Vol.2010-OS-113 No.15
2010/1/28

EDF スケジューリングにおける DVFS を用いた CPU 省電力化手法

林　　和宏[†1]　並木　美太郎[†1]

計算機システムにおける代表的な省電力化技術の一つに，CPU の DVFS 機能がある．DVFS による電力削減にはプログラム実行性能の低下を伴うため，性能制約の厳しいリアルタイムシステムにおいては，特に慎重な制御が求められる．本論文では，シングルコア CPU 上の EDF スケジューリングを対象とした OS スケジューラによる DVFS 制御手法を提案する．提案手法では，スケジューリングにおける余裕時間を効率的に見積もることで，極力少ない計算量で効果的な CPU 省電力化を行う．さらに，タスクごとの実行時情報に基づいた性能予測を行うことで，従来手法よりも高い省電力性とリアルタイム性を実現する．

実機およびシミュレーションによる評価では，特にメモリバウンドなタスクに対し，従来手法と比較して最大 13.4% の CPU エネルギー消費量と，11.6% のデッドラインミス率を削減した．

A CPU Power-Saving Task Scheduler with DVFS for EDF Scheduling Algorithm

Kazuhiro Hayashi[†1] and Mitaro Namiki [†1]

DVFS is an effective technique to reduce CPU power consumption. DVFS control that keeps deadlines is needed on real-time systems because DVFS causes performance decrease. In this paper, we propose a DVFS control algorithm by OS task scheduler for EDF scheduling on a uni-core processor. The proposed method reduces CPU power consumption effectively with less calculation overhead by estimating scheduling slack time efficiently. In addition, the proposed method achieves more power reduction and lower deadline miss ratio than traditional way predicting each task performance based on task behaviors.

In evaluation, the proposed method reduced at most 13.4% CPU energy consumption and 11.6% deadline miss ratio of memory bound task sets, compared to the traditional way.

1. 緒　　言

近年，計算機システムの性能向上に伴う消費電力の増大により，サーバシステムなどでは消費電力と発熱の増大が問題視されている．また，携帯機器の普及にも伴い，電力効率向上のための技術は日々その必要性を増している．様々な分野で計算機システムにおける省電力化の必要性が高まる中，ハードウェア・ソフトウェアの両観点から省電力化に関する多くの研究・開発が行われている．ソフトウェアによる省電力化の例として，CPU の電圧・周波数動的調整機能である DVFS（Dynamic Voltage and Frequency Scaling）を用いた電力制御が代表的である．DVFS では，CPU の動作電圧・周波数を低下することで効果的な動的電力の削減が可能であるが，一方で CPU 周波数低下によるプログラム実行時性能の低下を招く．このため，ソフトウェアは電力と性能のトレードオフを十分に考慮したうえで DVFS 制御を行う必要がある．特に，性能制約の厳しいリアルタイムシステムにおいては，軽率な周波数低下はデッドラインミスの原因となるため，システム状況に応じた慎重な DVFS 制御が求められる．そこで，本論文では，リアルタイムシステムを対象として，性能への影響を十分に考慮しつつ効果的な CPU 省電力化を実現する DVFS 制御手法を提案する．

リアルタイムスケジューリングは，タスク優先度の決定方法によって，固定優先度スケジューリングと動的優先度スケジューリングに分類される．代表的なアルゴリズムとして，固定優先度スケジューリングには RM（Rate-Monotonic），動的優先度スケジューリングには EDF（Earliest Deadline First）がある．本論文では，上記の中でも EDF スケジューリングを対象とした DVFS 制御手法を提案する．EDF は，デッドライン時刻の早い順にタスクに高い優先度を動的に与えるアルゴリズムである．EDF では，全タスクの合計 CPU 使用率が 100%を超えない限りすべてのタスクがデッドラインを守ることを保証できるため，CPU 資源を最大限活用できるという利点がある．

EDF スケジューリングを対象とした DVFS 制御については，過去に様々な手法が提案されている．主に，CPU 使用率の合計が 100%を超えない範囲で全タスクの動作周波数を均一に低下させる手法や，DVFS 制御時刻におけるシステムの余裕時間を計算し，これを使い切るように周波数を低下させる手法が存在する．リアルタイムスケジューリングにおける DVFS 制御では，各タスクのリアルタイム性を保証したうえで，どの程度正確にシステム

†1 東京農工大学
Tokyo University of Agriculture and Technology

の余裕時間を見積もれるかが省電力性に大きく影響する．しかし，余裕時間は厳密に計算するほど計算オーバヘッドが大きくなる．また，動作周波数変更時のタスク実行時性能をいかに正確に予測できるかが，アルゴリズムの省電力性とリアルタイム性に大きく左右する．多くの従来手法では，タスクごとの動作特性を無視した性能予測を行っている．しかし，実際の性能はタスクごとに異なる場合が多く，従来手法では予測精度に問題がある．加えて，従来手法の多くは，その周波数決定プロセスにおいて，タスク単位の性能予測が困難な設計となっている．

本論文では，OS スケジューラよるタスク単位の DVFS 制御手法を提案する．提案手法では，EDF スケジューリングにおける余裕時間を効率的に見積もることで，極力少ない計算オーバヘッドで効果的な省電力化を実現する．加えて，タスク単位の動作特性に基づいた性能予測を行うことで，従来予測よりも高い省電力性とリアルタイム性を実現する．本論文では，複数の CPU アーキテクチャに対して性能予測モデルを構築し，CPU エネルギー消費量とデッドラインミス率の測定を通じて提案手法の評価を行う．

