Vol.2014-SLDM-168 No.4
2014/11/26

同一命令セットヘテロジニアスマルチコアに適したリアルタイムシステム向けタスクマイグレーション手法

岩田 淳[†1,a)]　高瀬 英希[†1]　高木 一義[†1]　高木 直史[†1]

概要：近年，組込みシステムのアーキテクチャとして，同じ命令セットで異なる性能を持つ，同一命令セットヘテロジニアスマルチコアが注目されている．このアーキテクチャにおいてデッドライン制約を保証した上で消費エネルギーを削減するためには，適切なタスクスケジューリングや動的電圧周波数制御（DVFS），更には動作コアの切り替えを行う必要がある．本研究は，同一命令セットヘテロジニアスマルチコアに適した，消費エネルギー削減のためのタスクマイグレーション手法を提案する．タスクマイグレーションの実行には，組込みシステム向け DVFS アルゴリズムを利用して判断する．タスクマイグレーションによって，コア間の負荷を平準化しながら高電力効率コアのみでの動作を実現する．実験の結果，DVFS のみ適用する場合と比較して，提案手法は最大約 10.4%の消費エネルギー削減効果があることが示された．

キーワード：組込みシステム，タスクマイグレーション，ヘテロジニアスマルチコア，消費エネルギー削減

A Task Migration Method for Real-time Systems on Heterogeneous Multi-Cores with Single Instruction Set Architecture

Iwata Atsushi[†1,a)]　Takase Hideki[†1]　Takagi Kazuyoshi[†1]　Takagi Naofumi[†1]

Abstract: Recently, single instruction set heterogeneous multi-core architecture have focused attention for use in embedded systems. In order to reduce energy consumption on this architecture, while guaranteeing the deadline constraints of embedded real-time systems, appropriate task scheduling, including task allocation, dynamic voltage frequency scaling (DVFS) and selecting active cores, is necessary. In this paper, we propose a task migration method for real-time systems on single instruction set heterogeneous multi-core architecture. In the proposed method, DVFS algorithms for real-times systems are utilized for deciding whether tasks should be eventually migrated or not. The task migration method performs load balancing between cores and preferential use of energy efficient cores simultaneously, which leads to the reduction of energy consumption. We evaluated the amount of energy savings of the proposed method by simulating a task scheduling process. The results show that our method can save about 10.4% energy consumption in the best case compared to the existing DVFS method.

Keywords: Embedded Systems, Task Migration, Heterogeneous Multi-Cores, Energy Reduction

1. はじめに

家電製品や機械に組み込まれているコンピュータのことを組込みシステムという．多くの組込みシステムは，高いリアルタイム性を要求され，各タスクの実行を，それぞれ所定の時刻までに完了する必要があるというデッドライン制約を保証する必要がある．自動車や航空機を制御するシステムが，そのようなリアルタイム性が要求される組込みシステムの一例である．近年の組込みシステムでは，より高い性能が求められる一方，その消費エネルギーを削減することが重要な課題となっている．

組込みシステムの消費エネルギー削減の有効な技術の 1 つに，動的電圧・周波数制御（Dynamic Voltage and Fre-

[†1] 現在，京都大学
Presently with Kyoto University
[a)] a.iwata@lab3.kuis.kyoto-u.ac.jp

quency Scaling, DVFS）がある．DVFS は，CPU の供給電圧および動作周波数を適切に制御してタスクを実行する技術である．CMOS 回路の消費エネルギーは，動作周波数の 2 乗に比例すると近似できる [1]．ゆえに，DVFS によってコアの供給電圧と動作周波数を下げてタスクを実行することで，消費エネルギーの削減が可能となる．ただし，組込みシステムに DVFS を適用する際には，デッドライン制約を保証する周波数に設定しなければならない．

