Dynamic Voltage and Frequency Scaling for Optimal Real-Time Scheduling on Multiprocessors

Kenji Funaoka, Akira Takeda, Shinpei Kato, and Nobuyuki Yamasaki
Graduate School of Science and Technology
Keio University, Yokohama, Japan
Email: {funaoka,takeda,shinpei,yamasaki}@ny.ics.keio.ac.jp

***Abstract*— Not only system performance but also energy efficiency is critically important for embedded systems. Optimal real-time scheduling is effective to not only schedulability improvement but also energy efficiency for the systems. In this paper, real-time dynamic voltage and frequency scaling (RT-DVFS) techniques based on the theoretically optimal real-time static voltage and frequency scaling (RT-SVFS) techniques proposed in our previous work are presented for multiprocessor systems. Simulation results show that RT-DVFS covers up the disadvantages of RT-SVFS in the sense that RT-DVFS are not practically affected by the difference among systems, whereas the energy consumption of RT-SVFS highly depends on the selectable processor frequency especially in high system utilization.**

I. Introduction

Multiprocessor architectures such as Simultaneous Multithreading (SMT) and Chip Multiprocessing (CMP) are becoming more attractive for intelligent embedded systems. It is important for embedded systems that real-time tasks such as robot controls and image processing meet their real-time constraints. Consequently powerful processors are desirable for these systems. On the other hand, the trade-off between system performance and energy efficiency is critically important for battery-based embedded systems. Real-time operating systems must go together with both requirements.

Real-time voltage and frequency scaling has been introduced to solve the problem. The processors of most recent computer systems are based on CMOS logic. Maximum processor frequency f depends on supply voltage V, and energy consumption E is proportional to processor frequency and square of supply voltage (i.e., $E \propto fV^2$) [1]. Real-time voltage and frequency scaling can potentially save energy at a cubic order, while they meet real-time constraints. Real-time voltage and frequency scaling is based on the essential characteristic of real-time tasks; namely the tasks can be executed slowly as long as all deadlines are met. This goal can be theoretically realized by real-time static voltage and frequency scaling (RT-SVFS). However there is still room for improvement on practical environments. The peak computing rate is much higher than the average throughput that must be sustained since most of real-time scheduling theories are based on the worst-case analysis to meet real-time constraints. Real-time dynamic voltage and frequency scaling (RT-DVFS) can save more energy by leveraging the characteristic.

Real-time voltage and frequency scaling techniques are constructed on real-time scheduling theories to meet real-time constraints. For single-processor systems, EDF [2] is an optimal real-time scheduling algorithm. On the other hand, EDF-FF and EDF-US, which are the extensions for multiprocessors, are not optimal [3], [4]. Approximately 50% processor time is wasted to meet real-time constraints on the algorithms at the worst-case. In other words, the algorithms theoretically require twice as many processors or powerful processors as optimal algorithms do. Accordingly the systems which leverage the algorithms expend more energy than ideal. Fortunately three optimal real-time scheduling algorithms for multiprocessors are presented (i.e., PD^2 [5], EKG [6], and LNREF [7], [8]). PD^2 incurs significant run-time overhead due to its quantum-based scheduling approach. EKG concentrates workloads on some processors due to the approach similar to partitioned scheduling. From the viewpoint of energy efficiency, energy consumption is minimized when the workloads are balanced among processors [9]. LNREF is an efficient algorithm on the balance as compared to the other optimal algorithms.

The remainder of this paper is organized as follows. The next section discusses the related work. In section III, we show the system model. Sections IV and V explain LNREF and RT-SVFS. In section VI, we present RT-DVFS techniques for optimal real-time scheduling on multiprocessors. Section VII evaluates the technique on practical environments. Finally we conclude with a summary and future work in section VIII.

II. Related Work

Many real-time voltage and frequency scaling techniques have been proposed in many aspects for single processor systems. Pillai and Shin [10] show a RT-SVFS technique based on the optimal real-time scheduling algorithm EDF [2]. Our uniform RT-SVFS on multiprocessors is analogous to EDF-based RT-SVFS. Real-time dynamic voltage and frequency scaling (RT-DVFS) techniques are also proposed for hard real-time systems [10], soft real-time systems [11], and dynamic real-time systems [12] to achieve more energy efficiency.

On the other hand, previous works such as [13], [14], [15] for multiprocessors are based on partitioned scheduling or non-optimal global scheduling. As mentioned above, the algorithms require twice as many processors or powerful processors as optimal algorithms do at the worst case. Our RT-SVFS techniques [16] are the first work which leverages optimal real-time scheduling on multiprocessors. No RT-DVFS technique based on optimal real-time scheduling is presented heretofore.

III. System Model

Optimal real-time voltage and frequency scaling on multiprocessors is a NP-hard partition problem since selectable processor frequency is discontinuous on practical systems. Consequently we assume that processor frequency can be controlled continuously at first. The effectiveness of the technique is shown in the simulation on practical environments.