2. 関 連 研 究

Pillai らの論文[1] では，周期タスクで構成されるハードリアルタイムシステムを対象とした複数の DVFS 制御手法が提案されている．いずれの手法も，f におけるタスクの実行時間は，最高周波数 $f_{\max}$ 時に対して $f_{\max}/f$ 増加するものとしてタスク実行時間を予測している．最も単純な制御手法に，全タスクの CPU 使用率から動作周波数を静的に決定する手法（以下，`STATIC`）がある．`STATIC` では，全タスクの CPU 使用率合計と $f_{\max}/f$ の積が 100%以下となる最低の周波数 f を，全タスクの動作周波数として決定する．`STATIC` は，ワークロードの動的な変化に対応できず，省電力効果は低い．Cycle-conserving（以下，`CC`）は，既に実行を完了したタスクは実際に要した実行時間，それ以外は最悪実行時間を用いて CPU 使用率を計算し，CPU 使用率合計と $f_{\max}/f$ の積が 100%以下となる最低の周波数を動的に選択する手法である．`CC` は，ワークロードの動的な変化にも対応できるが，多くの場合で，システムの余裕時間を効果的に見積ることができない．Look-Ahead（以下，`LA`）は，タスクの絶対デッドラインに基づいて動的に周波数を決定する手法であり，多くの場合で `CC` よりもシステム余裕時間を効果的に見積もれるため，高い省電力性を持つ．具体的には，デッドラインが遅いタスクから順に，最も早いデッドラインと自身のデッドラインの区間内で，できる限りタスクの残り実行時間を保証していく．最終的に，保証しきれなかったタスク実行時間の合計と $f_{\max}/f$ の積が，現在時刻から最も早いデッドラインまでの区間内に収まる最低の周波数を選択する．

Poellabauer らは，周期タスクからなるソフトリアルタイムシステムを対象に，メモリバウンドなタスク向けの DVFS 制御手法を提案している[2]．一般的に，タスクのメモリアクセス率は，CPU 周波数の変更に伴うタスク性能の変化量に大きく影響する．CPU の各電圧・周波数状態ごとに利用可能な最大のバス周波数が異なるようなアーキテクチャにおいては，メモリバウンドなタスクほど性能がバス周波数に大きく依存する．以上のことに着目し，Poellabauer らの手法では，CPU・バス周波数変更時のタスク性能をタスクのメモリアクセス率から予測することにより，その予測精度を高めている．一方で，タスク実行時間が周期ごとに大きく変化するような環境は想定しておらず，そのような環境下では実行時間誤差を十分に考慮した周波数選択が行えない可能性がある．また，タスクの最悪実行時間を用いずに CPU 使用率を計算するソフトリアルタイム向けの手法なため，厳密な意味でタスクのデッドラインを保証できない．

この他，全タスクが最悪実行時間で動作した場合に対する実際の余裕時間を概算し，これを使い切るようにカレントタスクの実行時間を伸ばす手法[3] や，1 タスクの実行領域を二つに分割し，前半は極力低い周波数，後半は最高周波数で動作させながら余裕時間を消費させるフィードバック制御手法[4] などが存在する．

提案手法では，タスクの最悪実行時間を用いて，EDF スケジューリングにおける余裕時間を低オーバヘッドで見積もることで，リアルタイム性を保証しながら効率的な DVFS 省電力化を行う．さらに，周波数変更時の性能をタスク単位の動作情報に基づいて予測することにより，従来の予測手法よりも予測精度を向上させ，結果として，スケジューリングにおける合計エネルギー消費量とデッドラインミスの削減をねらう．評価では，本節で述べた従来手法[1] のうち，`STATIC` や `LA` などと直接比較を行い，提案手法のオーバヘッドや電力効率，リアルタイム性について考察する．

3. 設　　計

3.1 システムモデル

提案手法では，シングルコアプロセッサ上で，合計 n 個の周期タスク $\tau_i\,(1 \leq i \leq n)$ が EDF アルゴリズムに従ってスケジューリングされるリアルタイムシステムを想定する．すべてのタスクは常にプリエンプト可能であり，タスク間は独立であると仮定する．このため，タスクは I/O 処理を行わないものとする．

各タスク τ_i は，周期 p_i ごとに繰り返し起動される．現在時刻 t におけるタスク τ_i のデッ

ドライン時刻 d_i は，τ_i の現在の周期の終了時刻（次の周期の開始時刻）に等しい．また，各タスクの最悪実行時間 w_i の値はあらかじめ得られているものとする．

リアルタイムスケジューラには，ウェイト・キュー T_{W} とレディ・キュー T_{R} の 2 種類のタスクが存在するものとする．T_{W} には現在の周期の実行を既に完了して次の周期の開始を待機しているタスク，T_{R} には実行を完了していないタスクがそれぞれ格納され，いずれも d_i が早い順にソートされているものとする．

3.2 DVFS 制御における基本方針

提案手法では，スケジューラ起動時の中でも，特にコンテキストスイッチ発生時にのみ DVFS 制御処理を行う．EDF スケジューリングにおいては，タスクの実行完了時と，最悪時にすべてのデッドライン時刻でコンテキストスイッチが発生する．`LA` では，タスク実行完了時と全デッドライン時刻で常に DVFS 制御を行うが，提案手法ではデッドライン時刻でコンテキストスイッチが発生しない可能性がある分，`LA` よりも制御処理回数を削減でき，結果的に DVFS 制御のオーバヘッドを削減できる可能性がある．