近年，高性能かつ低消費エネルギーを実現するアーキテクチャとして，ヘテロジニアスマルチコアが注目されている．ヘテロジニアスマルチコアとは，異なる性能をもつ複数のコアをもつアーキテクチャのことである．ARM 社が考案した big.LITTLE アーキテクチャ [2] は，同一命令セットをもつヘテロジニアスマルチコアの一例である．動作コアの適切な切り替えにより，高性能のコアのみ使用した場合と比較して，性能を落とすこと無く消費エネルギーを削減することができる [3] 今後，big.LITTLE アーキテクチャのような同一命令セットヘテロジニアスマルチコアを採用した組込みシステムの普及が見込まれ，高性能と低消費エネルギーの両立に大きく貢献することが期待される．

マルチコアの組込みシステムでは，デッドライン制約の保証のため，動作前にタスクのコアへの配置を固定してコア毎にスケジューリングする手法が一般的である．この場合，コア毎にシングルプロセッサ向け DVFS を適用できる．しかし，このスケジューリングは，動的な負荷の変動に対応できない欠点がある．DVFS は，コア間の負荷が平準化されている場合，消費エネルギー削減効果が最も高くなる [4]．文献 [5] では，マルチコア上で DVFS の消費エネルギー削減を高めるためのタスクマイグレーション手法が提案されている．しかし，同一命令セットヘテロジニアスマルチコアのアーキテクチャに適した手法はまだ提案されていない．

本研究では，同一命令セットヘテロジニアスマルチコアに適した消費エネルギー削減のためのタスクマイグレーション手法を提案する．タスクマイグレーションの実行判定には，組込みシステム向け DVFS アルゴリズムを利用する．タスクマイグレーションによって，コア間の負荷平準化と高電力効率コアでの動作を同時に実現することで，消費エネルギー削減をする．

2. 同一命令セットヘテロジニアスマルチコア

同一命令セットヘテロジニアスマルチコアとは，同じ命令セットを持ち，性能が異なる複数のプロセッサコアを持つアーキテクチャのことである．適切なタスクスケジューリングにより，高い性能対電力効率を実現することができる [3]．ARM 社が考案した big.LITTLE アーキテクチャ [2] は，性能が高いが消費電力も大きい big コアと比較的性能は低いが性能対電力効率が高い LITTLE コアを持つ同一命令セットヘテロジニアスマルチコアである．同一命令セットヘテロジニアスマルチコアでは，タスクマイグレーションと CPU マイグレーションを適用できる．

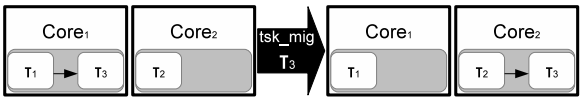

図 1 タスクマイグレーションの例

2.1 タスク割り付けとタスクマイグレーション

マルチコアの組込みシステムでは，システムの動作前に各コアにタスクが割り付けられる手法が一般的である．同一命令セットヘテロジニアスマルチコアでも，この割り付け手法が適用できる．タスクは，動作前に割り付けられたコアから移動することはないため，事前に動作が予測できる．そのため，リアルタイム性が要求される組込みシステムに適している．また，コア間にタスク依存関係が無い場合，消費エネルギーを削減するために，各々のコアで，シングルプロセッサコア向けの DVFS アルゴリズムを適用できる．しかし，この割り付け手法は，コア間の動的な負荷の変動に対応することができないという欠点がある．

あるタスクを，割り付けられたコアから別のコアに割り付けを移動させることをタスクマイグレーションという．図 1 は，$Core_1$ に割り付けられていたタスク τ_3 を，$Core_2$ にタスクマイグレーションしている例である．組込みシステムにおいて，動作時の負荷を平準化するために，タスクマイグレーションを実行することで，DVFS の効果を高めることができる．その場合，デッドライン制約を保証するため，適切なタスクマイグレーション手法が必要となる．

2.2 CPU マイグレーション

LITTLE コアおよび big コアのそれぞれ 1 個のコアを対応付けてコアペアとして，コアを排他的に動作させることを考える．つまり，一方のコアが動作している時，他方のコアはスリープ状態となる．その場合，コアペア内の一方のコアに割り付けられている全てのタスクを，他方のコアにタスクマイグレーションすることを CPU マイグレーションという．負荷が大きい場合は big コアを，そうでない場合は LITTLE コアを動作コアとして選択することにより，高い性能対電力効率を実現できる．図 2 は，CPU マイグレーションの例である．LITTLE コアと big コアの添字が同じコアを合わせてコアペアとして扱い，$Core_1$ が，LITTLE コアから big コアに CPU マイグレーションしていることを表している．

