach has its advantage and disadvantage [15]. In current multiprocessor systems, the partitioning approach is more common because of its simplicity and ease of implementation [3]. In this paper, we improve the energy efficiency under the partitioning approach.

Previous studies have shown that, with DVS, the total energy consumption of the multiprocessor is minimized when the workload is balanced among the processors [4] [7]. This is because of the convex relationship between the processor speed and the power consumption. These previous studies take the worst-case execution cycles (WCEC) of tasks as the workloads. However, if some tasks demand much fewer execution cycles in most cases than in their worst cases, the workloads may be poorly balanced. Even though the WCEC must be considered to guarantee meeting deadlines, the statistical distribution of the execution cycles should also be considered for better balancing. Such statistical information can be obtained through offline or online profiling [21].

In this paper, we address the problem of energy-aware scheduling for multiprocessor with the information of probabilistic distributions. We consider multiple periodic real-time tasks that execute on a set of identical processors. We derive the mathematical formulation for minimizing the expected total energy consumption while meeting the deadlines of all tasks with earliest deadline first (EDF) scheduling. As this problem is NP-hard, we present a polynomial-time heuristic method. First, we transform the problem to the load-balancing problem based on the cycle distributions of workloads, assuming that the processors have unbounded and continuous range of frequencies to choose from. The worst-fit decreasing bin packing heuristic is used to balance the load. Second, we modify the solution by including the maximum frequency as a new constraint and by assuming

DAC 2007, June 4-8, 2007, San Diego, California, USA.

bounded discrete frequencies. Our simulation results with synthetic, multimedia, and stereovision task sets show that our method saves significantly more energy than the workload balancing methods that are based on the worst case.

2. BACKGROUND

2.1 Probability-Based DVS

The probabilistic distribution of cycle demands has been studied to construct frequency schedules for a single processor. Lorch et al. [16] and Gruian [9] derive accelerating frequency schedules based on probability information. Yuan et al. [21] implement the accelerating scheduling in practical systems and show that the probability information can be obtained through low-overhead run-time profiling. These studies use the probability information for only intra-task scheduling. Zhang et al. [22] consider probability-based scheduling for multiple periodic tasks when they share a common period. Xian et al. further consider multiple tasks with different periods on a single processor [19].

2.2 DVS for Multiprocessor

Several previous studies have considered energy-efficient multiprocessor scheduling with DVS. Zhu et al. [23] propose a run-time scheme of slack reclamation for tasks sharing a single, global deadline. Aydin et al. [4] partition periodic real-time tasks with EDF using worst-case workload information. AlEnawy et al. [3] study the partitioning for tasks that are assigned rate-monotonic priorities. Kadayif et al. [2] consider the leakage power of multiprocessors and determine the number of processors needed for executing array-intensive applications. Chen et al. [7] propose an approximation algorithm to partition tasks with different power consumption functions. Juang et al. [13] show a coordinated DVS scheme for chip multiprocessor. None of these studies have taken advantage of the probabilistic workload information.

2.3 Statistically Optimal Frequency Schedule

Suppose there is a single processor and n periodic tasks. Let K_i be the i^{th} task. K_i's period (also the deadline) is T_i and the execution instance in each period consumes at most c_i cycles. The distribution of cycle demands is expressed by the cumulative distribution function (CDF), denoted as Ψ. Since each task may demand millions of cycles, Yuan et al. [21] suggest dividing the range $[0, c_i]$ into m_i bins for practical implementations. Each bin contains an equal number of cycles $b_i = \lceil \frac{c_i}{m_i} \rceil$. Ψ is then a function of bins. The probability that K_i consumes the j^{th} bin of cycles is $1 - \Psi_i(j-1)$. Note that $\Psi_i(0) = 0$.

Let f_{ij} be the frequency assigned to the j^{th} bin of K_i. The time for this bin is then $\frac{b_i}{f_{ij}}$. The processor's power is proportional to $v^2 f$ and $v \propto f$ (v: voltage). The energy for this bin is $(v_{ij}^2 f_{ij}) \times \frac{b_i}{f_{ij}} \propto b_i f_{ij}^2$, and the expected energy consumption is proportional to the product of the energy and the probability: $b_i f_{ij}^2 \frac{1-\Psi_i(j-1)}{T_i}$. The probability is divided by T_i because the probability is with respect to K_i's total number of instances and it is proportional to $\frac{1}{T_i}$ for a given duration. The optimization problem is to find f_{ij} such that the total expected energy consumption by all tasks is minimized while satisfying the schedulability constraint of EDF.