提案手法では，コンテキストスイッチごとにカレントタスク τ_{cur} に適した動作周波数を決定する．このとき，周波数決定のポリシとして，(a) 複数のタスクの周波数を均等に低下させる方法と，(b) 特定のタスクのみ周波数を低下させる方法が考えられる．タスク実行時間が常に一定であれば，(a) の方が全体の消費エネルギーは小さくなる[5]．しかし，実際のシステムでは，タスクの実行時間は周期ごとに変動する場合が多い．例えば，動画・音声のデコード処理や，リアルタイム画像処理などでは，タスク実行時間は入力データによって大きく変化する．タスクの実際の実行時間 c_i^{act} は，特に小さいケースで w_i の 13% 程度になることがわかっている[6]．c_i^{act} と w_i の差が大きい場合には，(b') τ_{cur} の動作周波数を優先的に下げることで，(a) よりも全体の消費エネルギーを削減できる可能性が高い．単純な例として，$(w_i, p_i) = (1, 4)$ なる二つのタスク τ_1，τ_2 を考える．両タスクについて，$c_i^{\mathrm{act}} = kw_i\,(0 < k \leq 1)$ であるとすると，1 周期に占めるアイドル時間 c_{idle} は，(a) では $4 - k(6 - 2k)$，(b') では $4 - k(7 - 3k)$ となる（**図 1**）．なお，周波数 f における実行時間増加率は $f_{\max}/f$ として計算している．$0 < k \leq 1$ ならば常に $k(7 - 3k) \geq k(6 - 2k)$ となり，(b') の方が結果的に全体のアイドル時間を短くできることがわかる．また，タスク単位の性能予測を行うことを考えた場合，(a) では 1 回の制御において全タスクに対する性能予測が必要となるためオーバヘッドが大きいのに対し，(b') では τ_{cur} に対する 1 回の予測のみで済む．以上の理由から，提案手法では DVFS 制御において τ_{cur} の動作周波数を優先的に下げるアプローチをとる．

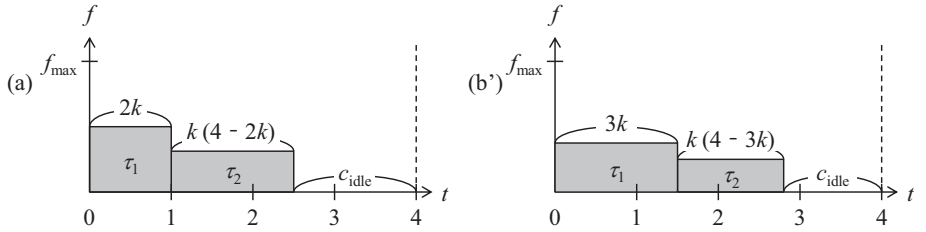

図 1 サンプルタスク実行時の周波数制御例（$k = 0.5$）
Fig. 1 Example of frequency scaling for the execution of sample tasks ($k = 0.5$)

τ_{cur} の動作周波数を決定するうえで，下記の 2 点が主な問題点となる．

(1) τ_{cur} が利用可能な実行時間（実行可能時間 c_{ava}）の計算
(2) 周波数低下に伴うタスク性能低下率の予測

(1) は，τ_{cur} 以外のタスクに必要な最低限の実行時間を保証した上で，τ_{cur} がより低い周波数で動作できるよう，極力大きく見積る必要がある．(2) は，τ_{cur} の実行時間が c_{ava} を超えない範囲でできる限り低い周波数を選択できるよう，τ_{cur} の実行時情報に基づいた正確な予測が要求される．両者は独立した問題であり，いずれもアルゴリズムの性能に大きく影響する．以降の節では，各々の実現方法について述べる．

3.3 実行可能時間の計算

カレントタスク τ_{cur} の動作周波数を極力下げるためには，他の全タスクがデッドラインを守るのに最低限必要な実行時間を極力少なく（c_{ava} を極力大きく）見積る必要がある．c_{ava} の値を厳密に計算することは可能だが，その計算量は，超周期（Hyper-period：全タスク周期の最小公倍数期間）内に存在する全タスク実体数を $N\,(N \gg n)$ として $O(N)$ であり[3]，実用性を考えると現実的ではない．そこで，提案手法では c_{ava} を近似的かつ効果的に見積もることで，極力少ないオーバヘッドで高い省電力性を実現する．

今，カレントタスクの絶対デッドラインを d_{cur} とする．コンテキストスイッチ時刻 t において，$T_{\mathrm{R}} = \phi$ かつ $d_i \geq d_{\mathrm{cur}}\,(\tau_i \in T_{\mathrm{W}})$ となるとき，区間 $[t, d_{\mathrm{cur}}]$ 内で実行可能なタスクは τ_{cur} 以外に存在しないため，c_{ava} は下記で与えられる．

$$c_{\mathrm{ava}} = d_{\mathrm{cur}} - t \tag{1}$$

$d_i < d_{\mathrm{cur}}\,(\tau_i \in T'_{\mathrm{W}} \subseteq T_{\mathrm{W}})$ となるようなタスクセット T'_{W} が存在する場合，これらのタスクは区間 $[t, d_{\mathrm{cur}}]$ で実行される可能性がある．このため，タスクセット T'_{W} が必要とする最低限の実行時間を保証した上で，c_{ava} を決定する必要がある．このとき，タスクセッ

ト T'_{W} に対して保証すべき合計実行時間を r_{W} とする．今，区間 $[t, d_{\mathrm{cur}}]$ におけるタスク τ_i のデッドライン時刻を t に近いものから順に $d_{i,0}, d_{i,1}, \cdots, d_{i,m}$ とする（**図 2**）．各タスク τ_i は，区間 $[d_{i,0}, d_{i,m}]$ において最悪 CPU 使用率 w_i/p_i 分の実行時間 $w_i \cdot m$ が保証されなければならない．加えて，$d_{i,m} < d_{\mathrm{cur}}$ となるタスクについては，区間 $[d_{\mathrm{cur}}, d_{i,m+1}]$ でできる限り実行時間を保証し，保証しきれなかった残り実行時間分を区間 $[d_{i,m}, d_{\mathrm{cur}}]$ で保証する必要がある．しかし，区間 $[d_{\mathrm{cur}}, d_{i,m+1}]$ で保証可能な最大実行時間の厳密な計算には，大きなオーバヘッドを必要とする．具体的には，T'_{W} に属するタスクの仮のデッドラインを $d_{i,m+1}$ としたうえで，全タスクをデッドライン順にソートする必要がある．提案手法では，オーバヘッドを極力小さくするため，最悪 CPU 使用率分の実行時間を区間 $[d_{i,m}, d_{\mathrm{cur}}]$ で保証するものとする．結果として，r_{W} は次式で与えられる．

$$r_{\mathrm{W}} = \sum_{\tau_i \in T'_{\mathrm{W}}} (d_{\mathrm{cur}} - d_i) \cdot \frac{w_i}{p_i} \tag{2}$$