The problem of scheduling a set of hard periodic tasks with voltage and frequency scaling on a multiprocessor system is presented. The system is modeled as a taskset $\mathbf{T} = \{T_1, \ldots, T_N\}$, which is a set of N periodic tasks to be executed on M processors $\mathbf{P} = \{P_1, \ldots, P_M\}$. Each processor P_k is characterized by continuous normalized processor frequency α_k $(0 \leq \alpha_k \leq 1)$. Each processor can execute at most one task simultaneously. Each task can not be executed in parallel among processors. Each task T_i is characterized by two parameters, worst-case execution time c_i and period p_i. A task T_i executed on a processor P_k requires c_i/α_k processor time at every p_i interval. The relative deadline d_i is equal to its period p_i. All tasks must complete the execution by the deadlines. The ratio c_i/p_i, denoted u_i $(0 < u_i \leq 1)$, is called task utilization. $U = \sum_{T_i \in \mathbf{T}} u_i$ denotes taskset utilization. Maximum task utilization is defined as $U_{\max} = \max\{u_i | T_i \in \mathbf{T}\}$. We assume that all tasks may be preempted and migrated among processors at any time, and are independent (i.e., they do not share resources and do not have any precedence).

In this paragraph, the differences between the system model and practical environments are discussed. (1) In practical environments, operable processor frequencies are discontinuous. The set of operable frequencies is defined as $\mathbf{f} = \{f_1, \ldots, f_m | f_1 < \cdots < f_m\}$. The lowest frequency $f_i \in \mathbf{f}$ such that $\alpha_k \leq f_i/f_m$ will be selected to bridge the gap between theory and practicality. (2) Processor throughput is not proportional to processor frequency in many cases as opposed to the system model described above. In practical systems, the frequency which can achieve the corresponding system throughput will be selected. (3) The system model assumes that no overhead occurs at run-time. In practical environments, the scaled frequency interferes with the scheduling even if the frequency is not changed dynamically. The worst-case overhead must be included in the worst-case execution time.

IV. T-N Plane Abstraction

T-N Plane Abstraction [7], [8] is an abstraction technique of real-time scheduling. T-N Plane Abstraction is based on the fluid scheduling model [17]. In the fluid scheduling model, each task is executed at a constant rate at all times. Figure 1 illustrates the difference between the fluid schedule and a practical schedule. The figure represents time on horizontal axis and task's remaining execution time on vertical axis. In practical scheduling, the task will be blocked by the other tasks as shown in the lower of the figure since a processor can execute only one task simultaneously. In the fluid scheduling model, each task T_i is executed along its fluid schedule path, the dotted line from (r_i, c_i) to $(r_i + p_i, 0)$, where r_i is the release time of the current job. It is impossible for

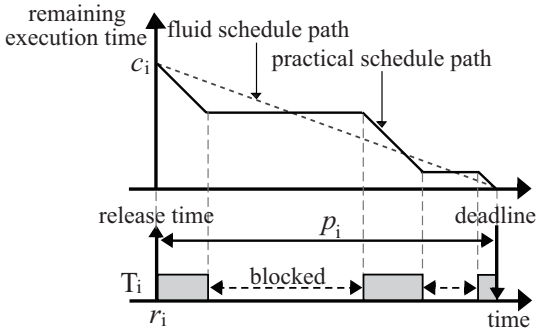

Fig. 1. Fluid schedule and a practical schedule.

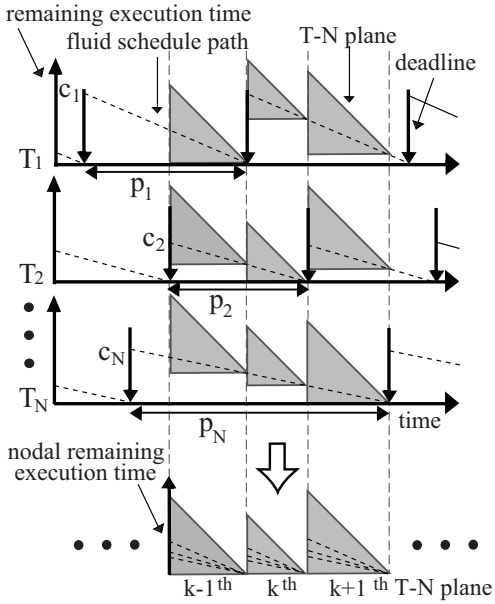

Fig. 2. T-N Plane Abstraction.

the fluid scheduling model to realize optimal schedule on practical systems since one processor must execute multiple tasks simultaneously. Notice that tasks need not constantly track their fluid schedule paths. Namely deadlines are the only time at which tasks must track the fluid schedule paths.