3. 提案するタスクマイグレーション手法

3.1 要件と方針

big コアと LITTLE コアをコアペアとして扱い，コアペ

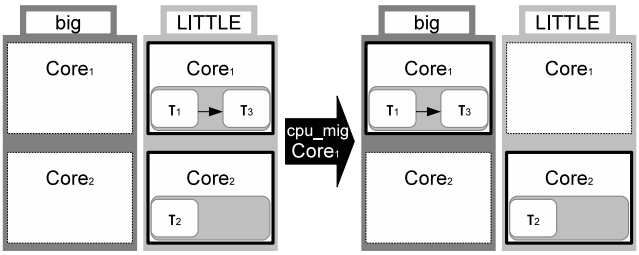

図 2 CPU マイグレーションの例

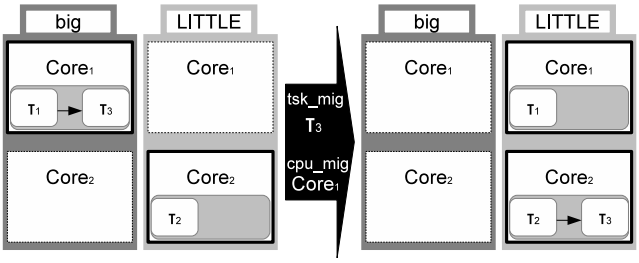

図 3 提案手法の適用例

ア毎に独立してタスクスケジューリングおよび DVFS を適用する．タスクの割り付けは，各コアペアに静的に行われるものとする

組込みシステム向けの DVFS アルゴリズムは，これまで数多く提案されている．既存のタスクマイグレーション手法 [5] では，特定の DVFS アルゴリズムに依存した手法である．本研究では，DVFS アルゴリズムに依存しない一般性のあるタスクマイグレーション手法を提案する．それにより，多くのシステムに対応可能な柔軟性をもつ．

タスクが初期のコアペア割り付けに固定される場合，動的な負荷の変動に対応できず，コアペア間の負荷に差が生じる．負荷の差が大きい場合，DVFS による消費エネルギー削減効果が小さくなる．そのため，コアペア間の負荷を平準化するためにタスクマイグレーションを実行する．しかし，頻繁にタスクマイグレーションを実行してしまうと，そのオーバヘッドが大きくなり，消費エネルギー削減効果が期待できないことが考えられる．

対象システムは，big.LITTLE アーキテクチャのように，big コアと LITTLE コアの性能隊電力効率が異なるアーキテクチャとする．本アーキテクチャでは，同じコアペア性能となる周波数の場合，big コアより，LITTLE コアを動作させた方が消費エネルギーを低く抑えることが出来る．

以上から，負荷の平準化によって big コアを使用せず LITTLE コアで動作可能な場合のみ，タスクマイグレーションを実行することを方針とする．図 3 は，この方針に従ったタスクマイグレーションの適用例である．コアペア $Core_1$ の動作コアが big コアの場合に，$Core_1$ に割り付けられているタスクを，他方のコアペア $Core_2$ に対しタスクマイグレーションすることを考える．この例では，τ_3 をタスクマイグレーションすることで，$Core_1$ 及び $Core_2$ が共に LITTLE コアで動作可能となることを表している．

