Utility Maximized Scheduling for Uncertain Task Completions with Limited Energy Budget

Wan Yeon Lee, Hyogon Kim, and Heejo Lee

CIC-CCS-TR 08-002

Korea University

Abstract

The completion times of tasks are not always predictable, and scheduling such tasks under energy constrained environments has become an important issue. In this paper, we solve the utility maximization problem in the execution of multiple tasks under a resource constraint. The tasks have probabilistic execution times and are executable on discrete operation modes having different utilities with different resource requirements. Armed with the theoretical solution to the problem, we design two adaptive scheduling methods that efficiently work for the tasks having widely varying execution times under a limited energy budget: *Optimal Method* and *Approximate Method*. The Optimal Method statistically yields the maximum utility at the cost of heavy run-time overhead. The Approximate Method, on the other hand, provides a near-maximum utility with much less overhead, where the utility decrease is bounded. Extensive experiments on the adaptive MPEG streaming of multimedia tasks show that the proposed methods give higher utility, by up to about 150%, than existing methods that solve for the worst-case execution time.

Index Terms

Energy constraint, Utility maximization, Probabilistic execution time, Optimal algorithm, Approximate algorithm.

I. Introduction

For battery-operated mobile devices, energy management is an issue of paramount importance. Numerous studies have been conducted to reduce operational energy consumption of the devices, thereby

W. Y. Lee is with the Hallym University.

H. Kim and H. Lee are with the Korea University.

extending the battery lifetime. However, there has been a dearth of works that accommodate the possibility of quality degradation for the purpose of completing the given task. In many tasks on mobile devices, such flexibility could be of considerable value. On mobile IPTV devices, for instance, completing a live sports broadcast in lowered quality is usually preferred to sudden termination in the middle of high quality broadcasting due to lack of available energy [1]. Attempting to reduce the energy consumption leaving the level of performance intact as in the aforementioned schemes has a more limited solution space that it can lead to significantly lower user-perceived utility, *i.e.*, unexpected task abortion instead of lower quality. Thus we are motivated to develop an energy management framework that maximizes the system performance subject to the battery lifetime for the (statistically) specified task duration.

In this paper, we assume that a given task can choose one of the *energy-aware operation modes* and the task can change to another operation mode at any time. Each of energy-aware operation modes provides different *utility* with correspondingly disparate energy consumption. Our solution framework focuses on the scheduling decisions to be made over the set of energy-aware operation modes to attain the objective of maximizing the utility value subject to battery lifetime constraint. Probabilistic Chip [2], selected as one of the 10 emerging technologies of 2008 by MIT, is the most prominent device supporting a set of energy-aware operation modes which trade a degree of computation accuracy induced by noise in CMOS devices for substantial energy savings. Other representative examples of energy-aware operation modes are: CPU voltage and frequency scaling [3], [4], energy-aware networking [5], [6], power-aware page allocation [7], adaptive disk spin-down [8], power-aware backlight scaling [9], adaptive speed control of motor [10], [11], and adaptive forward error correction (FEC) [12]. When these energy-aware operation modes are executed with higher energy consumption, they yield better performance such as faster speed of CPU, transmission of larger images, faster RAM access, faster disk access, brighter screen, faster motor revolution, and more robust resiliency to error, respectively. In another work, combinations of heterogeneous operations are handled as a set of modes in the cross-layer framework [13]. In this paper, these performance metrics are quantified with what we call the *utility* value. If tasks are executed in the energy-aware operation mode with a higher utility, the battery is consumed at a faster rate and vice versa. In this setting, we investigate how to *maximize the cumulative utility gained by performing dynamic scheduling over the energy-aware operation modes of multiple tasks* with the requirement to sustain the energy support until their completion under a given energy budget.

For better utilization of available energy, we take into account the statistical execution times of tasks when determining their operation modes, instead of the worst case. The execution times of most multimedia tasks such as 3G cellular calls and IPTV streaming service are usually not predictable.

This uncertainty makes it difficult to determine the energy consumption rates (*i.e.*, the operation modes) without premature exhaustion of battery. The most conservative model to deal with the execution time unpredictability is to schedule for the worst-case execution time, but this approach results in significant waste of resources if the average execution time is much shorter than the worst case. As an alternative, therefore, we utilize the probability distribution of the given task's execution time [3], [4], [14], [15] for better energy management. If an enough number of the execution samples of the given task type are collected, we can obtain the probability distribution of the completion time of the task, so as to statistically estimate the execution time for the task's next run.

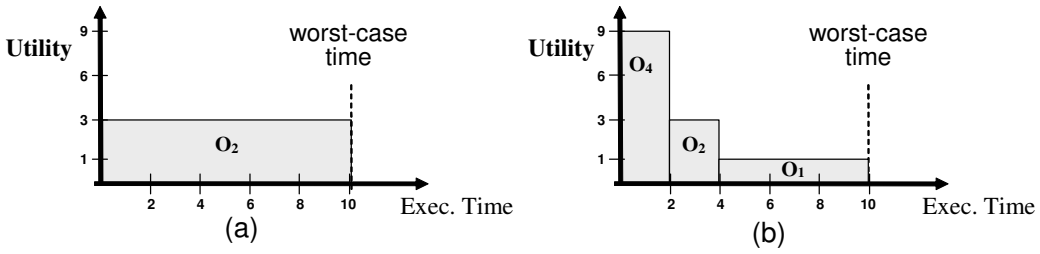

Fig. 1. Motivation example: (a) the conservative approach and (b) the probabilistic approach

In contrast to the existing proposals that select the operation mode based on the assumption that tasks always carry on through their worst-case execution, our method assigns the high-utility high-energy operation modes to those parts of the task execution that are more likely to perform. It can be shown that our method statistically yields a higher utility than the method using any single operation mode throughout the whole task. As an example of the application of the proposed idea, let us take a Major League Baseball game played on a mobile IPTV device. The playing times of baseball games are between 2 and 4 hours in most cases, but rarely exceed 4 hours with the worst-case value of about 8 hours. If the device's battery is running short, the proposed method assigns a high-resolution MPEG-4 streaming mode [16], [17] for 2 hours, and schedules increasingly lower-resolution modes for the execution thereafter. Statistically, this approach can provide higher average resolution than the previous methods assigning the fixed streaming mode, and guarantees that the IPTV device will work for 8 hours without battery exhaustion. Fig. 1 shows an example of assigning available energy of 30 Joules to a task having widely varying completion times, where the task is executable in one of three operation modes whose energy consumption rates are 9, 3 and 1 watts with their respective utility values of 9, 3 and 1 frames/sec. In Fig. 1(a), a fixed operation mode executes the task for the worst-case time of 10 seconds, whereas the operation mode executing the task is changed along with the elapsed execution time in Fig. 1(b). If the task is executed

with the completion time of 2 seconds, the cumulative utility gained during its execution in Fig. 1(a) is 6 frames, which is tripled in Fig. 1(b). In case the task is executed for the whole worst-case time, both methods provide the same cumulative utility of 30 frames with the same energy consumption. If the task is executed six times with completion times 2, 2, 2, 2, 2 and 10 seconds, the mean value of its cumulative utility is 10 frames in Fig. 1(a), which is doubled in Fig. 1(b).

The contribution of this paper is threefold. We first show that the problem of maximizing the cumulative utility of multiple tasks with the constraint of a single resource can be formulated into a tractable scheduling problem of operation modes, if tasks have exact probabilistic execution times and are executable on a set of discrete operation modes providing different utility with their correspondingly disparate resource requirements. Second, armed with a derived optimal solution to the problem, we propose an adaptive scheduling method, called *Optimal Method*, which determines an instant operation mode so as to maximize the cumulative utility of multiple tasks while guaranteeing the completion with a given budget of energy. The average time complexity of the Optimal Method is $O(\log_{\frac{M}{M-1}} E_{max} \cdot \sum_{i=0}^{M}(N^i \cdot \log_2 W^i \cdot \log_2 E_{max} + W^i))$ for M tasks, where N^i and W^i are the number of available operations and the worst-case execution time of i^{th} task and E_{max} is given energy budget. Third, in order to address the complexity of the optimal algorithm, we propose another scheduling algorithm, called *Approximate Method*, which provides a near optimal utility with drastically reduced run-time overhead. The Approximate Method is applied to on-line processing in a semi-static manner, where a set of solutions is computed at compile time and one of them is selected at runtime according to the given energy budget. We show that the utility reduction in the Approximate Method is bounded to a small number. Through experimental evaluation, we show that the proposed methods provide higher utility than existing methods, up to about 150% in the cases.

The rest of this paper is organized as follows: In section 2, we provide background to facilitate the understanding of the proposed methods and review the prior work. In section 3, we formulate the maximization problem of cumulative utility gained from multiple tasks' execution under a resource constraint, and theoretically derive an optimal solution to the problem. In section 4, we describe the detailed implementation of the optimal solution, called the Optimal Method, that adaptively schedules the energy-aware operation modes with relatively heavy run-time overhead. As the main result of this paper, we also describe a practically modified version of the optimal solution, called the Approximate Method, which provides a near maximum utility with low run-time overhead. In section 5, we evaluate the proposed methods and finally conclude this paper in section 6.

II. Background and Previous Work

In this section, we lay the groundwork for our proposal by defining key notions, and discuss the related work.

A. Utility of Energy-aware Operation Mode and Measurement of Energy Consumption and Lifetime

Not surprisingly, empirical studies show that the operation modes consuming higher energy yield better performance. For instance, the speed of Dynamic Voltage Scaling (DVS)-enabled CPU is exponentially proportional to the energy consumption rate [3], [4], and the size of transmitted images on wireless networks is linearly proportional to the energy consumption [18]. Essentially, we can see that there is a tradeoff relation between the energy consumption and the performance. The DVS scheme can change the clock frequency supplied to the CPU, Fine-Granular-Scalability (FGS) video coding [16], [17] and adaptive FEC [19], [12] schemes can change the size of transmitted images per second, power-aware backlight scaling [9] scheme can change the brightness of display screen, power-aware page allocation [7] and adaptive disk spin-down [8] schemes can change the access time of memories, and adaptive motor control [10], [11] scheme can change the moving speed of mobile robots. In this paper, we associate an operation mode with the notion of *utility*, which denotes the aforementioned performance measure. We assume that the utility value corresponding to each energy-aware operation mode is initially given by a user or system administrator. As more realistic model, we consider an arbitrary set of discrete operation modes instead of infinitely continuous modes.

When a given mobile device provides multiple modes, low-energy management mechanisms can make use of the tradeoff between the utility and the energy consumption for the purpose of energy-efficient operation. For instance, mobile robots can change their moving speed by controlling rotation speed of motors, and the risk of collisions with obstacles by adjusting the sensing range of real-time obstacle avoidance [10]. Assigning fast speed and wide sensing to earlier running time and vice versus to later running time can reduce the mean completion time of search & rescue (or retrieval) [20], rather than assigning a fixed speed and sensing range to the whole running time under the constraint of limited energy amount.

Several tools are available to assist and possibly enable the utility-energy tradeoff. PowerScope [21], Advanced Power Management(APM), Advanced Configuration and Power Interface(ACPI), PCCextend 140 CardBus extender [18], and an application-level tool [22] have been developed to allow the operating system (OS) to accurately measure the energy consumption rate for executing each operation mode. The lifetime prediction techniques [23], [24] enable the OS to estimate the lifetime of the battery based on

the energy consumption rate of running tasks and the available energy of a battery. To manage a battery lifetime for a given finite temporal horizon, OS can determine the maximally allocatable energy for the execution of tasks given their energy consumption rates [25], [21].

B. Probabilistic execution time

For simplicity and safety, battery-operated systems could be designed to schedule tasks by their worst-case execution times, *i.e.*, to administer energy usage for the worst case. But it would lead to unnecessarily low utility in case the execution time falls short of the worst case. Indeed, the execution times of tasks can wildly vary due to the diversity of input data or disparate user requests made on them [4], [26], [27]. On the other hand, task scheduling approaches tailored to the average execution times are likely to suffer from untimely energy depletion in case the task runs close to the worst case.