Figure 2 shows the way T-N Plane Abstraction abstracts real-time scheduling. All deadlines divide time as the vertical dotted lines as shown in the figure. The right isosceles triangles called T-N planes (Time and Nodal remaining execution time domain planes) are placed between every two consecutive deadlines. The rightmost vertex of the T-N plane coincides with the intersection of the fluid schedule path and the right side of the divided time-span. Since T-N planes in the same time-span are congruent, we have only to keep in mind an overlapped T-N plane shown in the lower of the figure at a time. The overlapped T-N plane represents time on horizontal axis and task's *nodal remaining execution time* on vertical axis. If the nodal remaining execution time becomes zero at the rightmost vertex of each T-N plane, the task execution follows the fluid schedule path at every deadline. Since T-N

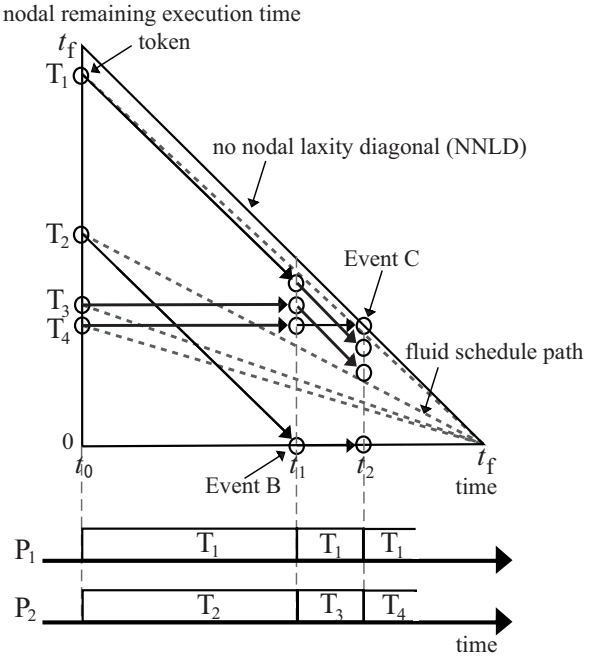

Fig. 3. Example of LNREF.

planes are repeated over time, good scheduling algorithms for a single T-N plane can help all tasks to meet their deadlines.

Figure 3 shows an overlapped T-N plane, where tokens representing tasks move from time t_0 to t_f. All tokens are on their fluid schedule paths at the beginning of the T-N plane. A token moves diagonally down if the task is executed; otherwise it moves horizontally. If all tokens arrive at the rightmost vertex, all tasks meet their deadlines. The successful arrival to the rightmost vertex is called *nodally feasible*. For the nodal feasibility, new events at which the scheduling decision is made again in the T-N plane are laid on. Event C and Event B occur when tokens hit the oblique side (NNLD) and the bottom side of the T-N plane, respectively. We assume that the jth event occurs at time t_j. M tokens which have the Largest Nodal Remaining Execution time are selected First (LNREF) on M processors at every event. LNREF is an optimal real-time scheduling algorithm for multiprocessors in the sense that any periodic taskset with utilization $U \leq M$ will be scheduled to meet all deadlines. If $U > M$, no algorithm can realize the successful schedule. Therefore we assume that $U \leq M$.

For example, there are four tasks (T_1, T_2, T_3, T_4) and two processors (P_1, P_2) as shown in Figure 3. Since there are two processors, two tasks can be executed simultaneously. At time t_0, T_1 and T_2 are executed on P_1 and P_2 in the LNREF order. Event B occurs at time t_1 since T_2 hits the bottom side of the T-N plane. Then two tasks T_1 and T_3 are selected again. Event C occurs at time t_2 since T_4 hits the oblique side (NNLD) of the T-N plane. The rescheduling is ingeminated at every event.

V. Static Voltage and Frequency Scaling

RT-SVFS proposed in our previous work [16] is based on the technique called T-N Plane Transformation. Figure 4 shows

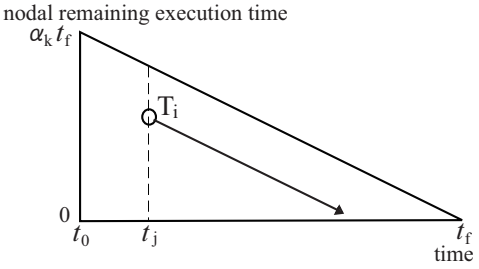

Fig. 4. T-N Plane Transformation ($\alpha_k = 0.5$).

Algorithm: DecideUniformFrequency
1: **foreach** 1...M **as** k
2: $\alpha_k = \max\{U_{\max}, U/M\}$
3: **end foreach**

Fig. 5. Uniform RT-SVFS.

a transformed T-N plane with frequency $\alpha_k = 0.5$. Selected tokens move diagonally down along the NNLD of the T-N plane. If α_k is given to the processor P_k, the voltage V_k is uniquely defined since maximum processor frequency f depends on supply voltage V. Thus the voltage and frequency scaling is equivalent to the frequency decision problem.

RT-SVFS techniques for different types of systems are presented in the following sections. SMT processors share resources among threads; thus the voltage and frequency can be controlled only uniformly among threads. On the other hand, the voltage and frequency can be controlled independently among processors in most of CMP and Symmetric Multiprocessing (SMP) processors. We first show a uniform RT-SVFS technique targeting for SMT processors. Then an independent RT-SVFS technique targeting for CMP and SMP processors is constructed upon the uniform RT-SVFS technique.

A. Uniform RT-SVFS

We assume that all processors have the same frequency $\alpha(= \alpha_1 = \ldots = \alpha_M)$. Figure 5 shows the uniform RT-SVFS algorithm. The algorithm is theoretically optimal as a static approach in the case where the voltage and frequency can be controlled only uniformly among threads or processors [16]. DecideUniformFrequency differs from the independent RT-SVFS technique described in the next section in the sense that DecideUniformFrequency is also optimal on practical systems, which can not control processor frequency continuously.