3.2 DVFS の適用方法

DVFS の適用方法は，big.LITTLE アーキテクチャに対して Linaro が開発した In-Kernel Switcher[9] に類似したものである．big コアと LITTLE コアは，マイクロアーキテクチャの違いから 1 サイクル毎に実行される平均命令数（平均 IPC）が異なる．このコア間の平均 IPC 比と動作周波数を考慮することで，コアペア内の CPU マイグレーションは DVFS の枠組みで考えることができる．予め big コアの最高周波数でのコアペア性能を 1 として，big コア及び LITTLE コアの各周波数設定値のコアペア性能を算出する．例えば，big コアの最高周波数が 1.0GHz，IPC が 2.0，LITTLE コアの IPC が 1.0 の場合，LITTLE コアの 500MHz のコアペア性能は $\frac{500}{1000} \times \frac{1.0}{2.0} = 0.25$ となる．LITTLE コアの最高周波数でのコアペア性能 $P_{LITTLE,max}$ とする．動作時，DVFS アルゴリズムで要求されるコアペア性能 P を導出した後，$P > P_{LITTLE,max}$ の場合は big コアを，$P \leq P_{LITTLE,max}$ の場合は LITTLE コアを動作コアとして選択し，DVFS を適用する．

3.3 タスクマイグレーションの実行判定

本節ではまず，簡単のため，コアペアが 2 個（big コアおよび LITTLE コアが各 2 個）のアーキテクチャを対象として説明する．図 4 および 5 は，それぞれ，提案タスクマイグレーション手法のタスクマイグレーション元のコアおよびタスクマイグレーション先のコアにおけるフローチャートである．図の太い矢印は，コアペア間の情報のやりとりを表している．

タスクマイグレーションの実行判定のタイミングは，オーバヘッドを考慮して，DVFS アルゴリズムでコアペア CP_1 に要求されるコアペア性能 P の導出する時のみとする．その際，以下の 3 個の条件を満たす場合にタスクマイグレーションの実行を判定する．

(1) $P > P_{LITTLE,max}$ となる
(2) タスクマイグレーション後に，要求されるコアペア性能 P' が $P_{LITTLE,max}$ 以下になる可能性がある
(3) 他方のコアペアの CP_2 が実行中のタスクがない状態または LITTLE コアで動作中である

タスクマイグレーションの実行判定の流れを説明する．まず，CP_1 に割り付けられたタスクのうち，低優先度のものから順番にあるタスク τ を除外した場合に CP_1 に要求されるコアペア性能 P' を，DVFS アルゴリズムによって導出する．$P' \leq P_{LITTLE,max}$ である場合，次に τ を CP_2 に追加する場合の CP_2 に要求されるコアペア性能 P'' を DVFS アルゴリズムによって導出する．$P'' \leq P_{LITTLE,max}$ である場合，タスクマイグレーションを実行する．その後，CP_1 および CP_2 で DVFS を実行し，通常のタスク実行に戻る．