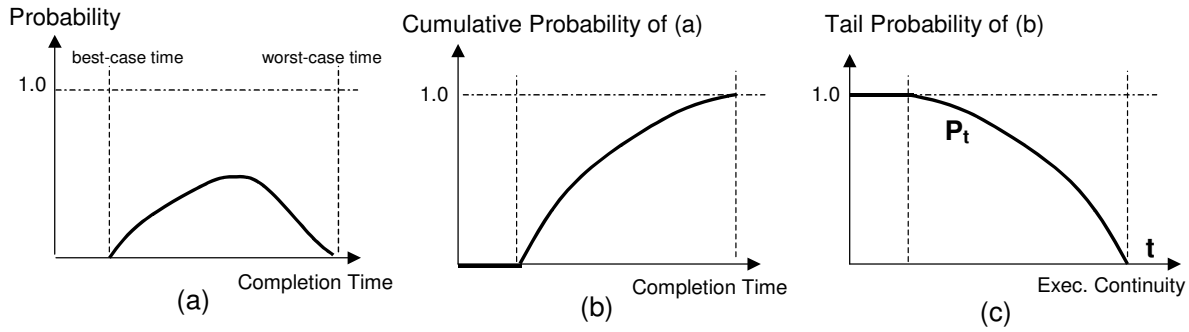

Fig. 2. Distribution of task completion times: (a) the probability distribution of completion times, (b) the cumulative distribution of Fig. (a), and the tail distribution of Fig. (b)

To overcome this problem, we propose in this paper that the probability distribution of execution times should be exploited. The distribution could be derived from statistical models of the variation sources, on-line profiling, or off-line profiling [3]. Fig. 2(a) shows an example model of probability distribution of the execution times of a task, where Fig. 2(b) is its cumulative probability distribution. Fig. 2(c) is the tail distribution of the cumulative probability distribution, denoted as P_t at time t. P_t is the probability that the task continues its execution for at least time t. Notice that if $x < y$, then $P_x \geq P_y$ because the cumulative distribution of all probability functions is always non-decreasing and thus its tail distribution is always non-increasing.

It is certainly true that we cannot know *a priori* the exact completion time of a task with varying execution times. Nevertheless, we still can make good use of the distribution of its previous completion

times, in order to boost the total utility of the task execution. As we can see in Fig. 2(c), the tail distribution of the task completion time is non-increasing, so the earlier processing time of the task always has higher probability to be executed. We will prove later that in such case assigning a larger portion of available energy to the earlier part of the processing leads to the maximum utility. The only remaining question is, then, how to deal out the energy as a function of t and P_t. In Section III, we will formulate this problem more formally, and find the optimal solution.

C. Related Work

Many low-energy management mechanisms [6], [28], [29], [30], [16] have been investigated as ways to reduce energy consumption rate and thereby extend the lifetime of batteries while satisfying the required minimum performance. Recently, some studies [21], [25], [31] have considered the control mechanisms that accommodate performance degradation as a way to sustain the battery operation for a specified critical duration. Graceful degradation [21], [25] or stopping unimportant tasks [31] when energy is low can allow critical tasks to run for the specified duration. Lee, *et al.* [32], [33] addressed the problem of maximizing the utilities, which consists of multi-dimensional values, with multiple tasks executing under resource constraints. They showed that this problem is NP-hard and proposed iterative algorithms to find a suboptimal solution with polynomial time complexity. Recent studies [34], [35] addressed the problem of maximizing the utility of a real-time task while satisfying energy and real-time constraint. Unfortunately, however, all of these studies considered the energy management of tasks only with their fixed execution time, *i.e.*, the worst-case execution time, lacking the consideration for many tasks with varying execution times. More recent studies [36], [37] addressed a similar approach that maximizes the utility of extra computation by dynamically exploiting the slack energy left by execution time variation. This method uses the slack energy of previously scheduled task, earlier completed than the schedule, for the additional computation of the later scheduled tasks, and works with an infinite continuum of operation modes. Our approach contrasts with the work in two aspects: it can work *during* the execution of concurrent tasks, and it assumes *discrete* operation modes, which is more realistic.

The probability distribution of task's execution times has been utilized for the design of less energy consumption in many real-time systems [3], [4], [15], [38], [39], [40]. These studies considered the minimum energy consumption of CPU processing in the DVS framework, which dynamically accelerates and decelerates the clock frequencies supplied to the CMOS-based processors along the execution of tasks. Some of these methods [39], [40] could be applied to the problem tackled in this paper with some modifications: changing the deadline constraint with the energy constraint, and changing the energy

minimization of CPU with the utility maximization of a task. However, their practicability is an issue as they are built on the exhaustive search over a finite number of transition points available at a given point of execution. Since transitions are allowed at any point of a possibly large execution path, the heavy computational cost makes them prohibitive for on-line processing. Furthermore, these methods deal with a *single* task. Finally, speaking of a single task optimization, our previous work [27] maximizes total utility for the execution of a single task on continuous operation modes with a specific relationship between utility U and energy consumption rate R, *i.e.*, $U = (R)^{\alpha}$ for any $0 < R$ and $0 < \alpha$.

III. Utility maximization of multiple tasks under a resource constraint

A. Problem formulation

In this section, we formulate and solve the problem of maximizing the total utility gained during the execution of M tasks, subject to the constraint that the total energy used during the execution is no larger than the given energy budget. As assumed, the tasks have probabilistic execution times and are executable on a finite set of operation modes. We denote the j^{th} operation mode of the i^{th} task, its utility, and its energy consumption rate as O^i_j, U^i_j and R^i_j, respectively. Then an operation mode is a tuple given as $O^i_j = [R^i_j, U^i_j]$. If there are N^i operation modes feasible for the execution of the i^{th} task, we denote them as $\{\ O^i_1 = [R^i_1, U^i_1],\ \cdots,\ O^i_{N^i} = [R^i_{N^i}, U^i_{N^i}]\ \}$. To accommodate "no operation" (NOP) during the execution of this task in the formulation, we define a special (and virtual) operation mode $O^i_0 = [U^i_0, R^i_0]$, where $U^i_0 = 0$ and $R^i_0 = 0$. Finally, the utility is a concave function of the energy consumption rate, *i.e.*, if $R^i_x < R^i_y$, then $U^i_x \leq U^i_y$. The following is the list of notations used in the formulation below.

- $r^i(t)$: energy consumption rate of the instantaneous operation mode of i^{th} task at a time t.
- $u^i(t)$: utility of the instantaneous operation mode of i^{th} task at a time t.
- W^i: the worst-case execution time of i^{th} task.
- P^i_t: the probability that the i^{th} task continues its execution for at least t, as described in Section II-B.
- E_{max}: the energy budget.

The energy-constrained multi-task utility maximization problem can then be formulated as follows:

$$\max\{\ \sum_{i=1}^{M} \int_0^{W^i} u^i(t) \cdot P^i_t \, dt\ \}, \tag{1}$$

$$\text{subject to } \textstyle\sum_{i=1}^{M} \int_0^{W^i} r^i(t)\, dt \leq E_{max}$$

where $u^i(t)$ and $r^i(t)$ are given by the operation mode used at time t, and $r^i(t) \in \{R^i_0, \cdots, R^i_{N^i}\}$ and $u^i(t) \in \{U^i_0, \cdots, U^i_{N^i}\}$ for any $0 \leq t \leq W^i$.

B. Optimal solution with a single task

Before delving into the multi-task utility maximization problem directly, we first derive the optimal solution for the single-task case. For N' discrete operation modes of this task, Equation (1) is rewritten as follows:

$$\max\{ \int_0^W u(t) \cdot P_t \, dt \}, \text{ subject to } \int_0^W r(t) \, dt \leq E, \tag{2}$$

where E is the energy budget allocated for this task. We refer to the schedule producing the maximum utility as the *Optimal Schedule*. The Optimal Schedule has the following properties, which are formally proved in Appendix.

Lemma 1. The Optimal Schedule does not use the operation mode O_y such that $\frac{U_y - U_x}{R_y - R_x} < \frac{U_z - U_y}{R_z - R_y}$ for any $x < y < z$.

Lemma 2. The Optimal Schedule assigns higher-utility operation modes to the earlier processing times and lower-utility operation mode to the posterior processing times.

Lemma 3. The utility generated by the Optimal Schedule is a concavely increasing function of the given energy E.

We select a subset of N operation modes satisfying Lemma 1 among given N' operation modes, and denote them as $\{O_1, \cdots, O_N\}$ ($N \leq N'$). If P_t were identical over all t, choosing O_1 would maximize the utility. But as P_t is a decreasing function of t, the Optimal Schedule selects the operation mode according to Equation 2, restated so that it satisfies Lemma 2 as follows:

$$\max\{ U_N \cdot \int_0^{\pi_N} P_t \, dt + U_{N-1} \cdot \int_{\pi_N}^{\pi_{N-1}} P_t \, dt + \cdots + U_1 \cdot \int_{\pi_2}^{\pi_1} P_t \, dt \}, \tag{3}$$

$$\text{subject to } R_N \cdot \pi_N + R_{N-1} \cdot (\pi_{N-1} - \pi_N) + \cdots + R_1 \cdot (\pi_1 - \pi_2) \leq E,$$

where π_k is the transition point from O_k to O_{k-1} and $0 \leq \pi_k \leq \pi_{k-1} \leq W$ for any $0 < k \leq N$.

If we know the values of all π_ks, we directly get the Optimal Schedule. Unfortunately, however, π_k's depend on the given value of E. To find the values of π_k's, we first verify a relationship among π_ks and next exploit it to search π_k's in a bisection manner on the basis of the value of E. If there is a specific relationship among π_k's, we can obtain all π_k's satisfying the relationship from a fixed value of π_1. The obtained π_k's match with a value of E. Exhaustive search on the value of π_1 enables us to find π_k's satisfying the relationship and matching with the given value of E.

Theorem 1, proved in Appendix, shows a relationship among the values of π_ks. Up to the time point π_2, using O_2 instead of O_1 generates larger utility with the same energy amount than using O_1 instead

of O_0 after the time point π_1, because $P_{t_1} \cdot \frac{U_2-U_1}{R_2-R_1} \geq P_{t_2} \cdot \frac{U_1-U_0}{R_1-R_0}$ for any $t_1 \leq \pi_2$ and $\pi_1 \leq t_2$. Similarly, using O_x instead of O_{x-1} up to the point π_x generates larger utility with the same energy amount than using O_{x-1} instead of O_{x-2} after the time point π_{x-1}. Hence, $P_{\pi_2} \cdot \frac{U_2-U_1}{R_2-R_1} \geq P_{\pi_1} \cdot \frac{U_1-U_0}{R_1-R_0}$, $P_{\pi_3} \cdot \frac{U_3-U_2}{R_3-R_2} \geq P_{\pi_2} \cdot \frac{U_2-U_1}{R_2-R_1}$, $\cdots$, $P_{\pi_N} \cdot \frac{U_N-U_{N-1}}{R_N-R_{N-1}} \geq P_{\pi_{N-1}} \cdot \frac{U_{N-1}-U_{N-2}}{R_{N-1}-R_{N-2}}$.

Theorem 1.

$$\text{If } 0 < \pi_x < \pi_y \leq W \text{ for any } N \geq x > y \geq 1, \text{ then } P_{\pi_x} \cdot \frac{U_x-U_{x-1}}{R_x-R_{x-1}} = P_{\pi_y} \cdot \frac{U_y-U_{y-1}}{R_y-R_{y-1}}.$$

The procedure to find π_ks, whose sum of energy is equal to E while satisfying the condition of Theorem 1, works as follows. Initially, it assigns $\pi_1 = 1$ and searches the time points T_k such that $P_{T_k} = P_{\pi_1} \cdot \frac{U_1}{R_1} \cdot \frac{R_k-R_{k-1}}{U_k-U_{k-1}}$ for $2 \leq k \leq N$. If $P_{T_k} \leq 1$, then it assigns T_k to π_k so that the values of P_{π_k} satisfy the condition of Theorem 1. If $(R_N \cdot \pi_N + R_{N-1} \cdot (\pi_{N-1} - \pi_N) + \cdots + R_1 \cdot (\pi_1 - \pi_2)) < E$, it increases the value of π_1 by one, searches the time points T_k, and assigns each T_k to π_k for each k. This procedure is repeated until the sum $(R_N \cdot \pi_N + R_{N-1} \cdot (\pi_{N-1} - \pi_N) + \cdots + R_1 \cdot (\pi_1 - \pi_2))$ becomes equal to E. If $\pi_1 = W$ but the sum is still less than E, it increases the value of π_2 by one, searches the time points T_k for $3 \leq k \leq N$, and assigns each T_k to π_k. Similarly, it increases the value of π_{j+1} by one if $\pi_j = W$ but the sum is still less than E.