B. Independent RT-SVFS

The strategy of independent RT-SVFS is analogous to EKG [6]. All tasks are classified into either *heavy* or *light*. Each heavy task T_i is exclusively executed on one processor P_k with frequency $\alpha_k = u_i$. All light tasks are executed on the other processors by LNREF with DecideUniformFrequency.

Definitions for *heavy* and *light* are presented. $\mathbf{T}^{\text{heavy}}$ and $\mathbf{T}^{\text{light}}$ denote the sets of heavy and light tasks, respectively. Light taskset utilization is defined as $U^{\text{light}} = \sum_{T_i \in \mathbf{T}^{\text{light}}} u_i$. $U^{\text{light}}_{\max} = \max\{u_i | T_i \in \mathbf{T}^{\text{light}}\}$ denotes maximum utilization

Algorithm: DecideIndependentFrequency

Require: $u_1 \geq u_2 \geq \ldots \geq u_N$

$\mathbf{T}^{\text{heavy}} = \phi$
$\mathbf{T}^{\text{light}} = \mathbf{T}$
foreach $1...M$ **as** i
  if $U^{\text{light}}_{\max} > U^{\text{light}}/(M-H)$ **then**
    $\mathbf{T}^{\text{heavy}} = \mathbf{T}^{\text{heavy}} \cup \{T_i\}$
    $\mathbf{T}^{\text{light}} = \mathbf{T}^{\text{light}} \backslash \{T_i\}$
  else
    break
  end if
end foreach
foreach 1...M **as** k
  if P_k executes a heavy task T_k **then**
    $\alpha_k = u_k$
  else
    $\alpha_k = U^{\text{light}}/(M-H)$
  end if
end foreach

Fig. 6. Independent RT-SVFS.

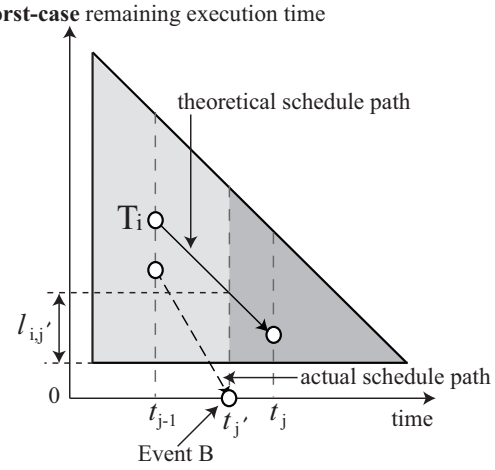

Fig. 7. Actual and theoretical schedules.

Algorithm: DecideUniformFrequencyDynamic

foreach 1...M **as** k
  $\alpha_k = \max\{S_{\max j'}, S_{j'}/M\}$
end foreach

Fig. 8. Uniform RT-DVFS.

of $\mathbf{T}^{\text{light}}$. The number of heavy tasks is represented as H. Heavy tasks are executed on the processors $(P_1, \ldots, P_H)$. The number of processors for the light taskset is $M - H$.

The independent RT-SVFS algorithm is shown in Figure 6. Tasks are sorted in decreasing utilization order at first, and we assume that all tasks are *light*. Then a light task T_i with utilization $u_i = U^{\text{light}}_{\max}$ is classified into *heavy* to dislodge the bottleneck of uniform RT-SVFS while $U^{\text{light}}_{\max} > U^{\text{light}}/(M-H)$ holds. The algorithm is theoretically optimal [16] in the sense that energy consumption is minimized. Furthermore the algorithm can solve the problem in polynomial time as opposed to the exhaustive algorithm [16].

VI. Dynamic Voltage and Frequency Scaling

RT-DVFS accommodates to the fluctuation of tasks' execution time. Since LNREF is based on the worst-case analysis, the voltage and frequency assigned by RT-SVFS is unnecessarily high in many cases. Each task T_i almost always completes the execution earlier than c_i is completely consumed because c_i represents "worst-case" execution time. Assume that a task T_i completes the execution at time $t_{j'}$ earlier than c_i is completely consumed as shown in Figure 7. Schedulers can detect the early completion, and Event B occurs at time $t_{j'}$. In this case, the time $l_{i,j'}$, which represents the theoretical value, can be reused for voltage and frequency scaling.