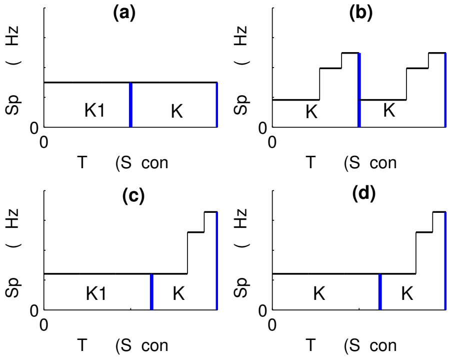

Figure 1: Frequency schedules from different partitioning methods.

EDF can schedule a set of tasks if the total CPU utilization is no more than one [14]. The mathematic formulation is

$$\text{minimize} \sum_{i=1}^{n} \sum_{j=1}^{m_i} b_i f_{ij}^2 \frac{1 - \Psi_i(j-1)}{T_i} \tag{1}$$

$$\text{subject to} \sum_{i=1}^{n} \frac{\sum_{j=1}^{m_i} \frac{b_i}{f_{ij}}}{T_i} \leq 1 \tag{2}$$

Equation (1) is the total expected energy from all tasks. Equation (2) guarantees the schedulability of EDF. Xian et al. [19] show that, assuming the processor has unbounded continuous frequencies, the minimum expected total energy equals Q^3, where Q is as follows.

$$\sum_{i=1}^{n} \frac{b_i}{T_i} \sum_{j=1}^{m_i} \sqrt[3]{1 - \Psi_i(j-1)} \tag{3}$$

The minimum is achieved with the frequency schedule:

$$f_{ij} = \frac{Q}{\sqrt[3]{1 - \Psi_i(j-1)}} \tag{4}$$

Since $1 - \Psi_i(j-1)$ decreases as j increases, the frequency f_{ij} increases for the later bins in the same task.

3. MULTIPROCESSOR SCHEDULING

This section presents our scheduling scheme for multiprocessor systems. We first provide a motivational example to illustrate the basic concept. Then we formulate the multiprocessor scheduling problem using probability information. As the problem is NP-hard, we present a polynomial-time heuristic algorithm in Section 3.4.

3.1 Motivational Example

Consider four periodic tasks K_1, K_2, K_3, and K_4 (as shown in Figure 1), starting at the same time and all have the same period of 2 seconds. Each task's deadline is the same as the task's period. All tasks have WCEC of 3 million cycles, divided into 3 equal-sized bins. Tasks K_1 and K_2 always consume 3 million cycles (3 bins). Tasks K_3 and K_4 have identical probability distribution: 90% probability to consume 1 million cycles (the first bin), 5% probability to consume 2 million cycles (the first two bins), and 5% probability to consume 3 million cycles (all three bins).

We compare two methods to partition the four tasks between two identical processors. The first method assigns tasks K_1 and K_2 to one processor and tasks K_3 and K_4 to the other. The frequency schedules of these tasks on each processors are calculated using Equation (4), as shown in Figure 1 (a) and (b). Both processors have the same worst-case workloads ($3 + 3 = 6$ millions cycles) in every two-second interval. However, considering the probabilistic distributions of the tasks' cycle demands, K_3 and K_4 consume much fewer cycles in most cases than K_1 and K_2.

Our method assigns K_1 and K_3 to one processor and K_2 and K_4 to the other. Since K_1 is identical to K_2 and K_3 is identical to K_4, the workloads on the two processors have much higher probability of being balanced than the previous method. The new frequency schedules of the two processors are shown in Figure 1 (c) and (d). Calculated by Equation (3), the new method saves 15% more energy than the first method. This example shows the importance of considering probability information to balance workloads.