ここで，定数 w_i/p_i は小数点以下切り上げの値とする．

$T_{\mathrm{R}} \neq \phi$ のときは，タスクセット T_{R} が必要とする最低限の実行時間 r_{R} を保証した上で，c_{ava} を決定する必要がある．タスク $\tau_i \in T_{\mathrm{R}}$ は他のタスクによってプリエンプトされた可能性があるため，w_i を基準とした残り実行時間を c_i^{left} とすると，$c_i^{\mathrm{left}} \leq w_i\ (\tau_i \in T_{\mathrm{R}})$ である．これらのタスクはいずれも $d_i \geq d_{\mathrm{cur}}$ であり（**図 3**），c_{ava} を極力大きくするためには，区間 $[d_{\mathrm{cur}}, d_i]$ で各タスクの実行時間をできる限り保証することが望ましい．しかし，既に述べたように，上記の保証時間を厳密に計算するには大きなオーバヘッドを要する．提案手法では，まず T_{R} に属する各タスクについて，最悪 CPU 使用率分の実行時間を区間 $[d_{\mathrm{cur}}, d_i]$ で保証し，c_i^{left} をすべて保証できない場合は，不足分を区間 $[t, d_{\mathrm{cur}}]$ で保証する．つまり，タスク $\tau_i \in T_{\mathrm{R}}$ に対し区間 $[t, d_{\mathrm{cur}}]$ で保証すべき実行時間を r_i とすると

$$r_i = c_i^{\mathrm{left}} - (d_i - d_{\mathrm{cur}}) \cdot \frac{w_i}{p_i} \quad (\text{ただし } r_i < 0 \text{ ならば } r_i = 0) \tag{3}$$

となり，r_{R} はこれらの総和

$$r_{\mathrm{R}} = \sum_{\tau_i \in T_{\mathrm{R}}} r_i \tag{4}$$

として与えられる．

最終的に，τ_{cur} に与えられる実行可能時間 c_{ava} は，式 (1)，式 (2)，式 (4) より

$$c_{\mathrm{ava}} = (d_{\mathrm{cur}} - t) - r_{\mathrm{R}} - r_{\mathrm{W}} \tag{5}$$

として計算できる．本手法では，τ_{cur} の実行時間が c_{ava} を超えない限り，T_{R} および T_{W} に属する全タスクのデッドラインを保証できる．また，c_{ava} の計算オーバヘッドは，T_{R}，T'_{W} に属するタスク数をそれぞれ n_{R}，$n_{\mathrm{W}'}$ として $O(n_{\mathrm{R}} + n_{\mathrm{W}'})$ であり，$n_{\mathrm{R}} + n_{\mathrm{W}'} \leq n$ である．

3.4 周波数変更時の性能予測

スケジューラは，3.3 節で得られた実行可能時間 c_{ava} を用いて，カレントタスク τ_{cur} の動作周波数を低下させる．今，周波数状態 f における，最高周波数 $f_{\max}$ 時に対するタスク τ_i の性能比を $y(f, \tau_i)$ とする．τ_{cur} の動作周波数は，その残り実行時間を $c_{\mathrm{cur}}^{\mathrm{left}}$ として

$$\frac{c_{\mathrm{cur}}^{\mathrm{left}}}{y(f, \tau_i)} \leq c_{\mathrm{ava}} \tag{6}$$

を満たす f のうち，電圧が最小になるものとして決定できる．ここで問題となるのが，周波数変更に伴う性能予測手法，すなわち $y(f, \tau_i)$ の定義である．

従来の多くの研究[1)3)4)] では，性能比が単純に CPU 周波数の比になるものとして全タスクに対し同一の性能予測を行っている．つまり，

$$y(f, \tau_i) = \frac{f}{f_{\max}} \tag{7}$$

となる．しかし，実際の性能比は，メモリアクセスや I/O といったプログラムの動作特性に大きく左右され，また，その特性はタスク単位で異なる可能性が高い．性能予測における誤差は，結果として性能の過小評価による消費電力の浪費や，性能の過大評価によるデッドラインミス率の増加を招く．このため，タスクの動作特性を考慮せず，全タスクに対して同一の予測を行う式 (7) を用いた性能予測手法では，省電力性およびリアルタイム性の両観点

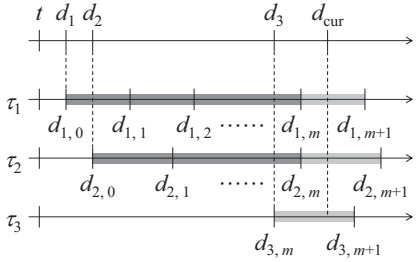

図 2 $\tau_i \in T'_{\mathrm{W}}$ の実行保証区間
Fig. 2 Interval where execution of $\tau_i \in T'_{\mathrm{W}}$ must be guaranteed