RT-SVFS presented in the previous section resolves the problem based on the task utilization, while RT-DVFS controls the voltage and frequency based on *nodal utilization* introduced by Cho et al [7]. $l_{i,j}$ denotes the *nodal remaining execution time* of a task T_i at time t_j. The *nodal utilization* of T_i at time t_j is defined as $r_{i,j} = l_{i,j}/(t_f - t_j)$. $S_j = \sum_{T_i \in \mathbf{T}} r_{i,j}$ denotes *total nodal utilization* at time t_j. $S_{\max j} = \max\{r_{i,j} | T_i \in \mathbf{T}\}$ denotes maximum nodal utilization of $\mathbf{T}$ at time t_j. The nodal utilization of light taskset at time t_j is defined as $S^{\text{light}}_j = \sum_{T_i \in \mathbf{T}^{\text{light}}} r_{i,j}$. $S^{\text{light}}_{\max j} = \max\{r_{i,j} | T_i \in \mathbf{T}^{\text{light}}\}$ denotes maximum nodal utilization of $\mathbf{T}^{\text{light}}$ at time t_j. The utilization of RT-SVFS can be changed to the nodal utilization to realize RT-DVFS as shown in Figures 8 and 9. The RT-DVFS techniques are performed at time t_0 and arbitrary time. The nodal remaining execution time of completing tasks is zero since the tasks do not need to be executed until next releases. Thus the nodal utilization of completing tasks is zero. RT-DVFS achieves lower energy consumption than RT-SVFS does since RT-DVFS takes account of the zero nodal utilization. If the voltage and frequency scaling is performed at time $t_{j'}$ shown in Figure 7, rescheduling by LNREF at time $t_{j'}$ is required since the T-N Plane is transformed at time $t_{j'}$.

A. Feasibility

The feasibilities of the RT-DVFS techniques are presented. The feasibility of the independent RT-SVFS technique is provided by the uniform independent RT-SVFS technique [16]; thus the independent RT-DVFS technique provides the feasibility if the uniform RT-DVFS technique provides the feasibility. In other words, we have only to keep in mind whether the uniform RT-DVFS technique can provide the feasibility.

Critical moment [7] is the first time when more than M tokens simultaneously hit the NNLD as shown in Figure 10. Cho et al. [7] show that critical moment is the sufficient and necessary condition where tokens are not nodally feasible in T-N Plane Abstraction. It is also available in the transformed T-N plane in the same manner since $\alpha_k \leq S_{\max 0}$ holds for all k in the uniform RT-DVFS technique (i.e., all tokens are in the transformed T-N plane at time t_0 [16]). Theorem 1 shows that the condition where a critical moment occurs is derived from total nodal utilization.

Theorem 1 (Total Nodal Utilization at Critical Moment): If a critical moment occurs at time t_j, $S_j > \alpha M$.

Algorithm: DecideIndependentFrequencyDynamic

Require: $r_{1,j'} \geq r_{2,j'} \geq \ldots \geq r_{N,j'}$

$\mathbf{T}^{\text{heavy}} = \phi$
$\mathbf{T}^{\text{light}} = \mathbf{T}$
foreach $1...M$ **as** i
  if $S^{\text{light}}_{\max\, j'} > S^{\text{light}}_{j'}/(M-H)$ **then**
    $\mathbf{T}^{\text{heavy}} = \mathbf{T}^{\text{heavy}} \cup \{T_i\}$
    $\mathbf{T}^{\text{light}} = \mathbf{T}^{\text{light}} \backslash \{T_i\}$
  else
    break
  end if
end foreach
foreach 1...M **as** k
  if P_k executes a heavy task T_k **then**
    $\alpha_k = r_{k,j}$
  else
    $\alpha_k = S^{\text{light}}_{j'}/(M-H)$
  end if
end foreach

Fig. 9. Independent RT-DVFS.

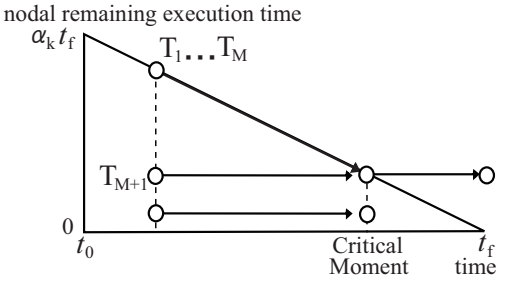

Fig. 10. Critical moment.

Proof: This proof is the similar fashion as that of LNREF [7] since frequency scaling is the lengthways time scaling of T-N planes. The remaining time $l_{i,j}$ of the tasks on the NNLD at the critical moment is $\alpha(t_f - t_j)$. Thus

$$S_j = \sum_{i=1}^{M+1} \frac{\alpha(t_f - t_j)}{t_f - t_j} + \sum_{i=M+2}^{N} \frac{l_{i,j}}{t_f - t_j} > \alpha M,$$

where $(T_1, \ldots, T_{M+1})$ are on the NNLD at time t_j. ■

The contraposition of Theorem 1 implies that no critical moment occurs if $S_j \leq \alpha M$ holds for all j. If $S_0 \leq \alpha M$ holds, no critical moment occurs since total nodal utilization is monotonically decreasing as shown in the following theorem.

Theorem 2 (Total Nodal Utilization): If $S_0 \leq \alpha M$, no critical moment occurs throughout the current T-N plane.