C. Optimal solution with multiple tasks

We next consider the problem of maximizing the total utility of M tasks. In this section, we use superscripts to refer to tasks for convenience. When Ψ^i is the maximum value derived from Equation 3 for given energy E^i, Equation 1 can be reformulated as follows:

$$\max\{\ \Psi^1 + \ldots + \Psi^M\ \},\ \text{subject to } (E^1 + \ldots + E^M) \leq E_{max} \tag{4}$$

According to Lemma 3 proved in Appendix, the value $\frac{\Psi^i}{\partial E^i}$ decreases as E^i increases. To find the solution of Equation 4, we can use the Lagrange Multiplier Method [41].

$$L(E_1, E_2, \cdots, E_N, \lambda) = (\Psi^1 + \ldots + \Psi^M) + \lambda \cdot (E_{max} - (E^1 + \ldots + E^M)),$$
$$\frac{\partial L}{\partial \lambda} = E_{max} - (E^1 + \ldots + E^M) = 0,$$
$$\text{and } \frac{\partial L}{\partial E^k} = \frac{\Psi^k}{\partial E^k} - \lambda = 0 \text{ for } 1 \leq k \leq N.$$

The above equations show that we get the maximum value when the values of $\frac{\Psi^i}{\partial E^i}$ become identical. In other words, the maximum value of Equation 4 is achieved when the derivative of each Ψ^i with regard to E^i becomes the largest. We can numerically find this largest derivative of Ψ^i with regard to E^i as

follows. Initially, zero energy is assigned to each task. It calculates the increment of additional utility of each task when assigning an additional unit of energy, and allocates the additional unit energy only to the task having the largest increment of utility. This procedure is repeated until the available energy is completely allocated.

The derived solution maximizes the total utility of multiple tasks, if tasks have exact probabilistic execution times and are executable on a set of discrete operation modes providing different utility with their correspondingly disparate resource requirements. In the next section, for tasks having uncertain and varying execution time, we will describe how to efficiently implement the derived solution for the adaptive scheduling method of energy-aware operation modes. The derived solution will be reformed to guarantee the support of minimum utility for the worst execution time, operate with low run-time overhead, and accommodate the switching overhead between operation modes.

IV. Approximate Scheduling Method with Low Run-Time Overhead

For the execution of tasks having widely varying execution time, the proposed scheduling method assigns the operation mode with higher utility to the more likely parts of the task execution with a purpose to statistically maximize total utility, based on the solution derived in Section III-C. The proposed method departs from the solution in that it always provides at least the minimum utility U_1 for the worst-case time whereas the solution does not. That is, the proposed method does not use the operation mode O_0. Then, $\pi_1^i = W^i$ for each i hence $E_{max} \geq \sum_{i=1}^{M} R_1^i \cdot W^i$. Henceforth, we assume $E_{max} \geq \sum_{i=1}^{M} R_1^i \cdot W^i$. The proposed method estimates the probabilistic execution time of the tasks on the basis of the distribution of previous completion times[1], and assigns the running times of operation modes before starting their next execution.

In this section, we ultimately propose an approximate scheduling method which provides a near-maximum utility with acceptable run-time overhead. We first explain how to efficiently calculate the assigned running time of energy-aware operation modes for a single task with arbitrarily given energy E. Next we explain how to efficiently distribute available energy E_{max} to M tasks so as to maximize the total utility without consideration of acceptable run-time overhead. Finally, we describe its practical variant which operates with little run-time overhead by allowing a bounded utility decrease.

[1]In this paper, we assume that the information about the probability distribution of previous completion times is transmitted from the application server. Transmission protocol of this information is out of scope.

A. Partitioning of scheduling ranges in a single task

In this subsection, for a single task, we search the running interval $[\pi_{k+1}, \pi_k]$ of each operation mode O_k in the Optimal Schedule along with the value of E. We first determine the value ranges of E in which the same set of operation modes is used in the Optimal Schedule. Next, for each value range $[e_x, e_y]$ of E and each operation O_k in the set, we search the time range $[T_{min}, T_{max}]$ of the switching point from O_k to O_{k-1}, *i.e.*, the switching point is T_{min} when $E = e_x$ and T_{max} when $E = e_y$. Finally, we make narrow the value range of E and the time range of switching point by dividing them using the bisectional method.

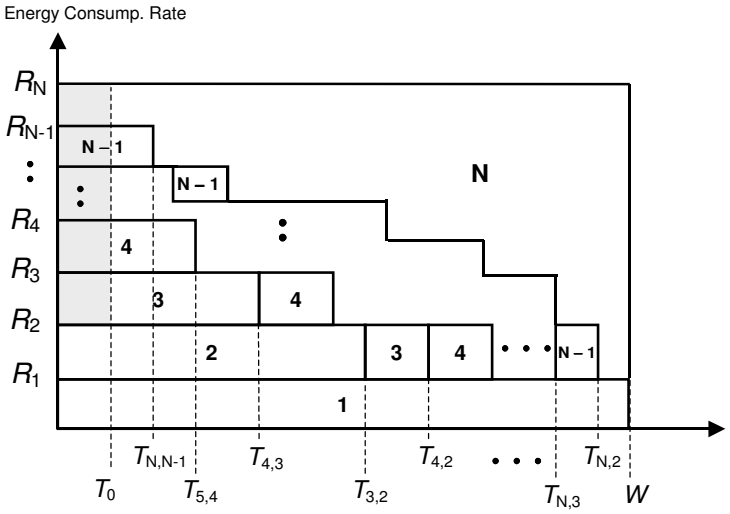

Fig. 3. Scheduling ranges determined by the Coarse-grain Partitioning procedure

Even though the operation modes O_k satisfies the condition of Lemma 1, the Optimal Schedule does not use it if $P_{\pi_k} > 1$ or $\pi_k > W$. Among the operation modes satisfying the condition of Lemma 1, the value of E determines the subset of operation modes used in the Optimal Schedule. Based on the points $T_{x,y}$ and T_0 defined as follows, we calculate the value range of E where the same subset of operation modes is used in the Optimal Schedule.

- $T_{x,y}$: $P_{T_{x,y}} = \frac{U_x - U_{x-1}}{R_x - R_{x-1}} \cdot \frac{R_y - R_{y-1}}{U_y - U_{y-1}}$ for each x and y such that $N \geq x > y > 1$.
- T_0: $P_{T_0} = 1$ and $P_{T_0+1} < 1$.

Then, $P_{T_0} \cdot \frac{U_x - U_{x-1}}{R_x - R_{x-1}} = P_{T_{x,y}} \cdot \frac{U_y - U_{y-1}}{R_y - R_{y-1}}$, which satisfies the condition of Theorem 1. If $R_1 \cdot W < E \leq ((R_2 - R_1) \cdot T_{3,2} + R_1 \cdot W)$, the Optimal Schedule uses only O_1 and O_2. Similarly, the value range of E using only $O_1, \cdots, O_k$ in the Optimal Schedule is

$$(R_{k-1} - R_{k-2}) \cdot T_{k,k-1} + \cdots + (R_2 - R_1) \cdot T_{k,2} + R_1 \cdot W < E \leq$$
$$(R_k - R_{k-1}) \cdot T_{k+1,k} + \cdots + (R_2 - R_1) \cdot T_{k+1,2} + R_1 \cdot W.$$

The energy range using $O_1, \cdots, O_k$ in the Optimal Schedule is smaller than that using $O_1, \cdots, O_{k-1}$ because $T_{x,k} > T_{y,k}$ if $x > y > k$, such as shown in Fig. 3. We sort these N ranges in the increasing order of their values and call the k^{th} range *Range* k. In the Range k, the Optimal Schedule uses only $O_1, \cdots, O_k$. For further explanation, we use the following notations for each Range k:

- e_k: the maximum energy of Range k. That is, $e_k = \sum_{j=1}^{k} A_j$ where A_j is the area of rectangles having index j in Fig. 3.
- ψ_k: the maximum utility provided in Range k. That is, $\psi_k = U_k \cdot \int_0^{T_{k+1,k}} P_t \, dt + U_{k-1} \cdot \int_{T_{k+1,k}}^{T_{k+1,k-1}} P_t \, dt + \ldots + U_1 \cdot \int_{T_{k+1,2}}^{W} P_t \, dt$.
- ϕ_k: the efficiency of energy utilization in Range k. That is, $\phi_k = \frac{\psi_k - \psi_{k-1}}{e_k - e_{k-1}}$.

We refer to the procedure to calculate e_k, ψ_k, ϕ_k, and $T_{x,y}$ for $1 \leq k \leq N$ and $N \geq x > y \geq 1$ and store them as *Coarse-grain Partitioning*. Table I shows an example of four ranges calculated with four operation modes by the Coarse-grain Partitioning procedure. In this example, the Optimal Schedule uses O_4 from 0 to W if $E > e_4$. The computational complexity of this procedure is $O(N^2 \cdot \log_2 W + W)$ because the operation to find all $T_{x,y}$'s requires $O(N^2 \cdot log_2 W)$ steps and the operation to calculate the values of ψ requires $O(W)$ steps. Its memory complexity is $O(N^2)$ to store their transition points $T_{x,y}$'s for $N \geq x > y \geq 1$ and the values of e_k, ψ_k and ϕ_k for $1 \leq k \leq N$.

TABLE I

EXAMPLE OF COARSE-GRAIN PARTITIONING

Range	Energy	Selected Modes: Range	Transition Points	Range of Max Utility
1	$E \leq e_1$	$O_1 : 0 \sim \pi_1$	$\pi_1 = \frac{E}{R_1}$	$\Psi \leq \psi_1$
2	$e_1 < E \leq e_2$	$O_2 : 0 \sim \pi_2$, $O_1 : \pi_2 \sim W$	$\pi_2 = \frac{E - R_1 \cdot W}{R_2 - R_1}$	$\psi_1 < \hat{\Psi} \leq \psi_2$
3	$e_2 < E \leq e_3$	$O_3 : 0 \sim \pi_3$, $O_2 : \pi_3 \sim \pi_2$, $O_1 : \pi_3 \sim W$	$0 < \pi_3 \leq T_{4,3}$, $T_{3,2} < \pi_2 \leq T_{4,2}$, $\pi_2 = \pi_3 \cdot \frac{U_3 - U_2}{R_3 - R_2} \cdot \frac{R_2 - R_1}{U_2 - U_1}$	$\psi_2 < \Psi \leq \psi_3$
4	$e_3 < E \leq e_4$	$O_4 : 0 \sim \pi_4$, $O_3 : \pi_4 \sim \pi_3$, $O_2 : \pi_3 \sim \pi_2$, $O_1 : \pi_2 \sim W$	$0 < \pi_4 \leq W$, $T_{4,3} < \pi_3 \leq W$, $T_{4,2} < \pi_2 \leq W$, $\pi_2 = \pi_3 \cdot \frac{U_3 - U_2}{R_3 - R_2} \cdot \frac{R_2 - R_1}{U_2 - U_1}$ $= \pi_4 \cdot \frac{U_4 - U_3}{R_4 - R_3} \cdot \frac{R_2 - R_1}{U_2 - U_1}$	$\psi_3 < \Psi \leq \psi_4$

If $e_k \leq E \leq (e_k + (R_{k+1} - R_k) \cdot T_0)$, we can directly find the Optimal Schedule because $P_t = 1$ for $0 \leq t \leq T_0$. In this case, the operation mode O_{k+1} is used from time 0 to time $\frac{E - e_k}{R_{k+1} - R_k}$ and the

operation mode O_k is used from the time $\frac{E-e_k}{R_{k+1}-R_k}$ to the time $T_{k,k-1}$. The other operation modes are scheduled as in Range k. We define e_{k*} to be

$$e_{k*} = e_k + (R_{k+1} - R_k) \cdot T_0.$$

If $e_{k*} < E < e_{k+1}$, we cannot get the Optimal Schedule directly from the Coarse-grain Partitioning. In this case, we create more (sub)ranges within Range $(k+1)$, such as shown in Fig. 4.