3.2 Task and System Model

We consider a set of periodic, preemptive, and hard real-time tasks $K = \{K_1, K_2, ..., K_n\}$. All tasks are independent. Task K_i has the following parameters: (a) T_i is its period and also the deadline. The task has one execution instance per period. (b) c_i is its WCEC, i.e., the worst-case number of cycles needed by one execution instance. The range $[0, c_i]$ is equally divided into m_i bins and each contains b_i cycles [21]. (c) Ψ_i is the CDF of K_i's bins and $1 - \Psi_i(j-1)$ is the probability that the j^{th} bin of cycles is consumed.

The tasks are partitioned among a set of identical processors $C = \{C_1, ..., C_l\}$. Each processor has a limited range of discrete frequencies, denoted as $F = \{f_1, f_2, ..., f_h\}$ in ascending order. The frequency-switch time [21] and context-switch time [1] are in the microseconds range and the execution time for tasks is usually in the milliseconds range [22]. Hence, we ignore the frequency-switch time and the context-switch time. We assume the processors consume both dynamic and static power during busy periods and only static power during idle periods. The dynamic power is proportional to $v^2 f$ and the static power is a constant. We assume DVS can adjust only dynamic power so we ignore the static power because it is constant throughout the whole duration. Overall, the energy per cycle is proportional to $v^2 f \frac{1}{f} \propto f^2$ since $v \propto f$. The scheduling is based on EDF.

3.3 Problem Formulation

We address the following probability-based energy-aware scheduling problem: Given the task and the system model as described in Section 3.2, partition the set of tasks K among the set of processors C and compute the frequency for each bin of each task based on the bins' probabilities, such that (a) the tasks assigned to each processor can be scheduled using EDF, and (b) the total expected energy consumption of all processors is the minimum among all the feasible assignments. As explained in Section 1, we do not consider task migration among processors in this paper.

Let S_r $(r = 1, 2, ..., l)$ be the set of tasks assigned to processor C_r. Then the mathematical formulation of the problem is as follows.

$$\text{minimize} \quad \sum_{r=1}^{l} \sum_{\substack{K_i \in S_r \\ 1\le i\le n}} \sum_{j=1}^{m_i} b_i f_{ij}^2 \frac{1-\Psi_i(j-1)}{T_i} \tag{5}$$

$$\text{subject to} \sum_{\substack{K_i \in S_r \\ 1\le i\le n}} \frac{\sum_{j=1}^{m_i} \frac{b_i}{f_{ij}}}{T_i} \le 1, \; r = 1, ..., l \tag{6}$$

$$\bigcup_{r=1}^{l} S_r = K \tag{7}$$

$$S_g \cap S_h = \emptyset, g \ne h, 1 \le g, h \le l \tag{8}$$

Equation (5) is the total expected energy consumption of all tasks on all processors, where $b_i f_{ij}^2 \frac{1-\Psi_i(j-1)}{T_i}$ is the expected energy consumption of the j^{th} bin of task K_i and $\sum_{j=1}^{m_i} b_i f_{ij}^2 \frac{1-\Psi_i(j-1)}{T_i}$ is total expected energy consumption of K_i. Equation (6) is EDF's schedulability constraint, where $\frac{\sum_{j=1}^{m_i} \frac{b}{f_{ij}}}{T_i}$ is the worst-case processor utilization of K_i. For each processor the worst-case utilization of the assigned task set should be no more than one. This constraint gurantees that all tasks meet their deadlines. Equation (7) guarantees that all tasks are assigned. Equation (8) guarantees that every task is assigned to only one processor, i.e., for any two different processors, the intersection of their task sets should be empty. This problem is NP-Hard because the POWER-PARTITION [4] can be reduced to this problem. The proof of the NP-Hardness is omitted due to space limit.

3.4 Polynomial-Time Heuristic Algorithm

In this section, we first consider processors with continuous frequencies between 0 and infinity such that the problem can be transformed into a load balancing problem based on the tasks' probability information. For processors with bounded discrete frequencies, we further modify the solution to consider the maximum frequency as a constraint.