Proof: This proof is shown by the inductive method. The induction hypothesis is:

$$\begin{aligned} & S_{j-1} = S \\ \Leftrightarrow & \sum_{T_i \in \mathbf{T}} \frac{l_{i,j-1}}{t_f - t_{j-1}} = S \\ \Leftrightarrow & \sum_{T_i \in \mathbf{T}} l_{i,j-1} = S(t_f - t_{j-1}), \end{aligned} \tag{1}$$

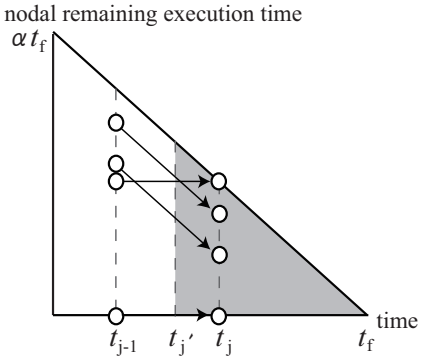

Fig. 11. RT-DVFS performed at time $t_{j'}$

where $S \leq \alpha M$. Assume that $N'(\leq M)$ tokens can be selected at time t_{j-1}. Since all tasks are in the T-N plane at first, $r_{i,j-1} \leq 1$ for all i; thus $S_{j-1} = S \leq \alpha N'$ is derived from $S \leq M$ and $r_{i,j-1} \leq 1$ for all i. The total remaining execution time decreases by $\alpha N'(t_j - t_{j-1})$ between time t_{j-1} and t_j. S_j is calculated as follows:

$$\begin{aligned} S_j &= \frac{1}{t_f - t_j} \sum_{T_i \in \mathbf{T}} l_{i,j} \\ &= \frac{1}{t_f - t_j} \left(\left(\sum_{T_i \in \mathbf{T}} l_{i,j-1} \right) - \alpha N'(t_j - t_{j-1}) \right) \\ &\Leftarrow \text{since Equation (1)} \\ &= \frac{S(t_f - t_{j-1}) - \alpha N'(t_j - t_{j-1})}{t_f - t_j}. \end{aligned}$$

We have

$$\begin{aligned} & S_{j-1} - S_j \\ =& S - \frac{S(t_f - t_{j-1}) - \alpha N'(t_j - t_{j-1})}{t_f - t_j} \\ =& \frac{t_j - t_{j-1}}{t_f - t_j}(\alpha N' - S) \geq 0 \\ \Rightarrow& S_{j-1} \geq S_j. \end{aligned}$$

If $S_0 \leq \alpha M$, no critical moment occurs from Theorem 1 since total nodal utilization is monotonically decreasing. ■

We assume that the voltage and frequency scaling is performed at time $t_{j'}$ as shown in Figure 11. Note that $t_{j'}$ does not represent the exact time. Consequently the voltage and frequency scaling can be performed at any time. All tokens remain nodally feasible as shown in the following theorem.

Theorem 3 (Feasibility in RT-DVFS): All tokens are nodally feasible even if the voltage and frequency scaling is performed at any time.

Proof: $S_{j'} \leq \alpha M$ holds before the voltage and frequency scaling since $S_{j-1} \leq \alpha M$ from Theorem 2 and no token hits the NNLD between time t_{j-1} and t_j. When the voltage and frequency scaling is performed at time $t_{j'}$, the shaded triangle shown in Figure 11 is transformed by the RT-DVFS algorithms. Assume that the frequency changes from α to α'.

TABLE I

SYSTEMS FOR SIMULATION.

System 1		System 2		System 3	
α	V	α	V	α	V
0.5	3	0.5	3	0.36	1.4
0.75	4	0.75	4	0.55	1.5
1.0	5	0.83	4.5	0.64	1.6
		1.0	5	0.73	1.7
				0.82	1.8
				0.91	1.9
				1.0	2.0

If $S_j \leq \alpha' M$ for all $j > j'$, all tokens are nodally feasible from Theorem 1. Since S_j is monotonically decreasing from Theorem 2, $S_j \leq \alpha' M$ holds for all $j > j'$. Therefore all tokens remain nodally feasible from Theorem 1. ■

Theorem 3 shows that the voltage and frequency scaling can be performed at any time. Frequent voltage and frequency scaling can reduce much energy consumption; however it incurs significant run-time overhead. Therefore the frequency of the voltage and frequency scaling is the trade-off between energy efficiency and system performance.

B. Practical Implementation

The previous section shows that the RT-DVFS techniques can be performed at any time. Assume that the RT-DVFS techniques are performed at every Δ interval. If $\Delta \to 0$, energy consumption based on the RT-DVFS techniques is minimized theoretically; however it is unrealistic from the viewpoint of system overhead. The balance between practicality and energy efficiency is important. The events (i.e., time t_0, Event C, and Event B) are good timings of voltage and frequency scaling since (1) additional rescheduling at voltage and frequency scaling is not required, and (2) the task sort of the independent RT-SVFS can be omitted since LNREF sorts tasks in decreasing nodal remaining execution time. The implementation can accommodate to the early completion shown in Figure 7 since Event B occurs at the early completion.