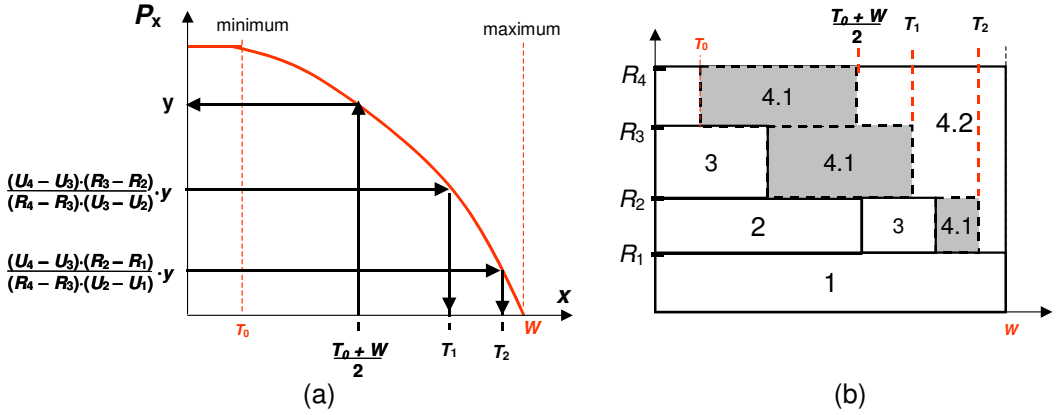

Fig. 4. Example of the Fine-grain Partitioning procedure: (a) searching the transition points and (b) dividing a range of energy into two subranges

For further partitioning within Range k, we use the value of $P_{(\frac{T_0+T_{k+1,k}}{2})}$ where $T_{N+1,N} = W$, denoted as y in Fig. 4(a). From this value, we search $(k-1)$ transition points T_x such that $P_{T_x} = \frac{U_k-U_{k-1}}{R_k-R_{k-1}} \cdot \frac{R_{k-x}-R_{k-x-1}}{U_{k-x}-U_{k-x-1}} \cdot y$, such as shown in Fig. 4(a). Then, Range k is divided into two subranges, denoted as Range $k.1$ and Range $k.2$. The maximum energy of Range $k.1$ is

$$e_{k.1} = (R_k - R_{k-1}) \cdot \frac{T_0+T_{k+1,k}}{2} + (R_{k-1} - R_{k-2}) \cdot T_1 + \ldots + (R_3 - R_2) \cdot T_{k-1} + R_1 \cdot W.$$

We call this procedure *Fine-grain Partitioning*. The computational complexity of the Fine-grain Partitioning procedure is $O(N \cdot \log_2 W)$. To find the Optimal Solution, this Fine-grain Partitioning procedure is repeated until one of subranges has the maximum energy almost equal to E. Its average complexity is $O(N \cdot \log_2 W \cdot \log_2 E_{max})$ because this procedure divides each range into two subranges having roughly equal amount of energy. Calculating the maximum utility of each subrange requires $O(W)$ steps.

B. Optimal scheduling of multiple tasks

In this subsection, we propose an adaptive scheduling method, called *Optimal Method*, which distributes available energy E_{max} to M tasks and finds the Optimal Schedule of each task with its distributed energy. This method works in runtime and proceeds as follows. Each step, it calculates for each task the expected utility increment for even allocation of available energy. Then it carries on the allocation of the divided energy only to the task expected to have the largest increment. This procedure is repeated until the available energy E_{max} is completely allocated. Finally, according to the amount of energy allocated to each task, this method assigns the running times of operation modes for the tasks by exploiting the Coarse-grain Partitioning and the Fine-grain Partitioning procedures.

The following pseudo-algorithm describes the Optimal Method, where E^i and ΔE denote the amount of energy allocated to i^{th} task and that remaining after allocating E^i, *i.e.*, $\Delta E = E_{max} - \sum_{i=1}^{M} E^i$, respectively.

Optimal Method

Step 1. Assign the energy $R_1^i \cdot W^i$ to E^i and calculate $\Delta E \Leftarrow E_{max} - \sum_{i=1}^{M} R_1^i \cdot W^i$.

Step 2. Find the two maximum utility values when assigning E^i and $(E^i + \frac{\Delta E}{M})$ to each task by exploiting the Coarse-grain Partitioning and the Fine-grain Partitioning procedures. Calculate the utility increments when assigning the additional energy $\frac{\Delta E}{M}$ to each task.

Step 3. Assign the additional energy $\frac{\Delta E}{M}$ to the task having the largest utility increment in Step 2. Update the value of ΔE, *i.e.*, $\Delta E \Leftarrow (\Delta E - \frac{\Delta E}{M})$.

Step 4. Repeat Steps 2 and 3 until $\Delta E \simeq 0$.

Step 5. Determine the running times of operation modes according to the value of E^i for each task by exploiting the Coarse-grain Partitioning and the Fine-grain Partitioning procedures.

The computational complexity of the Optimal Method is $O(\log_{\frac{M}{M-1}} E_{max} \cdot \sum_{i=0}^{M} (N^i \cdot \log_2 W^i \cdot \log_2 E_{max} + W^i))$. In Step 2, finding the maximum utility of each task requires $O(N^i \cdot \log_2 W^i \cdot \log_2 E_{max} + W^i)$ operations. The number of repetitions in Step 4 is $\log_{\frac{M}{M-1}} E_{max}$. In Step 5, determining the running time of operation modes for each task requires $O(N^i \cdot \log_2 W^i \cdot \log_2 E_{max})$ operations. When the values of W^i and E_{max} are large, the computational overhead is too heavy to carry out this method at runtime. Therefore in the next section, we modify this method to find a near-optimal solution with acceptable compile-time overhead and little run-time overhead.

C. Approximate scheduling of multiple tasks

In order to reduce the runtime overhead, we propose another adaptive scheduling method, called *Approximate Method*, which simply selects one of pre-computed decisions for each task. To make a set of pre-computed decisions, this method performs the Coarse-grain Partitioning and the Fine-grain Partitioning at compile-time until $\frac{e^i_{k+1}-e^i_k}{e^i_k} \leq \epsilon$ for each k and i. We refer to the value of ϵ as *Error Bound*, which is given by the user or the system administrator and denotes the difference bound from the solution of the Optimal Method. These subranges computed at compile-time are stored to guide the runtime decisions.

The computational complexity at compile-time is $O(\max\{N \cdot log_2 W,\ \ W\} \cdot \log_{(1+\epsilon)} \frac{R^i_{N^i}}{R^i_1})$, where $\log_{(1+\epsilon)} \frac{R^i_{N^i}}{R^i_1}$ is the number of ranges generated at compile-time for i^{th} task because $(1+\epsilon)^x \cdot R^i_1 \cdot W^i \simeq R^i_{N^i} \cdot W^i$. The operation to find all $T_{x,y}$'s of each subrange k for $N \geq x > y \geq 1$ requires $O(N \cdot log_2 W)$ steps as explained in section IV-A. The operations to calculate e_k, ψ_k and ϕ_k of each subrange k require $O(N)$, $O(W)$ and $O(1)$ steps, respectively. The memory complexity to store these pre-computed solutions of all subranges is $O(\sum_{i=1}^{M} N^i \cdot \log_{(1+\epsilon)} \frac{R^i_{N^i}}{R^i_1})$. Fig. 5 shows the data structure to store these pre-computed solutions of a task.

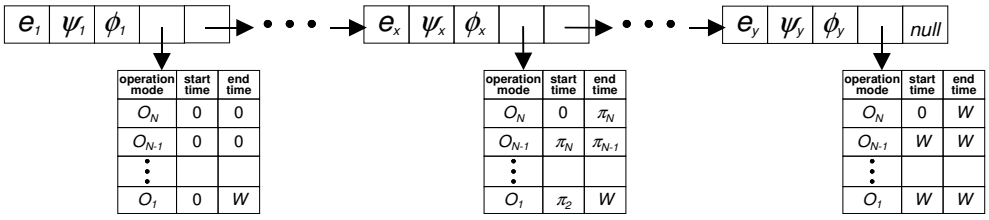

Fig. 5. Data structure to store the pre-computed solutions for each task

To distribute the available energy E_{max} to M tasks during on-line processing, this method proceeds as follows. It first assigns the energy e^i_1 to E^i for the Range 1 of each task. Next, it selects the largest value of ϕ^m_2 among all values of ϕ_2, and allocates the addition energy $(e^m_2 - e^m_1)$ to E^m, *i.e.*, $E^m = e^m_2$. Among the values of remaining ϕ^i_k not selected, it selects the largest value ϕ^m_k and additionally allocates $(e^m_k - e^m_{k-1})$ to E^m. This procedure is repeated unless $\Delta E < (e^m_k - e^m_{k-1})$. The following pseudo-code describes the Approximation Method, where α_i denotes the subrange finally selected for the execution of i^{th} task. The computational complexity of the Approximation Method is $O(\sum_{i=1}^{M} \log_{(1+\epsilon)} \frac{R^i_{N^i}}{R^i_1})$, where $\log_{(1+\epsilon)} \frac{R^i_{N^i}}{R^i_1}$ is the number of subranges generated at compile-time for each task.

Approximation Method

Step 1. Allocate $e_1^i (= R_1^i \cdot W^i)$ to E_1^i for each i. That is, $\alpha_i \Leftarrow 1$ and $\Delta E \Leftarrow E_{max} - \sum_{i=1}^{M} E_1^i$.

Step 2. Search the largest value $\phi_{\alpha_m+1}^m$ among all $\phi_{\alpha_i+1}^i$s, and allocate the energy $(e_{\alpha_m+1}^m - e_{\alpha_m}^m)$ to E^m. That is, $E^m \Leftarrow E^m + (e_{\alpha_m+1}^m - e_{\alpha_m}^m)$.

Step 3. Update the values of ΔE and α_m. That is, $\Delta E \Leftarrow \Delta E - (e_{\alpha_m+1}^m - e_{\alpha_m}^m)$ and $\alpha_m \Leftarrow \alpha_m + 1$.

Step 4. Repeat Steps 2 and 3, if $\Delta E > 0$.

Step 5. Assign the running times of operation modes according to the schedule of the finally selected subrange for each task.

Now, let us consider the difference between results of the Optimal Method and the Approximation Method. We define *error ratio* as $\frac{\tilde{\Psi}-\hat{\Psi}}{\tilde{\Psi}}$, where $\tilde{\Psi}$ and $\hat{\Psi}$ are the results of the Optimal Method and the Approximation Method, respectively. Then the Theorem 2 in the Appendix shows that the mean value of the error ratio is less than the Error Bound ϵ for any $M \geq 1$. The value of ϵ can be given by the user or the system administrator at compile-time and controls the tradeoff between the solution's quality and overhead of the Approximation Method. If the value of ϵ is decreased, the Approximation Method provides higher utility at the cost of larger computational overhead at compile-time and larger memory overhead to store more solutions at run-time.

D. Switching overhead between operation modes

To make the analysis more precise, in this section we consider an energy overhead to dynamically change the operation mode of a task. Let $\Lambda_{k+1,k}$ denote the energy overhead to change the operation mode from O_{k+1} to O_k. When $e_k < E \leq e_{k+1}$, $\Lambda_{k+1,k}$ is required for the transition from O_{k+1} to O_k because of additional use of O_{k+1}. In that case, while $e_k < E \leq e_k + \Lambda_{k+1,k}$, using the energy $\Lambda_{k+1,k}$ for $O_2, \ldots, O_k$ provides more utility than using the energy $\Lambda_{k+1,k}$ for the execution of O_{k+1}. We replace the value of e_k with $(e_k + \Lambda_{k+1,k})$ and the values of $\pi_k, \ldots, \pi_2$ so that

$$e_k + \Lambda_{k+1,k} = (R_k - R_{k-1}) \cdot \pi_k + \ldots + (R_2 - R_1) \cdot \pi_2 + R_1 \cdot W$$
$$\text{and } P_{\pi_j} = P_{\pi_k} \cdot \tfrac{U_k - U_{k-1}}{R_k - R_{k-1}} \cdot \tfrac{R_j - R_{j-1}}{U_j - U_{j-1}} \text{ for each } 1 < j < k.$$

To find the values of $\pi_k, \ldots, \pi_2$, we use the Fine-grain Partitioning procedure for a large $\Lambda_{k+1,k}$ and increase the original values of $\pi_k, \ldots, \pi_2$ one by one for a small $\Lambda_{k+1,k}$. The computational complexity accounting for the transitions with M tasks is $O(\sum_{i=1}^{M} (N^i)^2 \cdot \log_2 W^i)$, because the number of $\Lambda_{k+1,k}$ is $(N^i - 1)$ and the operation required for each transition is $O(N^i \cdot \log_2 W^i)$.