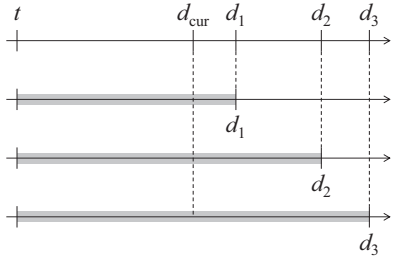

図 3 $\tau_i \in T_{\mathrm{R}}$ の実行保証区間
Fig. 3 Interval where execution of $\tau_i \in T_{\mathrm{R}}$ must be guaranteed

から不十分であるといえる．

非リアルタイム OS を対象とした DVFS 制御に関する筆者の過去の研究成果[7] では，タスク単位の動作特性に基づく性能予測を行うことにより，従来予測に比べて消費電力の削減を実現している．本論文の提案手法でも，同様の性能予測手法を用いる．本手法では，タスク τ_i の動作特性情報 x_i を説明変数とする線形回帰式

$$y(f, \tau_i) = b_0(f) + b_1(f) \cdot x_i \quad (8)$$

を性能予測モデルとして用いる．ここで，キャッシュミス回数がタスクの実行時性能に支配的であることに着目し，x_i には 1 命令あたりのキャッシュミス率を用いる．回帰係数 b_0 および b_1 は，事前に回帰分析によって周波数状態 f ごとに取得する．

DVFS 機能をもつプロセッサの中には，CPU 周波数を単独で制御できるものと，CPU 周波数とバス周波数が独立でないものが存在する．前者の場合は，単純に CPU 周波数 f_{cpu} のみを式 (8) の周波数状態 f として適用する．後者の場合，周波数状態 f は CPU 周波数 f_{cpu} とバス周波数 f_{bus} の組からなり，f_{cpu} と f_{bus} の組合せごとに，式 (8) の回帰係数 b_0，b_1 を決定する必要がある．ここで，b_0 はキャッシュミス率が 0 となる際の性能比であるため，f_{bus} には依存しないと考えられる．実際に，過去の成果[7] では，b_0 は CPU 周波数比とほぼ同等の値となることがわかっている．一方，b_1 はキャッシュミス率が性能にどの程度影響を与えるかを示すパラメータであるため，f_{cpu} と f_{bus} の両方に依存する．以上より，式 (8) は f_{cpu} と f_{bus} を用いると下記のように表せる．

$$y(f, \tau_i) = b_0(f_{\mathrm{cpu}}) + b_1(f) \cdot x_i \quad (f = f_{\mathrm{cpu}} \text{または } f = (f_{\mathrm{cpu}}, f_{\mathrm{bus}})) \quad (9)$$

4. 実　　装

4.1 RTAI を用いた実環境におけるシステム実装

Linux のリアルタイム拡張の一つである RTAI（Real-Time Application Interface）[8] を用いて提案手法に従って DVFS 制御を行うリアルタイムスケジューラを実装した．RTAI は，Linux カーネルにパッチを適用し，そのカーネルモジュールとして HAL やリアルタイムスケジューラなどの機構を導入することで Linux をリアルタイム拡張したもので，カーネルレベルのリアルタイムタスクや Linux プロセスのリアルタイム化を実現している（**図 4**）．今回，RTAI のスケジューラモジュールに 3 章で述べた DVFS 制御機構を実装し，各リアルタイムタスクに対する省電力化スケジューリングを実現した．また，RTAI の API にスケジューラ制御用のインタフェースを追加した．

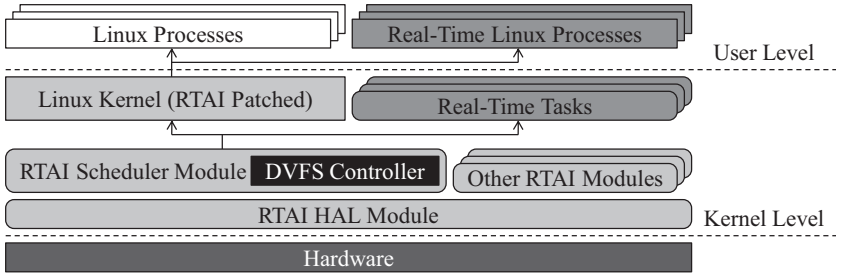

図 4　RTAI Linux の構成
Fig. 4　Structure of RTAI Linux

4.2 エネルギーシミュレータ

実機上で，各周波数・電圧状態における特定タスクの性能比と平均消費電力のデータを事前に取得し，この情報に基づいて，EDF スケジューリングを行ったときの合計エネルギー消費量とデッドラインミス率を計算するシミュレータを実装した．また，本シミュレータ上のスケジューラに 3 章で述べた DVFS 制御機構を実装し，入力タスクセットに対する提案手法の評価環境を構築した．

本シミュレータは，タスクセット情報として，各タスク τ_i の最悪実行時間 w_i，周期 p_i，動作特性を入力とし，このタスクセットに対する EDF スケジューリングをシミュレーションする．このとき，DVFS 制御時のタスク性能や実行時の平均電力を，事前取得したデータに基づいてタスクごとに計算する．最終的に，各タスクの実行時間とその消費電力の結果から，スケジューリング期間内の合計エネルギー消費量とデッドラインミス率を出力する．

4.3 使用プロセッサ

今回，DVFS 機能を搭載したプロセッサとして，Intel Pentium M 760 および Intel XScale PXA255 の二つをシステムの実装および評価に用いた．Pentium M では 0.8〜2.0[GHz] の計 9 段階，XScale では，99〜398[MHz] の計 7 段階の周波数・電圧状態を用いた．その各々で利用可能な最小電圧の組からなる計 9 段階の動作レベルを用いた．XScale では，DVFS 制御において，CPU 周波数だけでなくバス周波数も 50〜196[MHz] の範囲で変化する．また，両 CPU はパフォーマンスカウンタを備えており，実験ではこれを用いてタスク実行時の実行命令数とキャッシュミス率の取得を行った．なお，今回は RTAI を用いたシステムの実装・評価は Pentium M のみについて行い，XScale に関する評価はシミュレーションのみとした．