VII. SIMULATION

The advancement of RT-DVFS proposed in this paper is evaluated by comparing with RT-SVFS in terms of energy consumption in three systems shown in Table I. Each system has the operable sets of normalized frequency α and voltage V for each processor as shown in the table. The voltage and frequency scaling is performed only at every event to restrain the overhead of the voltage and frequency scaling. The simulation interval L is $[0, \min\{\text{lcm}\{p_i | T_i \in \mathbf{T}\}, 2^{32}\}]$. *Energy* represents the normalized energy consumption as follows:

$$\textit{EnergyRatio} = \frac{1}{L} \int_0^L \frac{\sum_{P_k \in \mathbf{P}} \alpha_k V_k^2}{M V_{\max}^2} \, dt,$$

where $V_{\max}$ represents the maximum voltage of each system.

Five cases are compared. **Static** represents the case where the voltage and frequency is controlled by RT-SVFS shown in Figure 6. The other four cases are that the voltage and frequency is controlled by RT-DVFS shown in Figure 9.

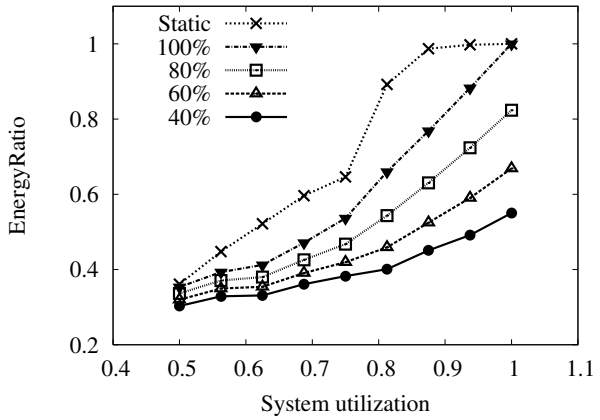

Fig. 12. Energy of System 1.

In RT-DVFS, the four cases where actual execution time uniformly varies in the range of $[0.4c_i, c_i]$ (**40%**), ($[0.6c_i, c_i]$) (**60%**), $[0.8c_i, c_i]$ (**80%**), and always c_i (**100%**) of worst-case execution time for each task T_i are presented in the results.

The other RT-DVFS algorithms shown in the previous papers can not be compared since the previous algorithms based on non-optimal real-time scheduling algorithms can not guarantee the schedulability in high system utilization.

A. Simulation Setup

Each simulation is modeled as four processors and a taskset. A taskset is initially empty. A new task is appended to the taskset as long as $U \leq U_{\text{target}}$, where U_{target} is the target utilization for each simulation. For each task T_i, its utilization u_i is computed based on a uniform distribution in the range of $[0.01, 0.1]$. Only the utilization of the last task is adjusted so that U becomes U_{target}. Each task T_i is generated with the period p_i in the integer range of $[100, 3000]$ and the worst-case execution time $c_i = u_i p_i$. In order to measure the average energy consumption, hundred simulations are conducted for each system utilization U/M between 0.5 and 1.0 where most traditional algorithms can not guarantee the schedulability.

B. Simulation Results

Figures 12, 13, and 14 show the results corresponding to that of Systems 1, 2, and 3, respectively. The figures show system utilization U/M on the horizontal axis and the average *Energy* on the vertical axis. The results of the three systems show that the results of RT-SVFS highly depend on the selectable voltage and frequency level in the higher system utilization. The reason comes from the fact that RT-SVFS must take account of the worst case execution time and can not change the voltage and frequency for ever. On the other hand, RT-DVFS can accommodate to dynamic environments even if actual execution time is always equal to worst-case execution time (**100%**). The reason comes from the fact that the voltage and frequency levels selected by RT-DVFS are unnecessarily high only at first; namely most of tasks complete the execution earlier than the ideal case since the set of higher voltage

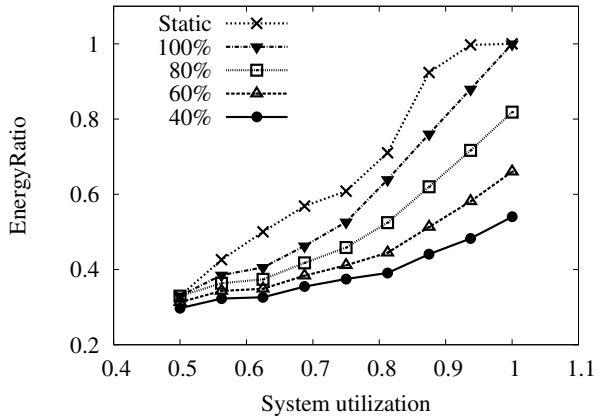

Fig. 13. Energy of System 2.

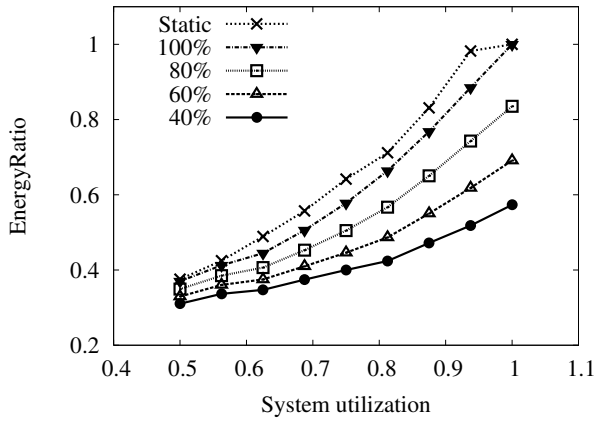

Fig. 14. Energy of System 3.

and frequency is selected to bridge the gap between theory and practicality as shown in the system model. After that, RT-DVFS can decrease the voltage and frequency. Therefore the curves of RT-DVFS smoothly decrease as compared to that of RT-SVFS. RT-DVFS can linearly reduce the energy consumption when actual execution time varies uniformly. In the lower system utilization, all results are mostly the same since the selectable voltage and frequency is bounded by the sets of lowest voltage and frequency shown in Table I.