3.4.1 Unbounded Continuous Frequencies

The statistically minimal energy consumption (Equation (3)) of a single processor with a given task set is found by our previous study [19]. Based on Equations (1), (2), and (3), Equations (5) and (6) can be combined into the following equation.

$$\text{minimize} \quad \sum_{r=1}^{l} \left(\sum_{\substack{K_i \in S_r \\ 1\le i\le n}} \frac{b_i}{T_i} \sum_{j=1}^{m_i} \sqrt[3]{1-\Psi_i(j-1)} \right)^3 \tag{9}$$

This optimization problem can be transformed into a load balancing problem. Let $Q_i = \frac{b_i}{T_i} \sum_{j=1}^{m_i} \sqrt[3]{1-\Psi_i(j-1)}$. Equation (9) can be rewritten as $\sum_{r=1}^{l} (\sum_{K_i \in S_r} Q_i)^3$. The Q_i value of task K_i is independent of task partitioning. Let $Q_{C_r} = \sum_{K_i \in S_r} Q_i$. Since $\sum_{r=1}^{l} Q_{C_r} = \sum_{i=1}^{n} Q_i$ and from Jensen's Inequality [12], we have

$$\sum_{r=1}^{l} Q_{C_r}^3 \ge l (\frac{\sum_{i=1}^{n} Q_i}{l})^3 \tag{10}$$

This states that the total energy is minimized if $Q_{C_1} = Q_{C_2} = ... = Q_{C_l} = \frac{\sum_{i=1}^{n} Q_i}{l}$. Since Q_is may have different values and each individual Q_i is indivisible, it is not always possible to make all Q_{C_r} equal to the constant $\frac{\sum_{i=1}^{n} Q_i}{l}$.

	T_i (ms)	$\Psi_i(1) \sim \Psi_i(4)$ (%)	Q_i(GHz)	U_i
K_1	45	5, 15, 25, 100	0.40	0.59
K_2	35	85, 90, 95, 100	0.31	0.76
K_3	70	35, 50, 65, 100	0.22	0.38
K_4	85	80, 90, 95, 100	0.13	0.31
K_5	95	90, 95, 99, 100	0.10	0.28

Table 1: Example task set. Each task requires four million cycles divided into four equal-sized bins.

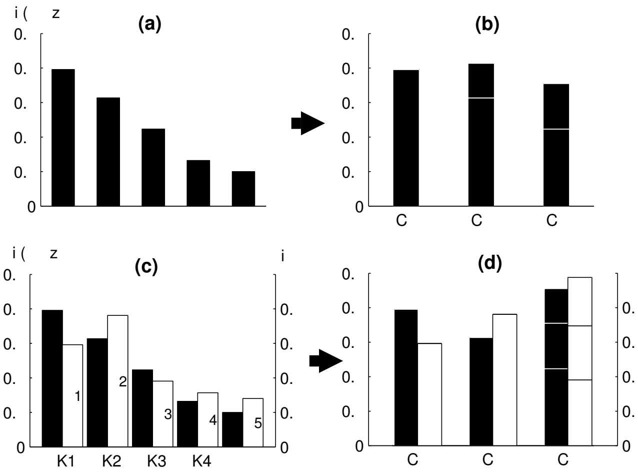

Figure 2: Partitioning five tasks among three processors: (a)(b) Considering only Q_i for unbounded frequencies. (c)(d) Considering both Q_i and U_i for bounded frequencies.

Hence, we minimize the total expected energy by making Q_{C_r} as balanced as possible.

We use the worst-fit decreasing (WFD) bin-packing algorithm to balance Q_{C_r} because WFD has been shown to be the best bin-packing heuristic for balancing loads among multiple bins [4]. With WFD, all tasks are sorted by Q_i descendingly before packing. Figure 2 (a) shows that the five tasks described in Table 1 are sorted by Q_i in descending order. Then the tasks are sequentially assigned to the processors. Each task is assigned to the processor that has the least Q_{C_r} before assigning the task. The packing results are shown in Figure 2 (b) where the packing order is Q_1, Q_2, Q_3, Q_4, and Q_5.

3.4.2 Bounded Discrete Frequencies

In order to schedule the task set assigned to each processor with the frequency set $F = \{f_1, f_2, ..., f_h\}$ (in ascending order), the partitioning method must satisfy the following constraint: for each processor the worst-case utilization of its assigned task set at the maximum frequency f_h should be no more than one, i.e.,

$$\sum_{\substack{K_i \in S_r \\ 1 \le i \le n}} \frac{c_i/f_h}{T_i} \le 1 \tag{11}$$

Here $\frac{c_i/f_h}{T_i}$ is task K_i's worst case processor utilization at frequency f_h. This constraint means that the task set assigned to processor C_r should be schedulable at least when all bins of the tasks use the maximum frequency f_h.