CP_1 のどのタスク 1 個を削除しても $P' \leq P_{LITTLE,max}$ を満たさない場合，タスクマイグレーションは実行しない．

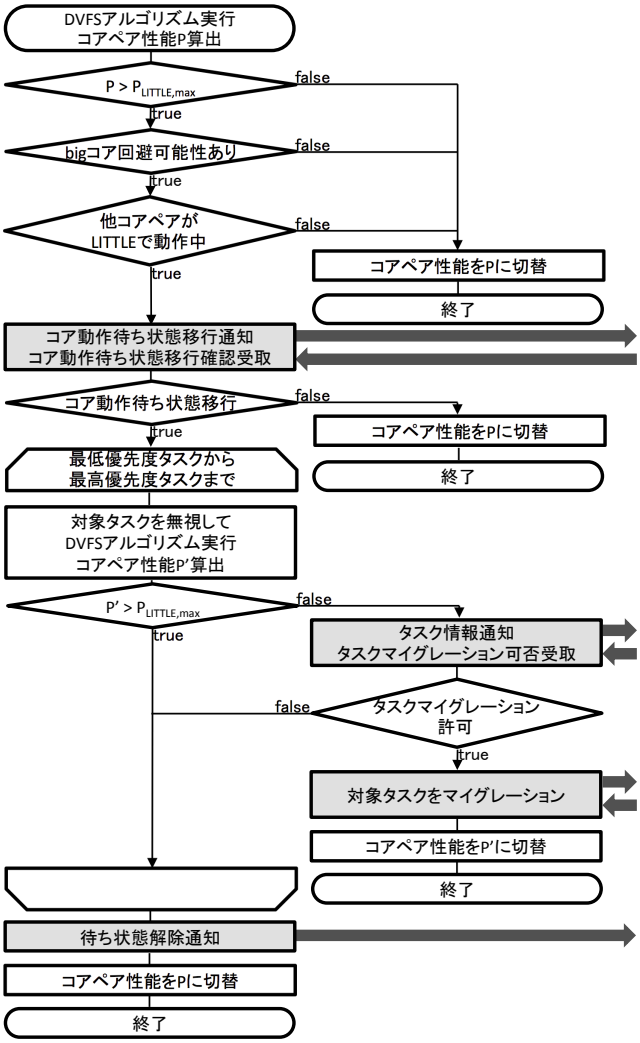

図 4 タスクマイグレーション元コアペア CP_1 のフローチャート

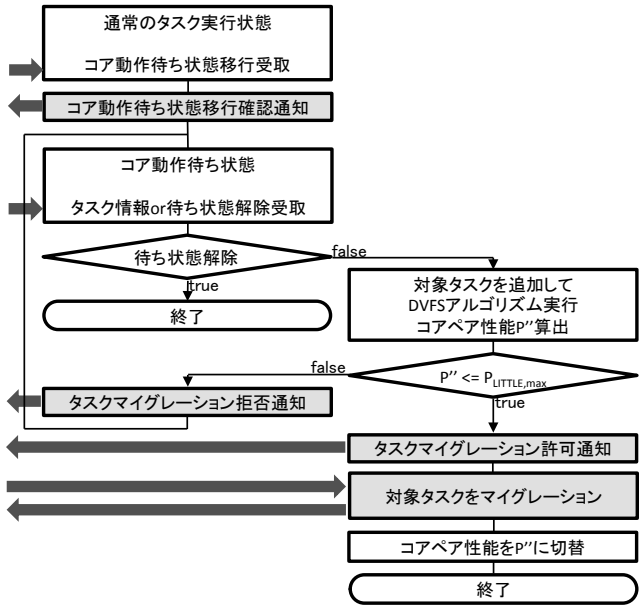

図 5 タスクマイグレーション先コアペア CP_2 のフローチャート

また，そのようなタスクが存在しても $P'' \leq P_{LITTLE,max}$ を満たさない場合も，タスクマイグレーションは実行しない．これらの場合，CP_1 で DVFS が実行され，通常のタスク実行に戻る．

タスクマイグレーションの実行判定には，組込みシステム向け DVFS アルゴリズムを利用する．提案手法において，$P'' \leq P_{LITTLE,max}$ がタスクマイグレーションの実行条件であるので，CP_2 はタスクマイグレーション後も LITTLE コアで動作する．これは，少なくとも，DVFS アルゴリズムで要求される CP_2 のコアペア性能が big コアの最高周波数を超える場合，タスクマイグレーションは実行されないことを意味する．よって，提案タスクマイグレーション手法を用いても，対象システムのデッドライン制約は保証される．

3.4 コアペアが 3 個以上のアーキテクチャへの対応

3.3 節では，タスクマイグレーション先の候補となるコアペアが複数ある場合を想定していない．例えば，コアペアが 3 個のアーキテクチャにおいて，1 個が big コア，その他 2 個が LITTLE コアで動作する場合，タスクマイグレーション先の候補となるコアペアは 2 個となる．

コアペアが 3 個以上のアーキテクチャに対応するために，ここで，タスクマイグレーション先のコアペアに関して，2 通りの方法が考えられる．

(A) DVFS アルゴリズムで要求されたコアペア性能が最も低いコアペアのみを対象とする

(B) DVFS アルゴリズムで要求されたコアペア性能が低い順に，全コアペアを対象とする

(A) は，コアペアの中で，要求されたコアペア性能が最も低いものが，タスクマイグレーション実行できる可能性が高いことを利用している．(B) は，全てのコアペアが対象となるので，(A) で調査されないコアペアへのタスクマイグレーションが実行できる場合がある．タスクマイグレーションの実行判定にかかるオーバヘッドは，全コアペアを対象とする (B) が，(A) に比べて大きくなると考えられる．