Vol.2010-OS-113 No.15
2010/1/28

表 1 評価用タスクセット
Table 1 Task sets

タスクセット名	タスク数	w_i[ms]	p_i[ms]
Experimental(n)	n	p_i/n	$10 \cdot i\,(i = 1, 2, \cdots, n)$
VideoPhone	4	1.8〜50.4	40.0〜66.7
AperiodicServer	10	0.1〜2.9	5.5〜38.5
ObjectRecognition	5	56.6〜121.2	141.5〜765.0

5. 評　　価

提案手法の他に，関連研究[1] における STATIC，LA の各アルゴリズムを実装し，**表 1** に示すタスクセットを用いて評価を行った．ここで，VideoPhone は典型的なテレビ電話アプリケーション[9]，AperiodicServer は非周期サーバのリアルタイムタスク情報をそれぞれ模擬したものである．Experimental を含めたこれらの三つのタスクの実体は，基本的な整数演算や条件分岐，メモリアクセスを繰り返す単純な内容になっている．ObjectRecognition は，入力された複数の画像に対して，タスクごとに異なる物体認識処理を行うアプリケーションである．

5.1 オーバヘッド評価

表 1 のタスクセットを RTAI 上で実行したときの，DVFS 制御に要する実行サイクル数を測定し，従来手法の中でも特に省電力効果の高い LA との比較を行った．特に，ObjectRecognition 実行時に，LA に比べ最大 43.4% のオーバヘッド削減を実現した．また，Experimental のタスク数 n を $1 \leq n \leq 10$ の範囲で変化させて測定を行ったところ，提案手法の LA に対する削減率は，$n = 5$ で 23.4%，$n = 7$ で 27.0%，$n = 10$ で 34.9% となり，n が大きいほどオーバヘッドを削減できることがわかった．

オーバヘッド削減の主な要因としては，LA では計算量 $O(n)$ であるのに対し，提案手法では $O(n_{\mathrm{R}} + n_{\mathrm{W}'}) \leq n$ であることや，提案手法ではコンテキストスイッチ時にのみ DVFS 制御を行うため，合計処理回数が少ないことなどが挙げられる．DVFS 制御のオーバヘッド増加は，OS 処理による CPU 時間の浪費となるため，これを削減することはリアルタイム性の確保だけでなく，システム全体の省電力化にも寄与する．特に，各タスク周期が短いタスクセットほど，その効果が顕著に現れると考えられる．

5.2 Intel Pentium M における CPU エネルギー評価

まず，Pentium M 上で行列演算やハッシュ計算などの複数のベンチマークプログラムを実行し，3.4 節で述べた性能予測モデル (9) の係数を回帰分析により決定した．次に，RTAI 上に実装した STATIC，LA，提案手法の各 DVFS 制御アルゴリズムのもとで，表 1 に示すタスクセットを実行したときの CPU エネルギー消費率 E_{rate} を測定した．主な結果を**図 5** に示す．E_{rate} の値は，DVFS 制御を行わなかった場合のエネルギー消費量を 100%として示してある．なお，全ケースでデッドラインミスは発生していない．実験では，タスクの実際の実行時間 c_i^{act} に変化を与えるため，平均値が $(b_i + w_i)/2$ の一様分布に従ってランダムに変化するよう c_i^{act} の値を決定した．ここで，b_i をタスク τ_i の最良実行時間（BCET：

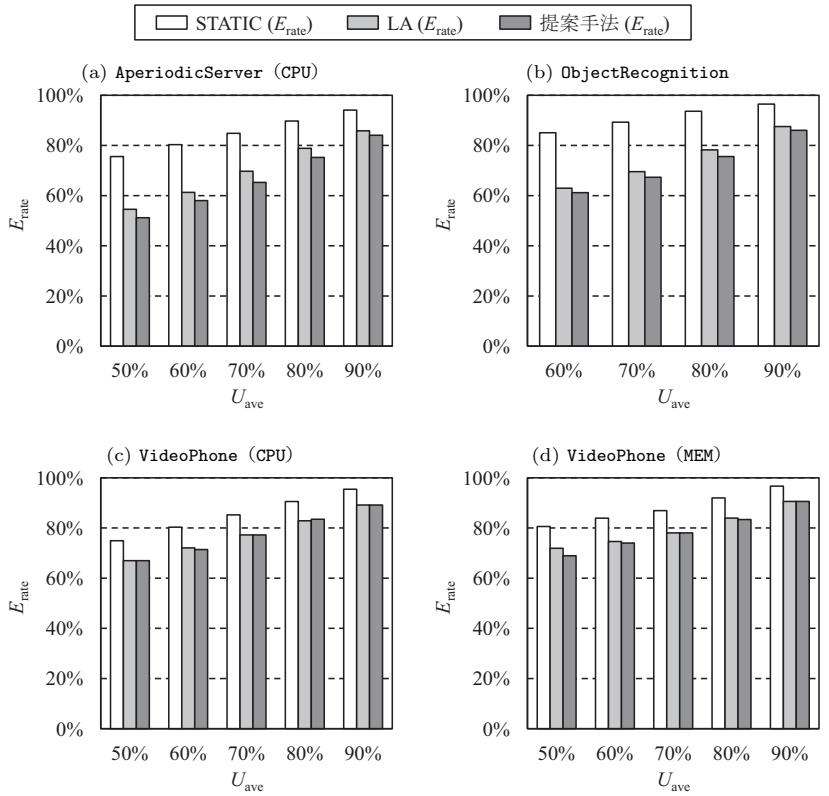

図 5 Pentium M における CPU エネルギー消費率 E_{rate}
Fig. 5 CPU energy ratio E_{rate} on Pentium M