E. Extension to other problems

The proposed method can be applicable to other similar problems, which need to maximize the total utility of tasks having probabilistic execution subject to a single resource constraint. For example, this solution maximizes the entire communication quality of voice calls with a limited amount of prepayment deposit, where charging per unit time depends on the communication quality and voice calls have widely varied communication times. As well, this solution can maximize the energy saving of CPU executing real-time tasks with a deadline constraint of the tasks, where the processing speed of DVS-enabled CPU depends on its energy consumption and the tasks have widely varied computation amount.

V. Performance evaluation

In this section, we compare the proposed methods with existing methods [21], [25]. For evaluation metric, we define *Utility Increment* to be

$$\frac{TU_p - TU_e}{TU_e} \times 100,$$

where TU_p and TU_e are the total utility provided by the proposed method and by the existing method, respectively. This is the ratio of the additional utility created by the proposed method to that produced in the existing method. We also define *Feasible Energy* to be $\frac{E_{max}-E_l}{E_h-E_l}$, where $E_l (= \sum_{m=1}^{M} R_1^m \cdot W^m)$ and $E_h (= \sum_{m=1}^{M} R_{N^m}^m \cdot W^m)$ are the minimum energy required to complete the task in the lowest operation mode and in the highest operation mode for the worst time, respectively. The Feasible Energy is the ratio of the remaining energy after the worst time execution in O_1 to the energy difference when assigning O_N and O_1 for the worst time execution. Because the worst-case execution time is usually selected with a sufficiently large value, the range $[E_l,\ E_h]$ covers the majority of battery's energy if the number of tasks or the energy difference between O_1 and O_N is large. Here, we do not take into account the energy overhead required to change operation modes because the previous work [21] showed that the overhead is relatively negligible.

A. Models of operation modes and varying execution times

To model the multi-task execution environment, we draw the data on the operation modes from the MPEG-4 FGS streaming [16] and the Pioneer 3DX robot systems [11]. The video quality of the MPEG streaming in the optimized communication and the straight moving speed of the robot without load are used for their corresponding utility. Fig. 6 shows the relationship between their utility values and their

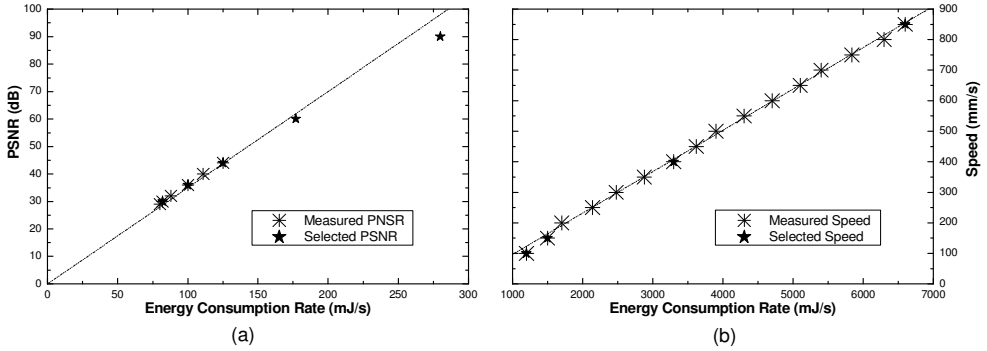

Fig. 6. Utility and energy consumption rate of measured data and selected data in (a) the MPEG streaming and (b) the Pioneer robot

TABLE II

MODELS OF OPERATION MODES

(a) Video qualities and energy consumption rates of MPEG-4 FGS streaming

Operation Mode	O_1	O_2	O_3	O_4	O_5
Utility = PSNR (dB)	30	36	44	60	90
Energy (mJ/s)	82	100	125	177	280

(b) Motor speeds and energy consumption rates of Pioneer 3DX robot

Operation Mode	O_1	O_2	O_3	O_4
Utility = Speed (mm/s)	100	150	400	850
Energy (mJ/s)	1200	1500	3300	6600

energy consumption rates. Their utility values are almost linearly proportional to the energy consumption rates of the 802.11b WLAN card and the robot's motors as follows, respectively.

- Energy rate of WLAN card (mJ/s) = 0.35 $\times$ PSNR of MPEG streaming (dB)
- Energy rate of robot's motors (mJ/s) = 7.4 $\times$ Moving Speed (mm/s) + 290

It is clear that the upper bound of Utility Increment is $\frac{U_N - U_1}{U_1}$. Evaluation results are limited by the upper bound of Utility Increment. In order to avoid low upper bound of Utility Increment, we select three data from the measured 6 data of the MPEG streaming and generate two data based on the above linear increment in Fig. 6(a), while we just select four data from the measured 16 data of the robot in Fig. 6(b). Table II(a) shows the five operation modes of the MPEG-4 FGS streaming, and Table II(b) shows the

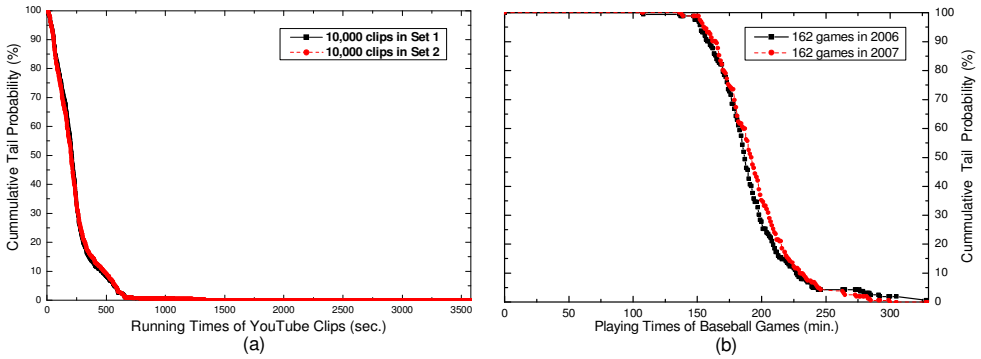

Fig. 7. Probability distribution of (a) the running times of YouTube clips and (b) the playing times of MLB baseball games

four operation modes of the robot, whose utility is a concave function of their energy consumption rate[2].

In order to evaluate the impact of different distributions of execution times, we use the data from both real-life multimedia tasks and synthetically generated tasks. The execution times of multimedia tasks are drawn from the running time distribution of 2 sets of YouTube video clips (Fig. 7(a), 10K clips per set) [42] and the playing time distribution of 2006 and 2007 New York Yankees baseball games (Fig. 7(a)) [43]. The longest running time of clips in the set 1, 3580 seconds, is used for the worst-case time. In the case of baseball games, the longest playing time in 2006 is 330 minutes.

The execution times of synthetic tasks are generated between 1 and 1,000 with normal, exponential, and uniform distributions. The worst-case execution time of all synthetic tasks is given as 1,000. Those exceeding the worst time are replaced with the worst time. The mean value is initially given as 50% of the worst time in the normal distribution and exponential distribution, and the standard deviation of the normal distribution is given as 10% of the mean value. In the performance evaluation, the average values of results obtained from 100,000 synthetic tasks are used.

B. Results of a single multimedia task

Fig. 8 shows the total utility gained from the execution of the multimedia tasks with the five operation modes in Table II(a). In the Approximate Method, the Error Bound ϵ is given as 0.05 (5%). Fig. 8(a)

[2]Even if there are operation modes whose utility values do not follow the concave function of their energy consumption rates, it is shown that they are never used in the Optimal Schedule by Lemma 2 in Section III-B.

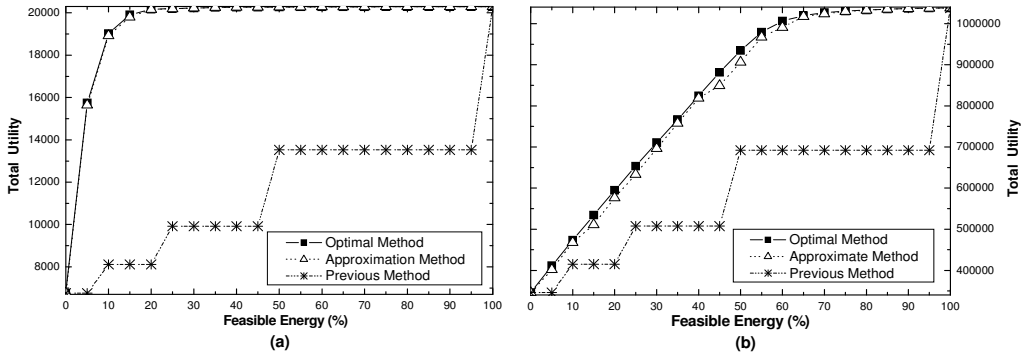

Fig. 8. Total utility against the values of Feasible Energy in (a) YouTube clips and (b) baseball games

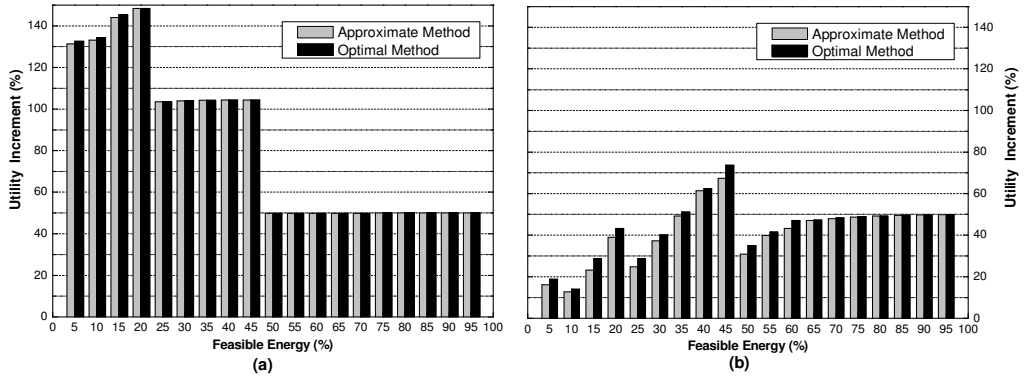

Fig. 9. Utility Increments against the values of Feasible Energy in (a) YouTube clips and (b) baseball games

and (b) show the average values of the total utility in 10,000 runs of the set 2's clips and in 162 runs of 2007 baseball games, respectively. The total utility in Fig. 8 is the integration of resolution gained from the whole MPEG-4 streaming. The total utility of the Optimal Method and the Approximate Method in Fig. 8(a) increases more rapidly, compared with those in Fig. 8(b). This is mainly due to the relatively smaller mean value of Fig. 8(a). The increase patterns of the existing method in Fig. 8(a) and (b), however, are almost identical.