4. 評価

4.1 実験環境

提案タスクマイグレーション手法の消費エネルギー削減効果を評価するために自作のタスクスケジューリングシミュレータで実験を行った．

まず，big コアおよび LITTLE コアはそれぞれ 2 個，つまりコアペアが 2 個での実験を行った．各コアの平均 IPC，設定できる動作周波数，および，それに対応する消費電力のパラメータは，big.LITTLE アーキテクチャのホワイトペーパー [2] の値を参考にした．ホワイトペーパーのプロットに従い，動作周波数は，3 段階の離散値に設定できるものとした．また，スリープ状態のコアは電力を消費しないものとした．各コアは，それぞれ独立の動作周波数に設定できるものとした．

入力するタスクセットは，最悪実行時間での CPU 使用率 *utilization* と最悪実行時間と実際の実行時間の比 *aet_ratio* を設定して自動生成した．各タスクは起動周期，最悪実行時間および実際の実行時間配列を持つものとした．タスクの総数は 20 個とし，各タスクの起動周期は 2ms 間隔の 2–100ms の一様分布とし，全タスクの周期の最小公倍数であるハイパーピリオドが 10s 以下となるものを選んだ．最悪実行時間および実際実行時間配列は，*utilization* および *aet_ratio* と正規分布を利用して決定した．

実験の対象としたタスクスケジューリングについて説明する．実験したタスク割り付けの種類は，各タスクを各々の負荷（最悪実行時間/起動周期）で降順ソートしたものをワーストフィットする Worst Fit Decreasing (WFD)[4] と，各コアの最悪実行時間での CPU 使用率が 1 以下となるようなランダムなものの 2 つを選んだ．DVFS を適用する際，WFD は電力対性能効率の高い割り付けである [4]．

各割り付けに対して，DVFS アルゴリズムを実行するスケジューリングと，更に提案タスクマイグレーション手法を加えたスケジューリングを実験した．タスクマイグレーションの実行および DVFS にかかる時間および消費エネルギーのオーバヘッドは無視した．

DVFS アルゴリズムには，laEDF[6] を利用した．laEDF は，Earliest Deadline First（EDF）スケジューリング方式 [7] に適用でき，高い消費エネルギー削減効果がある [8]．laEDF は，各タスクのリリース及びディスパッチ時に周波数を算出する．提案タスクマイグレーション手法では，タスクマイグレーション実行判定に，タスクの負荷を除外した場合や，負荷を加えた場合の周波数を算出する必要がある．laEDF は，各タスクの起動周期，最悪実行時間および残り最悪実行時間のみから周波数を算出できるため，提案タスクマイグレーション手法に容易に適用することができる．

ハイパーピリオドまでタスクスケジューリングをシミュレーションし，それまでの消費エネルギーを算出する．その値は，big の最高周波数で動作する場合の消費エネルギーで正規化したものとする．実験では，同じ *utilization* と *aet_ratio* で自動生成したタスクセットを 100 個生成し，それらで算出された消費エネルギーの平均を実験結果とした．*utilization* には 1.0，1.2，1.4，1.6 及び 1.8 を，*aet_ratio* には 0.2，0.4，0.6，0.8 及び 1.0 を入力パラメータとして，それらの合計 25 通りの組合せを実験した．

次に，3.4 節の 2 通りの方法を評価するため，コアペアが 2，4，6 および 8 個の場合の実験を行った．*utilization* およびタスクの総数は，コアペアの個数に比例させ，各コアペアに同等の負荷がかかる設定とした．