The previous section shows that Q_{C_r} should be made as balanced as possible to minimize the total energy. In this section, we balance Q_{C_r} under constraint (11). We sort the tasks by their values of Q_i in descending order, as shown in Figure 2 (c), where $U_i = \frac{c_i/f_h}{T_i}$. The tasks are then sequentially assigned to the processors. When assigning task K_i to a processor, we pack both Q_i and U_i into the processor, as shown in Figure 2 (d).

Each task should be assigned to the processor with the least Q_{C_r} as explained in previous section. However, this assignment may not be feasible considering the constraint in Equation (11). In this case, we sort all processors by their Q_{C_r} in ascending order and sequentially search for the first processor where assigning the task is feasible. For example, after assigning K_1, K_2, and K_3, we need to assign K_4. At this moment, processor C_3 has the least Q_{C_r} (only Q_3 in C_3). Meanwhile, adding U_4 to U_3 is less than one so assigning K_4 to C_3 is feasible. When assigning K_5, C_2 has the least Q_{C_r} (only Q_2 in C_2). However, adding U_5 to U_2 exceeds one and it is thus infeasible to assign K_5 to C_2. Processor C_3 has the second least Q_{C_r} at this moment. Meanwhile, adding U_5 to U_3 and U_4 is less than one. Hence, K_5 is assigned to C_3. This packing method makes Q_{C_r} as balanced as possible under the constraint.

After the partitioning, the frequency schedule of each processor is calculated using Equation (4). Then we use the following method to restrict the frequency schedule to the set F. We reassign f_h to those bins with frequencies higher than f_h and reassign f_1 to those bins with frequencies lower than f_1. For any other bin with a frequency not in the set F, we replace the frequency using its two adjacent frequencies in F, as proposed in [11]. Based on these frequency assignments, we calculate the worst-case utilization ($\sum_{K_i \in S_r} \frac{\sum_{j=1}^{m_i} \frac{b}{f_{ij}}}{T_i}$) for the processor. If the utilization is larger than one, we need to raise the frequencies of some bins to reduce the utilization. Since raising frequency increases energy consumption, we should raise the frequencies of those bins with lower probabilities. We first sort the tasks' bins by their probabilities ascendingly. Then we sequentially raise the bins' frequencies to f_h until the utilization decreases to be no more than one. This can be achieved because our partitioning method satisfies the constraint in Equation (11).

Let $m = \sum_{i=1}^{n} m_i$. It is the total number of bins of all tasks. Calculating Q_is and U_is takes $O(m)$ and $O(n)$ time, respectively. When assigning a task, sorting the processors by their Q_{C_r} takes $O(l \log l)$ time and sequentially searching for the first processor that allows a feasible assignment takes $O(l)$ time. Assigning all tasks takes $O(n(l \log l + l))$ time. After partitioning, searching for the two adjacent frequencies for all bins' frequencies takes $O(m \log h)$ time (assuming binary search). Finally, sorting the tasks' bins by their probabilities takes $O(m \log m)$ time and sequentially raising frequencies takes $O(m)$ time. Overall, our algorithm takes $O(n(l \log l + l) + m(\log h + \log m))$ time.

4. SIMULATION RESULTS

This section describes the simulation setup, and presents the results comparing the energy savings from our method and three existing solutions.

Frequency(MHz)	150	400	600	800	1000
Voltage(V)	0.75	1.0	1.3	1.6	1.8
Power(mW)	80	170	400	900	1600

Table 2: XScale's frequency/voltage and power.

Application	Description	$T(ms)$	W^*
mpegplay	MPEG video decoder	30	10,500
madplay	MP3 audio decoder	30	899
tmn	H263 video encoder	400	165,000
tmndec	H263 video decoder	30	12,700
toast	GSM speech encoder	25	240
adpcm	ADPCM speech encoder	80	6,816

Table 3: Multimedia Applications. * in 10^3 cycles

4.1 Setup

We use the frequency/voltage settings and power consumption of Intel XScale [20] (Table 2) for each processor. We use synthetic, multimedia, and stereovision tasks as benchmarks.