VIII. Conclusions and Future Work

In this paper, we have presented two algorithms for real-time dynamic voltage and frequency scaling based on an optimal real-time scheduling algorithm for multiprocessors. The RT-DVFS algorithms proposed in this paper is based on the optimal RT-SVFS techniques proposed in our previous work. The algorithms can be applied to both uniform settings and independent settings of the voltage and frequency among processors. RT-DVFS can accommodate to dynamic environments even if actual execution time is always equal to worst-case execution time. Additionally RT-DVFS can linearly reduce the energy consumption when actual execution time varies.

The practical implementation is a topic for the future work. The frequency of the voltage and frequency scaling is a trade-off between energy efficiency and system performance. Frequent voltage and frequency scaling incurs significant run-time overhead due to the physical limitation of processors. Therefore effective implementation and appropriate control of the voltage and frequency are required in practical systems.

Acknowledgement

This research is supported by CREST, JST.

References

[1] T. D. Burd and R. W. Brodersen, "Energy Efficient CMOS Microprocessor Design," in *Proc. of the 28th Annual Hawaii International Conference on System Sciences*, Jan. 1995, pp. 288–297.

[2] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *Journal of the ACM*, pp. 46–61, Jan. 1973.

[3] J. M. Lopez, M. Garcia, J. L. Diaz, and D. F. Garcia, "Worst-Case Utilization Bound for EDF Scheduling on Real-Time Multiprocessor Systems," in *Proc. of the 12th Euromicro Conference on Real-Time Systems*, June 2000, pp. 25–33.

[4] T. P. Baker, "An Analysis of EDF Schedulability on a Multiprocessor," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 8, pp. 760–768, Aug. 2005.

[5] J. H. Anderson and A. Srinivasan, "Early-Release Fair Scheduling," in *Proc. of the 12th Euromicro Conference on Real-Time Systems*, June 2000, pp. 35–43.

[6] B. Andersson and E. Tovar, "Multiprocessor Scheduling with Few Preemptions," in *Proc. of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, Aug. 2006, pp. 322–334.

[7] H. Cho, B. Ravindran, and E. D. Jensen, "An Optimal Real-Time Scheduling Algorithm for Multiprocessors," in *Proc. of the 27th IEEE Real-Time Systems Symposium*, Dec. 2006, pp. 101–110.

[8] ——, "Synchronization for an Optimal Real-Time Scheduling Algorithm on Multiprocessors," in *Proc. of the 2nd IEEE International Symposium on Industrial Embedded Systems*, 2007, pp. 9–16.

[9] H. Aydin and Q. Yang, "Energy-Aware Partitioning for Multiprocessor Real-Time Systems," in *Proc. of the 17th IEEE International Parallel and Distributed Processing Symposium*, Sept. 2003, pp. 22–26.

[10] P. Pillai and K. G. Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems," in *Proc. of the ACM Symposium on Operating Systems Principles*, 2001, pp. 89–102.

[11] J. A. Stankovic, C. Lu, and S. H. Son, "The Case for Feedback Control Real-Time Scheduling," in *Proc. of the 11th Euromicro Conference on Real-Time Systems*, June 1999, pp. 11–20.

[12] C. H. Lee and K. G. Shin, "On-Line Dynamic Voltage Scaling for Hard Real-Time Systems Using the EDF Algorithm," in *Proc. of the 25th IEEE Real-Time Systems Symposium*, Dec. 2004, pp. 319–335.

[13] C. Xian, Y.-H. Lu, and Z. Li, "Energy-Aware Scheduling for Real-Time Multiprocessor Systems with Uncertain Task Execution Time," in *Proc. of the 44th ACM/IEEE Design Automation Conrefence*, 2007, pp. 664–669.

[14] J.-J. Chen and T.-W. Kuo, "Allocation Cost Minimization for Periodic Hard Real-Time Tasks in Energy-Constrained DVS Systems," in *Proc. of the IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2006, pp. 255–260.

[15] D. Shu, R. Melhem, and B. R. Childers, "Scheduling with Dynamic Voltage/Speed Adjustment Using Slack Reclamation in Multiprocessor Real-Time Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 7, pp. 686–700, July 2003.

[16] K. Funaoka, S. Kato, and N. Yamasaki, "Energy-Efficient Optimal Real-Time Scheduling on Multiprocessors," in *Proc. of the 11th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, May 2008.

[17] P. Holman and J. H. Anderson, "Adapting Pfair Scheduling for Symmetric Multiprocessors," *Journal of Embedded Computing*, vol. 1, no. 4, pp. 543–564, May 2005.