Best-Case Execution Time）とする．図 5 の横軸 U_{ave} は実際の合計 CPU 使用率の平均値であり，b_i の値を調整することで決定している．また，`ObjectRecognition` 以外のタスクセットについては，処理内容が CPU バウンド（`CPU`）およびメモリバウンド（`MEM`）な場合の各々に対して評価を行った．

(a) や (b) では，すべてのケースで従来手法よりも低エネルギーとなっている．特に，省電力効果の高い `LA` と比較して，常に低オーバヘッドを維持しつつ最大 4.4% のエネルギー削減となった．(a) は CPU バウンドなタスクセットであり，性能予測による電力効率の差は生じないため，実行可能時間を従来手法よりも多く見積もれたことが省電力化の要因と考えられる．

(c) では，提案手法は `LA` とほぼ同等のエネルギー消費となった．これに対し，同一のタスク情報を持つメモリバウンドなタスクセットである (d) では，$U_{\mathrm{ave}} = 50\%$ において対 `LA` で 3% 程度エネルギーを削減している．`VideoPhone` は，タスク周期が 2 種類のみであり，加えて，ある 1 タスクの実行時間のみが極めて長いタスクセットとなっている．このため，一度決定された動作周波数で長時間タスク実行が持続する可能性が高く，結果的に 1 回の DVFS 制御における性能予測の重要度が高くなる．提案手法の性能予測では，キャッシュミス率の高いタスクほど従来予測よりも正確に性能を予測できるため，メモリバウンドな (d) において他の手法より周波数を低下でき，結果としてエネルギーを削減できたと考えられる．

5.3 Intel XScale における CPU エネルギーとデッドラインミス率の評価

Pentium M のときと同様に，XScale 上でも回帰分析により性能予測モデル (9) を構築し，これを用いて 5.2 節と同様の実験をシミュレータ上で行った．CPU エネルギー消費率 E_{rate} およびデッドラインミス率 D_{rate} の結果を**図 6** に示す．

(a) では，$U_{\mathrm{ave}} = 90\%$ 以外のすべてのケースで従来手法よりもエネルギーを削減できており，`LA` と比較した場合に最大 13.4% のエネルギー削減となっている．これは，図 5 の (d) と同様，性能予測の相違によるところが大きい．メモリバウンドなタスクの実行時性能は，CPU 周波数 f_{cpu} よりもバス周波数 f_{bus} に大きく依存するため，f_{cpu} が低くとも f_{bus} がより高い周波数状態を選択した方がエネルギー効率が良くなる場合がある．このため，f_{cpu} のみを用いて性能を予測する従来手法では，エネルギー効率が悪い周波数状態を選択する可能性が高い．提案手法では，キャッシュミス率を用いることでメモリバウンドタスクの性能をより正確に予測できるため，従来手法よりエネルギー効率の良い周波数を選択できる．

また，(a) の $U_{\mathrm{ave}} = 90\%$ では `STATIC` が最も低エネルギーとなっているが，デッドラインミス率が約 15% と大きく，リアルタイム性に問題がある．今回の評価環境では，デッドラインミスを起こしたタスクは即座に実行を中断して次の周期を開始する．このため，`STATIC` のエネルギー値は放棄された残りの処理分のエネルギーが除外されたものであり，省電力効果が高いことを示す結果とはいえない．

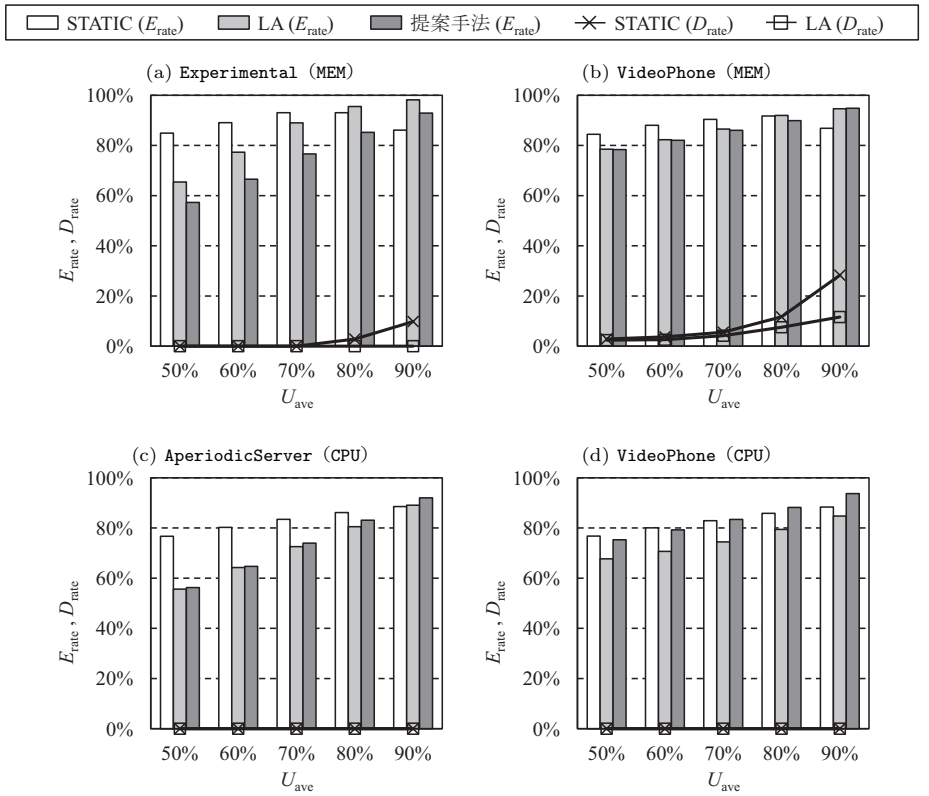

図 6 XScale における CPU エネルギー消費率 E_{rate} とデッドラインミス率 D_{rate}
Fig. 6 CPU energy ratio E_{rate} and deadline miss ratio D_{rate} on XScale