The synthetic task set has 30 tasks and is constructed in three steps. The first step assigns each task a period randomly chosen between 10 milliseconds and 10 second. The second step chooses for each task its WCEC randomly between 100,000 cycles and 500,000,000 cycles with the constraint that the task set is schedulable for two 1-GHz processors with the WFD heuristic. The third step determines a distribution function of cycle demands for each task. We consider two types of distributions, Gaussian and exponential, for cycle demands as suggested in [20,22]. For Gaussian distributions, task K_i's mean μ_i is randomly chosen within the range $(0, c_i]$ and the standard deviation is $\sigma_i = c_i/6$. Exponential distribution has one parameter λ ($\mu = \frac{1}{\lambda}$, $\sigma^2 = \frac{1}{\lambda^2}$). We choose for each task its μ_i (or $\frac{1}{\lambda_i}$) also randomly within the range $(0, c_i]$.

The multimedia task set contains six programs: `mpegplay`, `madplay`, `tmn`, `tmndec`, `toast`, and `adpcm`, as shown in Table 3. The distributions of their cycle demands are obtained by profiling offline traces. The distributions of these programs are also studied in [21].

The stereovision task set is based on the stereovision system that guides a robot's motion in real-time [6]. Previous studies in [17] show that the processor power of a mobile robot is comparable to its motion power. When the robot is exploring an environment, it periodically takes stereo photos and processes them with stereo algorithms to detect or recognize the surrounding objects. The image processing should finish before taking the next pair of photos. A robot can have multiple cameras for detecting the environment. The typical lens for a camera has a view angle of about 45^o [10]. We consider a robot with eight pairs of cameras facing the eight directions to cover the 360^o panorama. The stereo photos are taken every two seconds and are processed using the stereo algorithm implemented in [5]. The cycle distribution of the stereo algorithm is profiled using the stereovision images taken from the image database of the city of West Lafayette and Indianapolis in the state of Indiana [8]. Figure 3 shows the distribution of the needed cycles for running the algorithm on 700 pairs of images. Note that there is great potential for energy savings as the probability of the WCEC, p(WCEC), is only 0.136%.

We compare four methods to partition workloads and determine execution speeds. *WP0* : Tasks are partitioned based on their worst-case processor utilization and all tasks are assigned a uniform speed equal to the total CPU demand in the worst-case. We use this method as the baseline to compare the energy savings from the other three methods. *WP1* : Partitioning considers the worst-case and the execution speed is determined by runtime slack reclamation [18]. *WP2* : Partitioning still considers the worst case and the probability-based speed scheduling for uniprocessor [19] is adopted. *PP* : Tasks are partitioned and assigned speeds based on their probability information, as explained in Section 3.4. This is our method.

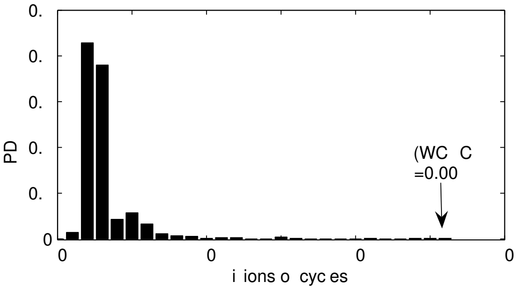

Figure 3: Profiled probability density function (PDF) of cycles for stereovision computations.

4.2 Energy Savings

Figure 4 shows the simulation results. Figures 4 (a), (b), (c) and (d) show the energy savings from the four task sets. The energy savings from methods *WP1*, *WP2*, and *PP* with respect to *WP0* are computed for different numbers of processors ranging from 2 to the total number of tasks.

Figure 4 shows that *WP1* always saves energy (ranging from 12% to 30%) with respect to *WP0.* The reason is that: (a) The partitioning results from *WP1* and *WP0* are the same. (b) *WP0* uses a uniform speed for all tasks while *WP1* can further lower down this speed by reusing the slack time upon early completion of task instances. Figure 4 also show that *WP2* saves more energy (ranging from 3% to 17%) than *WP1.* The is because *WP2* uses probability-based DVS that proactively minimizes the total expected energy while *WP1* reduces speed only when observing a slack at runtime. Probability-based DVS saves more energy.