4.2 実験結果

紙面の都合上，提案タスクマイグレーション手法の効果の特徴が表れている入力パラメータの結果のみを示す．

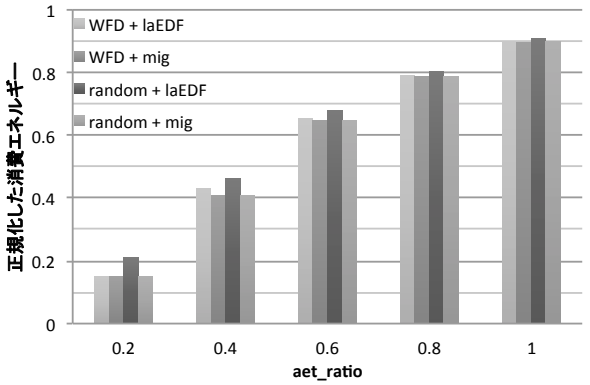

図 6 *utilization* が 1.2，タスクの総数が 20 個で，*aet_ratio* を変動させた場合の結果（コアペア 2 個）

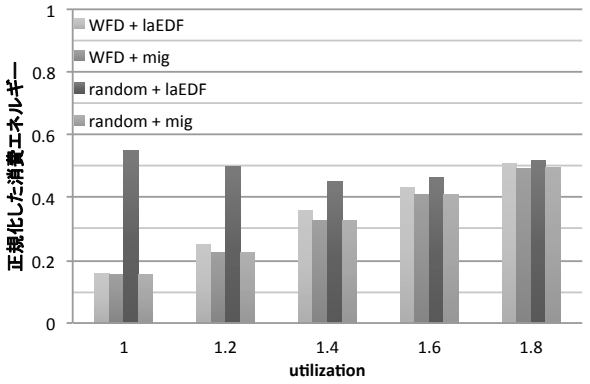

図 7 *aet_ratio* が 0.4，タスクの総数が 20 個で，*utilization* を変動させた場合の結果（コアペア 2 個）

まず，コアペアが 2 個の実験結果を示す．図 6 は，*utilization* を 1.2 とし，*aet_ratio* を変動させた場合の実験結果である．$utilization = 1.2$，$aet\_ratio = 0.4$ の時，提案タスクマイグレーション手法の消費エネルギーは，WFD 割り付けをして laEDF でスケジューリングしたものと比較して，約 10.4%の消費エネルギー削減効果があった．また，laEDF スケジューリングの場合，動作前のタスク割り付け方法によって，消費エネルギー削減効果に大きく差が出るのに対し，提案手法を用いたスケジューリングは，タスク割り付けに依らず，安定した消費エネルギー削減効果がある結果となった．

図 7 は，*aet_ratio* を 0.4 とし，*utilization* を変動させた場合の実験結果である．多くの場合，提案タスクマイグレーション手法の消費エネルギー削減効果が WFD 割り付けの laEDF より優れていることが分かった．しかし，*utilization* が 1 や 1.8 といった極端な値の場合，効果の差は小さくなった．また，図 6 と同じく，提案手法がタスク割り付けの種類に依らない消費エネルギー削減効果があった．

これ以外の入力パラメータの場合も，提案タスクマイグレーション手法による消費エネルギー削減効果は，WFD 割り付けをして laEDF でスケジューリングしたものと同

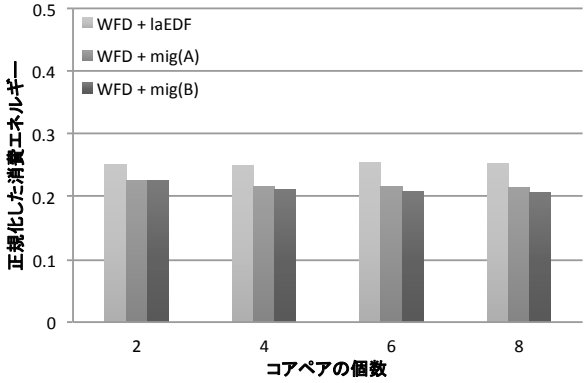

図 8 *utilization* が $0.6 \times N$，*aet_ratio* が 0.4，タスクの総数が $10 \times N$ で，N を変動させた場合の結果（コアペア N 個）

等か，それより優れていた．このことから，提案手法の有効性が示された．

次に，コアペアの個数 N を変動させた実験結果を示す．図 8 は，*utilization* を $0.6 \times N$，*aet_ratio* を 0.4，タスクの総数を $10 \times N$ とした場合の実験結果である．(A) および (B) のどちらの方法も，laEDF でスケジューリングしたものと比較して，消費エネルギーが削減された．(A) と (B) を比較すると，(B) が消費エネルギーをより削減しており，コアペア数が多くなるほどその傾向が強くなった．