Fig. 9(a) and (b) show the average values of Utility Increment. When the Feasible Energy is 20%, the Approximate Method and the Optimal Method show the best value, 148.4% and 148.5% in Fig. 9(a) and 67.4% and 73.7% when the Feasible Energy is 45% in Fig. 9(b), respectively. The mean Utility Increments of the Approximate Method and the Optimal Method in Fig. 9(a) are 83.0% and 83.2%, and those in

TABLE III

ERROR RATIO (%) OF THE APPROXIMATE METHOD

(a) YouTube clips

Feasible Energy	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%	55%	60%	65%	70%
$\epsilon = 3\%$	0.6	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$\epsilon = 5\%$	0.6	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$\epsilon = 10\%$	7	0.5	1.7	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$\epsilon = 20\%$	14.7	9	1.7	0.0	0.2	0.2	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$\epsilon = 30\%$	14.7	19.7	1.7	3.0	3.1	0.2	0.3	0.0	0.0	0.0	0.1	0.0	0.0	0.0

(b) Baseball games

Feasible Energy	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%	55%	60%	65%	70%
$\epsilon = 3\%$	2.3	1.2	1.3	0.3	0.6	2.0	1.2	0.7	1.8	1.3	0.1	0.7	0.2	0.1
$\epsilon = 5\%$	2.3	1.2	4.3	3.1	3.0	2.0	1.2	0.7	3.7	3.0	1.2	1.5	0.2	0.3
$\epsilon = 10\%$	2.3	1.2	4.3	8.5	3.0	2.0	1.2	0.7	7.1	5.7	4.5	1.5	0.2	0.9
$\epsilon = 20\%$	2.3	1.2	10.6	8.5	3.0	10.9	1.2	8.1	14.1	5.7	10.0	1.5	2.8	3.5
$\epsilon = 30\%$	2.3	1.2	10.6	19.7	3.0	10.9	1.2	8.1	14.1	5.7	10.0	1.5	2.8	3.5

TABLE IV

NUMBER OF PRE-CALCULATED SOLUTIONS FOR THE APPROXIMATE METHOD

Error Bound	3%	5%	10%	20%	30%
YouTube clips	95	69	45	28	23
Baseball games	66	41	22	14	12

Fig. 9(b) are 41.4% and 43.5%, respectively. This experiment shows that the proposed method has better performance for a smaller mean value compared with the worst-case execution time. If the worst-case time is determined too tightly, the risk to irresistibly stop the execution before completion due to lacking energy is increased. The large worst-case value expedites the profitable capability to accommodate the completion of long execution exceeding the expected maximum execution time.

We examine the error ratio of the Approximate Method against various values of Error Bound. Table III shows the error ratios, which are rounded off to two decimal places, when $\epsilon = 3$, $\epsilon = 5$, $\epsilon = 10$, $\epsilon = 20$ and $\epsilon = 30$. Because the error ratios when Feasible Energy $> 70\%$ are smaller than 1%, we do not display them. The average value of error ratios is smaller than the given Error Bound in all cases. The error ratios in Table III(a) are very close to 0 except when Feasible Energy $= 5\%$ or 10%, whereas those

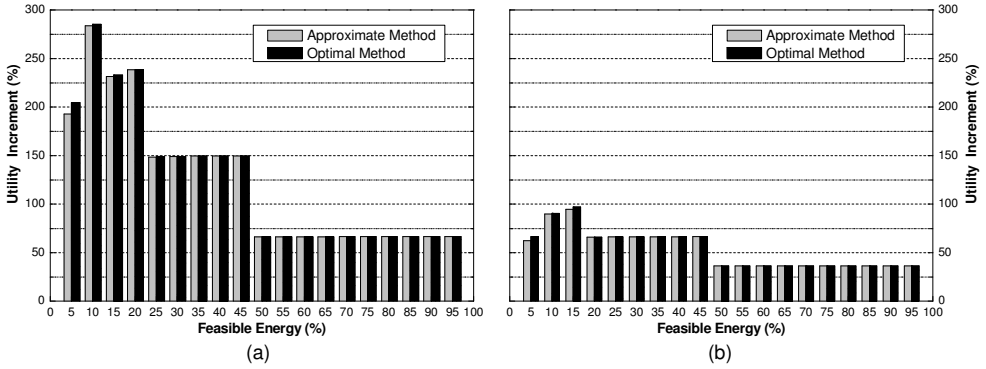

Fig. 10. Utility Increments against the values of Feasible Energy (a) when using six operation modes with additional $O_6 = [R_6 = 502,\ U_6 = 150]$ and (b) when using four operation modes without O_5

in Table III(b) are relatively large in several places of Feasible Energy. This is because the total utility in Fig. 8(a) becomes close to the upper bound of the total utility when Feasible Energy $= 15\%$ due to its relatively smaller mean execution time. In contrast, the total utility in Fig. 8(b) becomes close to the upper bound when Feasible Energy $= 65\%$ due to its relatively larger mean execution time.

Table IV shows the number of solutions calculated at compile-time and stored for the run-time selection of the Approximate Method, when $\epsilon = 3\%$, $\epsilon = 5\%$, $\epsilon = 10\%$, $\epsilon = 20\%$ and $\epsilon = 30\%$. As the value of Error Bound decreases, the number of pre-calculated solutions increases. The number of solutions for YouTube clips is relatively larger than that for baseball games. The number of subranges generated within Range 5 for YouTube clips is much larger than that for baseball games, while the numbers of subranges generated within Range 1, Range 2, Range 3 and Range 4 for YouTube clips are equal to or slightly smaller than those for baseball games. This is because the difference between the minimum and maximum energy of Range 5 for YouTube clips is larger than that for baseball games. The minimum energy of Range 5 for YouTube clips is equal to about Feasible Energy 7% , whereas that for baseball games is equal to about Feasible Energy 24%. The maximum energy of Range 5 is equal to Feasible Energy 100% in both cases.

To explore the effect of different utility definitions, we now change the utility assigned to each operation mode. Fig. 10(a) and (b) show the results of the YouTube clips when using six operation modes with an additional operation mode $O_6 = [R_6 = 502,\ U_6 = 150]$ and when when using four operation modes without the operation mode O_5, respectively. Through the comparison of these results with those of

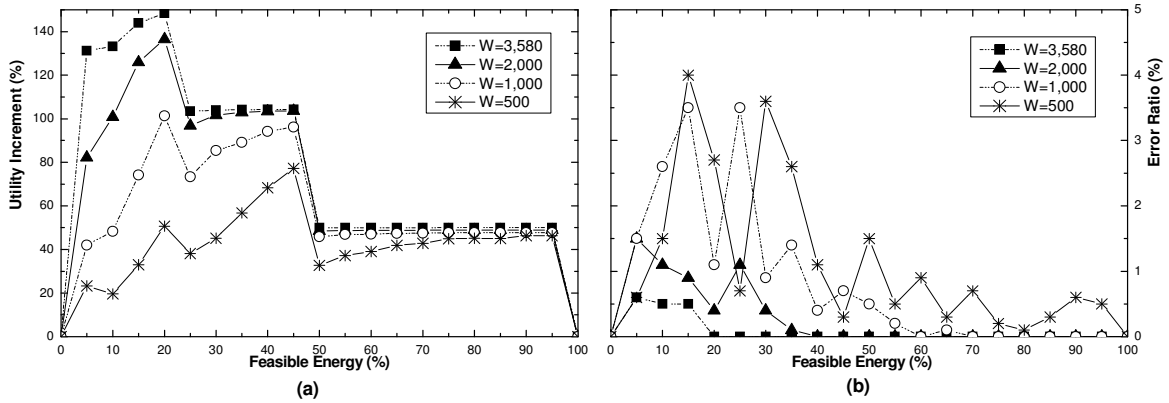

Fig. 11. (a) Utility Increments and (b) error ratios of the Approximate Method against the worst-case value

Fig. 9(a), it is confirmed that the proposed methods achieve better Utility Increment when the relative difference of utility values between O_1 and O_N, $\frac{U_N}{U_1}$, becomes larger.

To examine the impact of various distributions having different mean values, we change the worst execution time 3580 of the YouTube clips with 500, 1000 or 2000. The execution times exceeding the given worst value are truncated at the worst value. As the worst value decreases, the mean execution time of the YouTube clips becomes close to the worst value. Fig. 11(a) shows the average Utility Increments of the Approximate Method in the second set of 10,000 clips when the worst execution time is 500, 1000, 2000 or 3580. The Approximate Method achieves better utility enhancement when the mean value of execution times is smaller than their worst value. When the Feasible Energy is larger, the performance of the Approximate Method is less sensitive to the mean value of execution times. Fig. 11(b) shows the error ratios of the Approximate Method with the Error Bound 10%. The mean error ratios when $W = 500$, $W = 1,000$, $W = 2,000$ and $W = 3,580$ are 1.08%, 0.78%, 0.26% and 0.08%, respectively. The Approximate Method has larger error ratio when the mean value of execution times is closer to their worst value.

C. Results for synthetic tasks

We examine the performance impact when the probability distribution of execution times performed with the proposed method does not match exactly with the expectation of the probability distribution P_t. In this experiment, we use the synthetic tasks generated with normal and exponential distributions. Fig. 12 shows Utility Increments of the Approximate Method with the operation modes in Table II(a), where

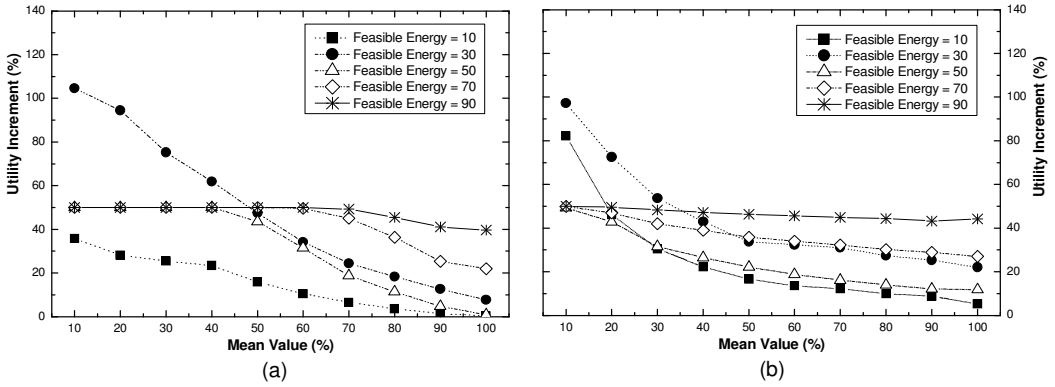

Fig. 12. Utility Increments with *inaccurate* expectation of the probability P_t in (a) normal distribution and (b) exponential distribution

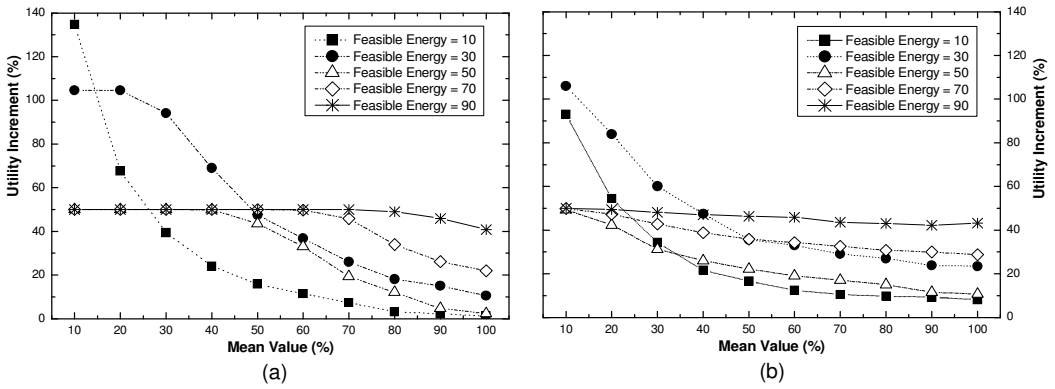

Fig. 13. Utility Increments with *accurate* expectation of the probability P_t in (a) normal distribution and (b) exponential distribution

'Mean Value' denotes the ratio of the mean execution time to the worst-case execution time. Fig. 12(a) and (b) show the results when tasks are generated with normal distribution and exponential distribution, respectively. The mean value of execution times used for the calculation of the probability distribution P_t is fixed to 50% of the worst time, whereas the mean value of execution times performed with the proposed method is changed. In the figure, the performance of the Approximate Method becomes worse as the mean value of actual execution times is closer to the worst execution time. As the value of Feasible

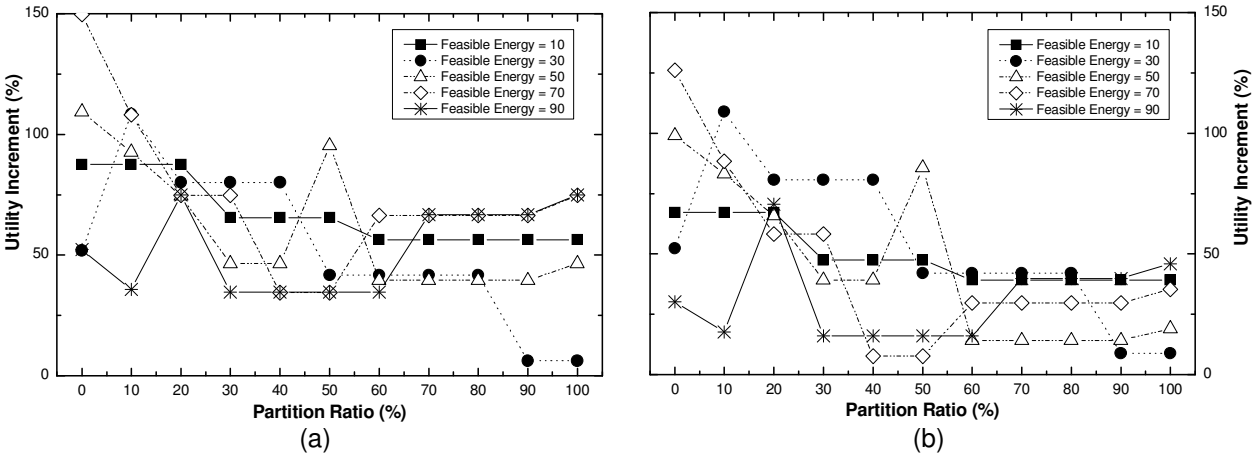

Fig. 14. Utility Increments of two tasks against the Partition Ratio of Feasible Energy

Energy becomes larger, the performance difference is less sensitive to the mean value of actual execution times.