Our method *PP* saves additional energy compared to *WP2*, up to 15.5%, 19.0%, 9.1%, and 14.6% for the four task sets. This is because *PP* performs probability-based workload balancing that is closely coupled with the probability-based DVS for individual processors. For the multimedia task set, *PP* achieves maximum additional energy savings for 2 processors and the savings decrease as the number of processors increases. The reason is that there are only 6 tasks and the task assignments from *PP* and *WP2* have less difference when the number of processors approaches 6. For 6 processors, both methods assign each processor a single task so their energy savings are the same. For the stereovision task set and the two synthetic task sets, the difference between *PP* and *WP2* first increases and then decreases as the number of processors increases. This is explained as follows. *WP2* balances the workloads based on U_i instead of Q_i. The effect is that Q_i is somewhat randomly packed into the processors. When the average number of tasks per processor is large (e.g., 30 tasks for 2 processors), the difference between

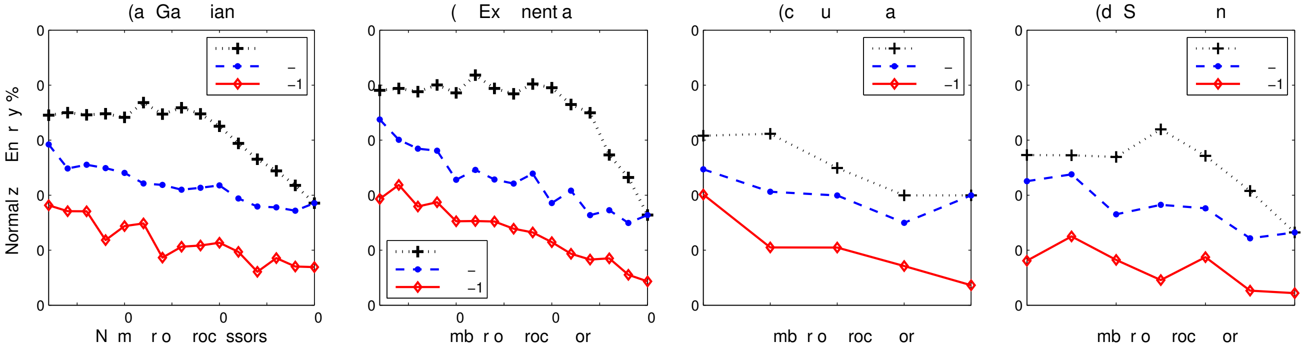

Figure 4: Energy savings from the four task sets: Gaussian, Exponential, multimedia, and stereovision. The energy savings from methods *PP*, *WP1*, and *WP2* are with respect to the method *WP0*.

the processors' Q_{C_r} is relatively small compared to the average of Q_{C_r}. Consequently, *WP2*'s energy savings are close to *PP* (*PP* balances the processors' Q_{C_r}). When the number of tasks per processor is relatively small (e.g., 30 tasks for 15 processors), the difference between the processors' Q_{C_r} becomes more significant compared to the average. In this case, *PP* saves significantly more energy by balancing the Q_{C_r}. However, if the number of tasks per processor is too small, (e.g., 30 tasks for 25 processors), most processors are assigned only a single task and the task assignments from *WP2* and *PP* have little difference so their energy savings become closer to each other.

5. CONCLUSIONS

This paper presents an energy-aware scheme for task partitioning in multiprocessor systems with uncertain workloads under hard real-time constraints. We achieve better workload balancing among multiple processors by utilizing the probabilistic distributions of the tasks' execution cycles. Our evaluation shows that our method outperforms the existing solutions for multimedia applications, stereovision tasks, and synthetic workloads. The future extension of this work will consider tasks with dependencies. Both dynamic and static power will be reduced by combining DVS and shutdown strategies.

6. ACKNOWLEDGMENTS

This work is supported in part by National Science Foundation Career CNS-0347466 and CCF-0541267. Any opinions, findings, and conclusions or recommendations are those of the authors and do not necessarily reflect the views of the sponsors.

7. REFERENCES

[1] S. Wang, S. Kodase, K. G. Shin, and D. L. Kiskis. Measurement of OS Services and Its Application to Performance Modeling and Analysis of Integrated Embedded Software. In *IEEE RTAS*, pp. 113–122, 2002.