(b) では，すべてのケースで従来手法がデッドラインミスを起こしているのに対し，提案手法ではデッドラインミスを起こさず，かつ従来手法と同等のエネルギーを維持している．U_{ave} が大きいときほどその優位性は高く，対 `LA` で最大 11.6% のデッドラインミス率削減となっている．`VideoPhone` では 1 回の性能予測の重要度が高いことを述べたが，この性能

予測において，従来手法では性能を過大評価することが多いのに対し，提案手法では正確な予測を行えることが，本結果の要因である．前述のように，XScale においてはメモリバウンドなタスクの性能は f_{cpu} よりも f_{bus} に大きく依存するため，f_{cpu} が同じであれば f_{bus} が低い周波数状態の方がタスク性能は劣化する．従来手法では，f_{cpu} が同じならば f_{bus} がより低い周波数状態を選択するため，メモリバウンドタスクの性能を本来よりも過大評価する可能性が高く，結果としてデッドラインミス率が上昇したと考えることができる．

以上の結果より，省電力性とリアルタイム性の両観点から，メモリバウンドなタスクに対してキャッシュミス率を考慮した性能予測を行うことの重要性が理解できる．これは，XScale のように，DVFS 制御においてバス周波数も変動するようなアーキテクチャでは特に重要であるといえる．

図 6 の (c) と (d) は，いずれも CPU バウンドなタスクセットに対する評価結果である．(c) では，従来手法である `LA` と比較してほぼ同等のエネルギー消費となっているが，U_{ave} が大きいときにややエネルギー効率が悪くなっている（最大 2.9% 増）．また，(d) では常に `LA` よりもエネルギー消費が高くなる結果となった（最大 8.9% 増）．CPU バウンドタスクに対しては，従来手法との間で性能予測結果に差異は生じないため，実行可能時間の見積もり精度に問題があるといえる．3.3 節に記したように，提案手法では，カレントタスクのデッドラインをまたぐような周期区間を持つタスクについては保証時間の厳密な計算を行わないため，このようなタスクが超周期内に多く存在するようなタスクセットに対しては，提案手法の省電力効果が低くなる可能性がある．`VideoPhone` では，タスク実行中の周波数変更の機会が `LA` に比べて少ないため，必要以上に高い周波数で長時間動作した結果，エネルギー効率が悪くなったと考えられる．この問題を解決するためには，実行可能時間をより効率的に見積もるアルゴリズム設計が求められる．

6. 結　　言

本論文では，周期タスクのみを扱う EDF スケジューリングを対象に，OS スケジューラによるタスク単位の DVFS 制御手法を提案した．提案手法では，タスクごとの動作情報に基づいて DVFS 制御時の性能予測を行うことで，特にメモリバウンドなタスクセットに対して，従来手法よりも高い省電力性とリアルタイム性を実現した．具体的には，特に省電力性に優れる従来手法と比較して，最大で 13.4% の CPU エネルギー削減と，11.6% のデッドラインミス率削減を実現した．

今後の研究課題としては，まず，CPU バウンドなタスクセットに対する省電力性向上のための，5.3 節で述べた実行可能時間の見積もり手法の改善が挙げられる．次に，XScale を用いた実システム上での評価や，性能予測モデルの改善などが挙げられる．現時点では，性能予測モデルは線形単回帰として実現し，説明変数にキャッシュミス率を用いているが，これ以外のモニタリング情報を含めた重回帰への発展や，予測精度改善のための非線形式としての定義などが考えられる．この他，タスクの I/O 処理も含めた DVFS 制御アルゴリズムの設計なども，今後検討すべき課題である．

参 考 文 献

1) Pillai, P. and Shin, K. G.: Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems, *Proc. 18th ACM Symposium on Operating Systems Principles*, pp.89–102 (2001).
2) Poellabauer, C., Singleton, L. and Schwan, K.: Feedback-Based Dynamic Voltage and Frequency Scaling for Memory-Bound Real-Time Applications, *Proc. 11th IEEE Real Time and Embedded Technology and Applications Symposium*, pp.234–243 (2005).
3) Kim, W., Kim, J. and Min, S. L.: A Dynamic Voltage Scaling Algorithm for Dynamic-Priority Hard Real-Time Systems Using Slack Time Analysis, *Proc. Design Automation and Test in Europe*, pp.788–794 (2002).
4) Zhu, Y. and Mueller, F.: Feedback EDF Scheduling Exploiting Dynamic Voltage Scaling, *Proc. 10th IEEE Real-Time and Embedded Technology and Applications Symposium*, pp.84–93 (2004).
5) Moncusi, M., Arenas, A. and Labarta, J.: Moving Average Frequency Reduction for Low Power in Hard Real-Time Systems, *Proc. 2nd International Workshop on Power-Aware Real-Time Computing (PARC)* (2005).
6) 池田雄児，加藤真平，山崎信行：固定優先度スケジューリング向け動的電圧周波数制御，先進的計算基盤システムシンポジウム（SACSIS2009），pp.407–414 (2009).
7) 林　和宏，金井　遵，丸山勝巳，並木美太郎：L4 マイクロカーネルにおける省電力スケジューラの開発，情報処理学会論文誌， Vol.2, No.1(ACS25), pp.96–109 (2009).
8) RTAI Team: RTAI - Official Website, https://www.rtai.org/.
9) Shin, D., Kim, J. and Lee, S.: Intra-Task Voltage Scheduling for Low-Energy, Hard Real-Time Applications, *IEEE Design and Test of Computers*, Vol. 18, No. 2, pp. 20–30 (2001).