4.3 考察

提案手法では，一部タスクセットの場合，WFD 割り付けの laEDF とほぼ同等の結果となった．そのようなタスクセットは，*utilization* と *aet_ratio* の値が共に小さい場合，または，大きい場合の二通りであった．前者の場合，タスクの負荷が非常に小さくなり，big コアが動作コアとなることが少なくなる．この結果，タスクマイグレーションを実行する機会が無くなり，提案手法が laEDF と同じスケジューリングとなることが理由である．後者の場合，各タスクの負荷が大きく，一度タスクマイグレーションを実行してしまうと，コア間の負荷の偏りが大きくなってしまう可能性がある．タスクセットの負荷が大きい場合，一時的な負荷の偏りのためにタスクマイグレーションを実行した後に，再度負荷の偏りがある際にタスクマイグレーション出来ないことが頻繁に生じると考えられる．その結果，負荷の偏りが大きい時間が，タスクマイグレーションをしない場合と比較して長くなり，消費エネルギー削減効果が減少してしまう．従って，提案手法は多くの場合で効果的であるが，タスクセットが LITTLE コアのみ使用する場合，または，各タスクの負荷が大きい場合のみ，DVFS アルゴリズムのものとほぼ同等の効果となる．

また，割り付け方法による消費エネルギー削減効果の差は，提案手法を用いると殆ど無い．これは，動作時のタスクマイグレーションによって，割り付け方法に関わらず負荷が平準化されているからだと考えられる．

コアペアが 3 個以上の場合の対応方法に関して，実験結果より，コアペアの個数が多くなるほど (B) の消費エネルギー削減効果が高くなった．しかし，(B) は，タスクマイグレーション先の候補として全コアペアを対象とするため，コアペアの個数が多くなると，タスクマイグレーションの実行判定にかかるオーバヘッドもより大きくなる．そのため，実環境に適用する際は，そのトレードオフを慎重に考えなければならない．

5. おわりに

本研究は，同一命令セットヘテロジニアスマルチコアに適した，消費エネルギー削減のためのタスクマイグレーション手法を提案した．提案手法を用い，DVFS の場合のみのもとと比較して，最大約 10.4%の消費エネルギー削減効果があることが示された．

今後は，タスクマイグレーションや DVFS で発生するオーバヘッドを考慮する方法を検討し，この手法を実用システム上で評価することを考えている．

謝辞 本研究の一部は JSPS 科研費 24300019 および 26870303 の助成による．また，本研究を進めるにあたり名古屋大学の松原豊助教，並びに，曾剛講師に貴重なご助言を頂いた．

参考文献

[1] Hu, S.X., et al.: Fundamental of Power-aware Scheduling, *Designing Embedded Processors: A Low Power Perspective*, Springer (2007).
[2] Greenhalgh, P.: Big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7, ARM White Paper (2011).
[3] Kumar, R., et al.: Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction,*Proc. of Int'l Sympo. on MICRO*, pp. 81–92 (2003).
[4] Aydin, H., et al.: Energy-Aware Partitioning for Multiprocessor Real-Time Systems, *Proc. of IPDPS* (2003).
[5] Seo, E., et al.: Energy Efficient Scheduling of Real-Time Tasks on Multicore Processors, *Proc. of IPDPS*, pp. 1540–1552 (2008).
[6] Pillai, P., et al.: Real-time dynamic voltage scaling for low-power embedded operating systems, *ACM SIGOPS Operating Systems Review*, Vol. 35, No. 5, pp.89–102 (2001).
[7] Liu, C. L., et al.: Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment, *Journal of the ACM*, Vol. 20, No. 1, pp. 46–61 (1973).
[8] Kim, W., et al.: Performance comparison of dynamic voltage scaling algorithms for hard real-time systems, *Proc. of RTCSA*, pp.219–228 (2002).
[9] In Kernel Switcher: A solution to support ARM's new big.LITTLE technology [online], `https://events.linuxfoundation.org/images/stories/slides/elc2013_poirier.pdf` (2013)