[2] I. Kadayif, M. Kandemir, and U. Sezer. An Integer Linear Programming Based Approach for Parallelizing Applications in On-Chip Multiprocessors. In *DAC*, pp. 703-708, June 2002.

[3] T. A. AlEnawy and H. Aydin. Energy-Aware Task Allocation for Rate Monotonic Scheduling. In *IEEE RTETAS*, pp. 213–223, March 2005.

[4] H. Aydin and Q. Yang. Energy-aware Partitioning for Multiprocessor Real-time Systems. In *IPDPS*, pp. 113–121, 2003.

[5] S. Birchfield and C. Tomasi. Depth Discontinuities by Pixel-to-Pixel Stereo. *International Journal of Computer Vision*, 35(3):269–293, December 1999.

[6] A. Broggi, C. Caraffi, R. I. Fedriga, and P. Grisleri. Obstacle Detection with Stereo Vision for off-road Vehicle Navigation. In *IEEE Workshop on Machine Vision for Intelligent Vehicles*, June 2005.

[7] J.-J. Chen and T.-W. Kuo. Energy-Efficient Scheduling of Periodic Real-Time Tasks over Homogeneous Multiprocessors. In *PARC*, pp. 30–35, September 2005.

[8] S. Gautam, G. Sarkis, E. Tjandranegara, E. Zelkowitz, Y.-H. Lu, and E. J. Delp. Multimedia for Mobile Users: Image Enhanced Navigation. In *IST/SPIE Symposium on Electronic Imaging*, 2006.

[9] F. Gruian. Hard Real-Time Scheduling for Low-Energy Using Stochastic Data and DVS Processors. In *ISLPED*, pp. 46–51, 2001.

[10] Integrated Publishing, Photography. (http://www.tpub.com/content/photography/14209/index.htm).

[11] T. Ishihara and H. Yasuura. Voltage Scheduling Problem for Dynamically Variable Voltage Processors. In *ISLPED*, pp. 197–202, 1998.

[12] J. Jensen. Sur Les Fonctions Convexes Et Les Inegalites Entre Les Valeurs Moyennes. *Acta Math*, 30:175–193, 1906.

[13] P. Juang, Q. Wu, L.-S. Peh, M. Martonosi, and D. W. Clark. Coordinated, Distributed, Formal Energy Management of Chip Multiprocessors. In *ISLPED*, pp. 127–130, 2005.

[14] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, 20(1):46–61, January 1973.

[15] J. Lopez, J. Diaz, M. Garcia, and D. Garcia. Worst-Case Utilization Bound for EDF Scheduling on Real-Time Multiprocessor Systems. In *Euromicro Workshop on Real-Time Systems*, pp. 25–33, 2000.

[16] J. R. Lorch and A. J. Smith. Improving Dynamic Voltage Scaling Algorithms with PACE. In *ACM SIGMETRICS*, pp. 50–61, 2001.

[17] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee. A Case Study of Mobile Robot's Energy Consumption and Conservation Techniques. In *ICAR*, pp. 492–497, 2005.

[18] P. Pillai and K. G. Shin. Real-time Dynamic Voltage Scaling for Low-power Embedded Operating Systems. In *ACM SOSP*, pp. 89–102, 2001.

[19] C. Xian and Y.-H. Lu. Dynamic Voltage Scaling for Multitasking Real-Time Systems with Uncertain Execution Time. In *GLSVLSI*, 2006.

[20] R. Xu, C. Xi, R. Melhem, and D. Moss. Practical PACE for Embedded Systems. In *ACM EmSoft*, pp. 54–63, 2004.

[21] W. Yuan and K. Nahrstedt. Energy-Efficient Soft Real-Time CPU Scheduling for Mobile Multimedia Systems. In *ACM SOSP*, pp. 149–163, 2003.

[22] Y. Zhang, Z. Lu, J. Lach, K. Skadron, and M. R. Stan. Optimal Procrastinating Voltage Scheduling for Hard Real-Time Systems. In *DAC*, pp. 905–908, June 2005.

[23] D. Zhu, R. Melhem, and B. Childers. Scheduling with Dynamic Voltage/Speed Adjustment Using Slack Reclamation in Multi-Processor Real-Time Systems. In *RTSS*, pp. 84–94, Dec 2001.