Fig. 13(a) and (b) show the results when the probability distribution of execution times performed with the proposed method matches well with the expectation of the probability distribution P_t. Comparing the results of Fig. 12 with those of Fig. 13, it is shown that the performance of the Approximate Method is less sensitive to the expectation accuracy of the probability distribution P_t, if the Feasible Energy becomes larger. In the experiments of Fig. 12 and Fig. 13, the error ratio of the Approximate Method performed with the Error Bound 5% is always smaller than 5%.

We examine the performance of the Approximate Method when two tasks are running concurrently. The first task is executed with the five operation modes of Table II(a) and generated with normal and exponential distributions. The second task is executed with the four operation modes of Table II(b) and generated with uniform distribution. Their mean values are the half of their worst values. The execution times used for the calculation of P_t are also generated with the same distribution for each task. Fig. 14(a) and (b) show the results when the first task is generated with normal distribution and exponential distribution respectively, where 'Partition Ratio' denotes the ratio of the energy amount assigned for the first task to that of E_{max}. Note that the previous method has different values of total utility according to the value of Partition Ratio, while the Approximate Method has an identical value for the same Feasible

Energy. It is interesting that the best case of Partition Ratio in the previous method depends on the value of Feasible Energy. The best case of Partition Ratio in the previous method is identical in Fig. 14(a) and (b). In this experiment, the error ratio of the Approximate Method performed with Error Bound 5% is less than 5% in all cases.

VI. Conclusion

For mobile wireless systems relying on limited budget of energy, we propose an adaptive scheduling method of discrete energy-aware operation modes, called Optimal Method, which statistically maximizes the total utility of multiple tasks having probabilistic execution time with a given energy budget. In order to release the computation overhead of the Optimal Method at runtime, we also propose another scheduling method, called Approximate Method, which provides a near maximum utility within a given error bound and little runtime overhead. The proposed methods assign the operation mode having higher utility to the execution parts having higher probability, whereas the existing method assigns a fixed operation mode derived from the assumption that tasks are always executed for the worst-case execution time.

Our experiments on the adaptive MPEG video streaming of multimedia tasks show that the proposed methods provide higher utility, up to about 150%, than the previous method. Through extensive experiments, we derive the following implications:

- The error ratio of the Approximate Method is smaller than the given value of Error Bound in all cases.
- The Approximate Method gives larger increment of utility over the previous method, when the difference between the utility values of operation modes becomes larger.
- The Approximate Method gives larger increment of utility over the previous method, when the mean value of varying execution times is relative smaller than their worst value. The increment of utility is less sensitive to the mean value of varying execution times, if the given budget of energy is larger.
- The Approximate Method gives larger increment of utility over the previous method with more accurate expectation of the probability distribution of varying execution times. The increment of utility becomes less sensitive to the expectation accuracy of varying execution times, if the given budget of energy is larger.
- In the previous method, the best partitioning ratio of available energy among multiple tasks depends on the value of the available energy amount.

In future work, we will implement a prototype system equipped with the proposed method.

REFERENCES

[1] L. D. Paulson, "TV comes to the mobile phone," *Computer*, vol. 39, no. 4, pp. 13–16, 2006.

[2] MIT, "Special report: The 10 emerging technologies of 2008," Technology Review, March/April 2008. [Online]. Available: http://www.technologyreview.com/

[3] J. R. Lorch and A. J. Smith, "PACE: a new approach to dynamic voltage scaling," *IEEE Trans. Computers*, vol. 53, no. 7, pp. 856–869, July 2004.

[4] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time CPU scheduling for mobile multmedia systems," in *ACM Symp. Oper. Syst. Principles*, October 2003, pp. 149–163.

[5] R. Kravets and P. Krishnan, "Power management techniques for mobile communication," in *MobiCom '98: Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. New York, NY, USA: ACM Press, 1998, pp. 157–168.

[6] C. Poellabauer and K. Schwan, "Energy-aware media transcoding in wireless systems," in *IEEE International Conference on Pervasive Comp. Comm.*, March 2004, pp. 135–144.

[7] A. R. Lebeck, X. Fan, H. Zeng, and C. Ellis, "Power aware page allocation," *ACM SIGOPS Operating Systems Review*, vol. 34, no. 5, pp. 105–116, 2000.

[8] F. Douglis, P. Krishnan, and B. N. Bershad, "Adaptive disk spin-down policies for mobile computers," in *MLICS '95: Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing*. Berkeley, CA, USA: USENIX Association, 1995, pp. 121–137.

[9] H. Shim, N. Chang, and M. Pedram, "A backlight power management framework for battery-operated multimedia systems," *IEEE Design & Test of Computers*, vol. 21, no. 5, pp. 388–396, 2004.

[10] J. Brateman, C. Xian, and Y.-H. Lu, "Energy-efficient scheduling for autonomous mobile robots," in *VLSI-SoC '06: IFIP International Conference on Very Large Scale Integration*, October 2006, pp. 361–366.

[11] 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 *12th International Conference on Advanced Robotics*, July 2005, pp. 492–497.

[12] Z. Zhou, P. K. McKinley, and S. M. Sadjadi, "On quality-of-service and energy consumption tradeoffs in FEC-enabled audio streaming," in *IWQoS '04: Proceedings of the 12th IEEE International Workshop on Quality of Service*, Montreal, Canada, 2004.

[13] S. Mohapatra, N. Dutt, A. Nicolau, and N. Venkatasubramanian, "DYNAMO: A cross-layer framework for end-to-end QoS and energy optimization in mobile handheld devices," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 4, pp. 722–737, May 2007.

[14] S. Manolache, P. Eles, and Z. Peng, "Memory and time-efficient schedulability analysis of task sets with stochastic execution time," in *ECRTS '01: Proceedings of Euromicro Conference on Real-time Systems*, June 2001, pp. 164–173.

[15] J. A. Barnett, "Dynamic task-level voltage scheduling optimizations," *IEEE Trans. Computers*, vol. 54, no. 5, pp. 508–520, May 2005.

[16] K. Choi, K. Kim, and M. Pedram, "Energy-aware MPEG-4 FGS streaming," in *DAC '03: Proceedings of the 40th Annual Conference on Design Automation*. Los Alamitos, CA, USA: IEEE Computer Society, 2003, pp. 912–915.

[17] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Trans. Cirtuits and Systems for Video Technology*, vol. 11, no. 3, pp. 301–317, March 2001.

[18] L. M. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," in *IEEE INFOCOM*, April 2001, pp. 1548–1557.

[19] K. H. Kim and J. Kim, "An energy-efficient fec scheme for weakly hard real-time communications in wireless networks," in *RTCSA '06: Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 415–419.

[20] A. Davids, "Urban search and rescue robots: From tragedy to technology," *IEEE Intelligent Systems*, vol. 17, no. 2, pp. 81–83, 2002.

[21] J. Flinn and M. Satyanarayanan, "Managing battery lifetime with energy-aware adaptation," *ACM Trans. Comp. Syst.*, vol. 22, no. 2, pp. 137–179, May 2004.

[22] C. Krintz, Y. Wen, and R. Wolski, "Application-level prediction of battery dissipation," in *ISLPED '04: Proceedings of International Symposium on Low Power Electronics and Design*. New York, NY, USA: ACM Press, 2004, pp. 224–229.

[23] D. Rakhmatov, S. Vrudhula, and D. A. Wallach, "Battery lifetime prediction for energy-aware computing," in *ISLPED '02: Proceedings of International Symposium on Low Power Electronics and Design*. New York, NY, USA: ACM Press, 2002, pp. 154–159.

[24] Y. Wen, R. Wolski, and C. Krintz, "Online prediction of battery lifetime for embedded and mobile devices," in *PACS '03: Third International Workshp on Power-Aware Computer Systems, Lecture Notes in Computer Science 3164*, December 2003, pp. 57–72.

[25] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat, "ECOSystem: managing energy as a first class operating systems resource," *ACM SIGOPS Operating Systems Review*, vol. 30, no. 5, pp. 123–132, December 2002.

[26] T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu, and J.-S. Liu, "Probabilistic performance guarantee for real-time tasks with varying computation times," in *IEEE Real-Time Tech. App. Symp.*, May 1995, pp. 164–173.

[27] W. Y. Lee, H. Lee, and H. Kim, "Energy-aware QoS adjustment of multimedia tasks with uncertain execution time," in *ICCS '07: International Conference on Computational Science, Lecture Notes in Computer Science 4490*, May 2007, pp. 709–712.

[28] Q. Qin, Q. Wu, and M. Pedram, "Dynamic power management in a mobile multimedia system with guaranteed quality-of-service," in *DAC '01: Proceedings of the 38th Conference on Design Automation*, June 2001, pp. 834–839.

[29] G. Anastasi, A. Passarella, M. Conti, E. Gregori, and L. Pelusi, "A power-aware multimedia streaming protocol for mobile users," in *IEEE International Conference on Pervasive Services*, July 2005, pp. 371–380.

[30] Y.-H. Lu, L. Benini, and G. D. Micheli, "Power-aware operating systems for interactive systems," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 10, no. 2, pp. 119–134, 2002.

[31] M. Tamai, T. Sun, K. Yasumoto, N. Shibata, and M. Ito, "Engery-aware video streaming with QoS control for portable computing devices," in *ACM International Workshop Net. Oper. Syst. Support for Digital Audio and Videos*, 2004, pp. 68–73.

[32] C. Lee, J. Lehoczky, R. Rajkumar, and D. Siewiorek, "On quality of service optimization with discrete QoS options," in *IEEE Real-Time Tech. App. Symp.*, June 1999, pp. 276–286.

[33] C. Lee, J. Lehoczky, D. Siewiorek, R. Rajkumar, and J. Hansen, "A scalable solution to the multi-resource qos problem," in *RTSS '99: Proceedings of the 20th IEEE Real-Time Systems Symposium*. Washington, DC, USA: IEEE Computer Society, 1999, p. 315.

[34] C. Rusu, R. Melhem, and D. Mossé, "Maximizing rewards for real-time applications with energy constraints," *ACM Trans. Embedded Computing Systems*, vol. 2, no. 4, pp. 537–559, December 2003.

[35] P. Pillai, H. Huang, and K. G. Shin, "Energy-aware quality of service adaptation," Technical Report CSE-TR-479-03, University of Michigan. [Online]. Available: citeseer.ist.psu.edu/668908.html

[36] C. Rusu, R. Melhem, and D. Mossé, "Multi-version scheduling in rechargeable energy-aware real-time systems," *Journal of Embedded Computing*, vol. 1, no. 2, pp. 271–283, 2005.

[37] L. A. Cortés, P. Eles, and Z. Peng, "Quasi-static assignment of voltages and optional cycles for maximizing rewards in real-time systems with energy c-onstraints," in *DAC '05: Proceedings of the 42nd Annual Conference on Design Automation*. New York, NY, USA: ACM Press, 2005, pp. 889–894.

[38] C. Xian, Y.-H. Lu, and Z. Li, "Energy-aware scheduling for real-time multiprocessor systems with uncertain task execution time," in *DAC '07: Proceedings of the 44th Annual Conference on Design Automation*. New York, NY, USA: ACM, 2007, pp. 664–669.

[39] R. Xu, C. Xi, R. Melhem, and D. Mossé, "Practical PACE for embedded systems," in *EMSOFT '04: ACM International Conferenc on Embedded Software*, September 2004, pp. 54–63.

[40] Z. Lu, Y. Zhang, M. Stan, J. Lach, and K. Skadron, "Procrastinating voltage scheduling with discrete frequency sets," in *DATE '06: Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 2006, pp. 456–461.

[41] S. Krantz, S. Kress, and R. Kress, *Jenen's Inequality*. Birkhauser, 1999.

[42] "Featured videos," http://www.youtube.com.

[43] ESPN, "MLB scoreboard," http://sports-ak.espn.go.com/mlb/scoreboard.

APPENDIX

Lemma *1:* If $\frac{U_y-U_x}{R_y-R_x} < \frac{U_z-U_y}{R_z-R_y}$ for any $x < y < z$, the operation mode O_y is not used in the Optimal Schedule.

Proof: Let us assume that the Optimal Schedule uses the operation mode O_y from time t to time $(t+\alpha)$. Then we can make another schedule by replacing O_y with the operation mode O_z from time t to time $(t + \frac{(R_y-R_x)\cdot\alpha}{R_z})$ and with the operation mode O_x from time $(t + \frac{(R_y-R_x)\cdot\alpha}{R_z})$ to time $(t+\alpha)$. Because P_t is non-increasing, $\int_t^{t+\alpha} U_y \cdot P_t\, dt < (\int_t^{t+\frac{(R_y-R_x)\cdot\alpha}{R_z}} U_z \cdot P_t\, dt + \int_{t+\frac{(R_y-R_x)\cdot\alpha}{R_z}}^{t+\alpha} U_x \cdot P_t\, dt)$. That is, this schedule provides more utilities than the Optimal Schedule with the same amount of energy. This is a contradiction on the definition of the Optimal Schedule. Hence, the Optimal Schedule does not use the operation mode O_y such that $\frac{U_y-U_x}{R_y-R_x} < \frac{U_z-U_y}{R_z-R_y}$ for any $x < y < z$. ■

Lemma *2:* In the Optimal Schedule, $u(t_1) \geq u(t_2)$ and $r(t_1) \geq r(t_2)$ for any $t_1 < t_2$.

Proof: Let us assume that $u(t_1) < u(t_2)$ in the Optimal Schedule for any $t_1 < t_2$. We can make another schedule by replacing the operation mode at a time t_1 with that at a time t_2. Then this schedule provides more utilities than the Optimal Schedule, because P_t is non-increasing function of t. This is a contradiction on the definition of the Optimal Schedule. Hence, $u(t_1) \geq u(t_2)$ and $r(t_1) \geq r(t_2)$ for any $t_1 < t_2$ in the Optimal Schedule. ■

Theorem *1:* The values of π_k in Equation 3 have the following relationship in the Optimal Schedule: $P_{\pi_x} \cdot \frac{U_x-U_{x-1}}{R_x-R_{x-1}} = P_{\pi_y} \cdot \frac{U_y-U_{y-1}}{R_y-R_{y-1}}$ if $0 < \pi_x < \pi_y \leq W$ for any $N \geq x > y > 0$.

Proof: By Lemma 2, the utility values of the operation modes used in the Optimal Schedule construct a concave function with input value of their energy consumptions. Let us apply the Lagrange Multiplier Method [41] to solve Equation 3.

$$L(\pi_N, \pi_{N-1}, \cdots, x_1, \lambda) = (U_N - U_{N-1}) \cdot \int_0^{\pi_N} P_t\, dt + \cdots + (U_2 - U_1) \cdot \int_0^{\pi_2} P_t\, dt + (U_1 - U_0) \cdot \int_0^{\pi_1} P_t\, dt + \lambda \cdot (E - ((R_N - R_{N-1}) \cdot \pi_N + \cdots + (R_2 - R_1) \cdot \pi_2 + (R_1 - R_0) \cdot \pi_1)),$$

$$\frac{\partial L}{\partial \lambda} = E - ((R_N - R_{N-1}) \cdot \pi_N + \cdots + (R_2 - R_1) \cdot \pi_2 + R_1 \cdot \pi_1) = 0,$$

$$\text{and } \frac{\partial L}{\partial \pi_k} = \frac{(U_k - U_{k-1}) \cdot \int_0^{\pi_k} P_t\, dt}{\partial \pi_k} - \lambda \cdot (R_k - R_{k-1}) = 0 \text{ for } 1 \leq k \leq N.$$

Then, $\frac{\int_0^{\pi_k} P_t\, dt}{\partial \pi_k} = P_{\pi_k} = \lambda \cdot \frac{R_k-R_{k-1}}{U_k-U_{k-1}}$ for $1 \leq k \leq N$ and the value of P_{π_k} is proportional to the value of $\frac{R_k-R_{k-1}}{U_k-U_{k-1}}$. Hence, $P_{\pi_x} \cdot \frac{U_x-U_{x-1}}{R_x-R_{x-1}} = P_{\pi_y} \cdot \frac{U_y-U_{y-1}}{R_y-R_{y-1}}$ for any $N \geq x > y \geq 1$. ■

Lemma *3:* The utility values of the Optimal Schedule construct a concavely increasing function with

a given energy E as input value.

Proof: By Lemma 2, $u(t_1) \geq u(t_2)$ and $r(t_1) \geq r(t_2)$ for any $t_1 < t_2$ in the Optimal Schedule. Then, $u_1(t) \leq u_2(t)$ for any $E_1 < E_2$ and any t where $u_1(t)$ and $u_2(t)$ denote utility values of the operation mode assigned at a time t in the Optimal Schedule with energy amount E_1 and E_2, respectively. Because $\frac{u_1(t)}{r_1(t)} \geq \frac{u_2(t)}{r_2(t)}$ for all operation modes used in the Optimal Schedule by Lemma 1 and P_t is non-increasing along with values of t, the increment of additional utilities with additional unit energy is non-increasing as the value of E increases. This means that the utility values of the Optimal Schedule construct a concavely increasing function with a given energy E as input value. ■

Lemma *4:* When $M = 1$, the mean value of the error ratio is smaller than the Error Bound ϵ.

Proof: When $e_k < E_{max} < e_{k+1}$, the Approximate Method provides the utilities of ψ_k and the Optimal Method provides at most the utilities of ψ_{k+1}. Because $\tilde{\Psi} < \psi_{k+1}$ and $\hat{\Psi} = \psi_k$, $\frac{\tilde{\Psi}-\hat{\Psi}}{\tilde{\Psi}} < \frac{\psi_{k+1}-\psi_k}{\tilde{\Psi}} < \frac{\psi_{k+1}-\psi_k}{\psi_k}$. Because the values of ψ_k construct a concavely increasing function with input of e_k by Lemma 3, $\frac{\psi_k}{e_k} \geq \frac{\psi k+1}{e_{k+1}}$ and thus $\frac{\psi_k}{\psi k+1} \geq \frac{e_k}{e_{k+1}}$. Then, $\frac{\psi_{k+1}-\psi_k}{\psi_k} \leq \frac{e_{k+1}-e_k}{e_k} \leq \epsilon$. Hence, $\frac{\tilde{\Psi}-\hat{\Psi}}{\tilde{\Psi}} < \epsilon$. ■

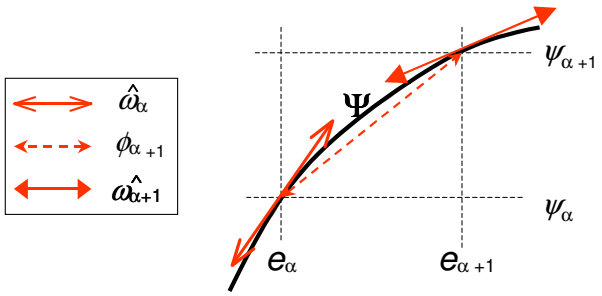

Fig. 15. Relationship among the values of $\hat{\omega^i_{\alpha_i}}$, $\phi^i_{\alpha_i+1}$ and $\hat{\omega^i_{\alpha_i+1}}$

Theorem *2:* The mean value of the error ratio is smaller than the Error Bound for any $M \geq 1$.

Proof: Lemma 4 shows that the error ratio of the Approximate Method is smaller than ϵ when $M = 1$. Let us check the case when $M \geq 2$. We use the following notation for clear explanation.

- $\tilde{E^i}$: the value of E^i assigned to i-th task by the Optimal Method.
- $\tilde{\Psi^i}$: the value of Ψ^i gained from i-th task in the Optimal Method.
- $\tilde{\omega^i}$: the instant derivative value of $\tilde{\Psi^i}$ with regard to the value of $\tilde{E^i}$.
- $\hat{E^i}$: the value of E^i assigned to i-th task by the Approximate Method.
- $\hat{\Psi^i}$: the value of Ψ^i gained from i-th task in the Approximate Method.

- $\hat{\omega^i}$: the instant derivative value of $\hat{\Psi}^i$ with regard to the value of $\hat{E}^i$.
- $\phi^i_{\alpha_i}$: the efficiency of energy utilization of the subrange finally selected by the Approximate Method for i-th task.

Then $\tilde{\Psi} = \sum_{i=1}^{M} \tilde{\Psi}^i$, $E_{max} = \sum_{i=1}^{M} \tilde{E}^i$, $\hat{\Psi} = \sum_{i=1}^{M} \hat{\Psi}^i$ and $E_{max} \geq \sum_{i=1}^{M} \hat{E}^i$.

Lemma 3 shows that the value of $\tilde{\omega^i}$ decreases as the value of $\tilde{E}^i$ increases for each i. It is clear that $\tilde{E}^i_x < \tilde{E}^i_y$ and $\tilde{\Psi}^i_x < \tilde{\Psi}^i_y$, if $\tilde{\omega^i_x} > \tilde{\omega^i_y}$ for each i. The Approximate Method preferentially selects the subrange having the largest value of ϕ^i among all tasks until $E_{max} < \sum_{i=1}^{M} \hat{E}^i$. Then $\hat{\omega^i} > \max_{1 \leq j \leq M}\{\phi^j_{\alpha_j+1}\} > \max_{1 \leq j \leq M}\{\hat{\omega^j_{\alpha_j+1}}\}$, such as shown in Fig. 15. The Optimal Method assigns additional energy to the task having the highest derivative value until $E_{max} = \sum_{i=1}^{M} \tilde{E}^i$. That is, $\tilde{\omega^i} > \max_{1 \leq j \leq M}\{\hat{\omega^j_{\alpha_j+1}}\} \geq \hat{\omega^i_{\alpha_i+1}}$ for each i. If $\tilde{\omega^i} > \hat{\omega^i_{\alpha_i+1}}$ for each i, then $\tilde{\Psi}^i < \psi^i_{\alpha_i+1}$ and $(\tilde{\Psi} - \hat{\Psi}) < \sum_{i=1}^{M}(\psi^i_{\alpha_i+1} - \psi^i_{\alpha_i})$. Finally, $\frac{\tilde{\Psi}-\hat{\Psi}}{\tilde{\Psi}} \leq \frac{\tilde{\Psi}-\hat{\Psi}}{\hat{\Psi}} < \frac{\sum_{i=1}^{M}(\tilde{\Psi}^i - \hat{\Psi}^i)}{\sum_{i=1}^{M}\hat{\Psi}^i}$ and $\frac{\sum_{i=1}^{M}(\tilde{\Psi}^i - \hat{\Psi}^i)}{\sum_{i=1}^{M}\hat{\Psi}^i} \leq \epsilon$ because $\frac{(\psi^i_{\alpha_i+1} - \psi^i_{\alpha_i})}{\psi^i_{\alpha_i}} \leq \epsilon$ by Lemma 4. Consequently, $\frac{\tilde{\Psi}-\hat{\Psi}}{\tilde{\Psi}} < \epsilon$ for any $M \geq 1$. ■