ヘテロジニアス計算機クラスタにおける省エネルギー化タスクスケジューリング手法

山下 良[†1] 近藤正章[†1]
平澤将一[†1] 本多弘樹[†1]

近年，データセンタの省エネルギー化への要求が高まっている．データセンタでは機器の更新が頻繁に行われるため，様々な計算機で構成されているヘテロジニアス構成であることが多い．そのため，同一タスクを処理するのに要する消費エネルギーはサーバ毎に異なり，スケジューリングによって，タスクセットの処理に必要な消費エネルギーも異なる．本稿では，先行制約を持つタスクセットを対象に，ヘテロジニアスなサーバ計算機環境を考慮した低消費エネルギー化タスクスケジューリング手法を提案する．提案手法は，従来の Heterogeneous Earliest Finish Time (HEFT) 法のスケジューリング結果を基に，プロセッサのアイドル時，またはスタンバイモード時の消費電力を考慮しつつ，サーバへのタスク再割り当てを行うことで，タスク処理のエネルギーを削減するものである．本提案手法を評価したところ，HEFT 法に比べ，タスクセットのスケジュール長を変えずに消費エネルギーを削減できることがわかった．

A Task Scheduling Method for Low-Energy Consumption on Heterogenius Cluster Systems

Ryo Yamashita,[†1] Masaaki Kondo,[†1]
Shoich Hirasawa[†1] and Hiroki Honda[†1]

Reducing energy consumption of data-centers is one of the important requirement for data-center operations. Since the hardware of server systems is replaced frequently, there is a heterogeneity in data-centers. Therefore, the energy consumption for processing a task depends on the server that the task is allocated. In this paper, we propose a task scheduling method to reduce energy consumption for processing a task set in which each task has dependency to other tasks. Our method is based on the Heterogeneous Earliest Finish Time (HEFT) scheduling algorithm. After HEFT scheduling, we re-allocate tasks to low-power servers without increasing the critical path length of the task set. We evaluate the proposed method and the evaluation results reveal that the proposed method successfully reduces energy consumption in most of the evaluated cases.

1. はじめに

近年，多数のサーバ計算機を保持するデータセンタでの電力需要の増大により，データセンタの省エネルギー化への要求が高まっている．米国では，データセンタにより消費される電力エネルギーが米国での全電力の 1.5%を占め，年率 10%の勢いで電力需要が増加していると言われており，米国の電力インフラ全体の問題ともなっている[1)]．データセンタで使用される電力のうち大きな割合を占めているものとして，データセンタに電力を供給する電力設備と，サーバ計算機自体で消費される電力がある．後者はデータセンタで使用される全電力のおよそ 4 割を占めているため[1)]，サーバ計算機の消費エネルギー削減のための研究は非常に重要である．

データセンタでは機器の更新が頻繁に行われるため，様々な計算機で構成されているヘテロジニアス構成であることが多い．サーバ計算機の処理性能と消費電力がヘテロジニアスであると，同一タスクを処理するのに要する性能や消費エネルギーはサーバ毎に異なり，スケジューリングの仕方によってタスクセットの処理に必要な合計の処理時間や消費エネルギーも異なる．従来から，タスク処理時間のヘテロジニアス性を考慮し，処理を高速化するためのタスクスケジューリングの研究が多く行われてきた[2)–5)]．しかし，サーバのヘテロジニアス性を考慮し，消費エネルギーを削減を狙う研究はこれまでにあまり行われていない．

そこで本稿では，構成要素である計算機の処理性能と消費電力がヘテロジニアスである計算機クラスタシステムにおいて，先行制約を持つタスクセットを対象に，低消費エネルギー化タスクスケジューリング手法を提案する．提案手法は，処理性能に着目したスケジューリング手法である Heterogeneous Earliest Finish Time (HEFT) 法のスケジュール結果を基に，計算機内にあるプロセッサのアイドル時，またはスタンバイモード時の消費電力を考慮しつつ，サーバへのタスク再割り当てを行うことで，タスク処理のエネルギーを削減するものである．提案手法では，HEFT 法に比べてスケジュール長を延ばさずにタスクセット処理に必要な消費エネルギーを削減できることが特徴である．

†1 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications

本稿では，従来手法である HEFT 法，および提案手法のスケジューリング方法について述べ，種々な特徴を持つランダムなタスクセットを対象に，HEFT 法と比較しつつ，提案手法の消費エネルギー，およびスケジューリングかかる時間について評価を行う．

2. 関 連 研 究

2.1 タスクスケジューリング手法に関する関連研究

これまでにも，処理性能がヘテロジニアスであるサーバ計算機環境において，依存関係を持つ複数のタスクで構成されるタスクセットの処理を高速に行う，すなわちタスクセットの処理時間を短くするための手法がいくつか提案されている．例えば，Heterogeneous Critical Path Fast Duplication (HCPFD)[2)]，Bubble Scheduling and Allocation (BSA)[3)]，Heterogeneous Earliest Finish Time (HEFT)[4)5)]，Critical Path on a Processor (CPOP)[4)5)]，Critical Path On a Cluster (CPOC)[5)] などがある．これらは，ヘテロジニアスクラスタ計算機環境でクリティカルパス上のタスクをできるだけ高速なコンピュータに割り当てることで，タスクセットの処理時間を短くすることを狙うものである．しかしながら，これらの手法においては，タスクセット処理の消費エネルギーについては考慮されていない．

一方で，タスクセット処理にかかる消費エネルギーを削減するタスクスケジューリングの研究も行われている[5)–8)]．文献 5) はホモジニアス計算機クラスタ環境において，また文献 6) は CPU の動作電圧がそれぞれ異なるヘテロジニアスな環境でプロセッサの DVFS 機能を利用して省エネルギー化を図る研究である．文献 7) はある単位時間におけるプロセッサの使用実績から，次の単位時間のプロセッサの使用予測を行い，予測に基づいて DVFS により消費エネルギー削減を狙うものである．文献 8) はプロセッサよりも，アクセラレータの方が高速に処理できるタスクは積極的にアクセラレータに処理を任せることで，タスク処理の消費エネルギー削減を図る研究である．これらの研究では，タスクによって異なるエネルギー消費や各サーバ計算機のアイドル時，スタンバイモード時の消費電力を意識したスケジューリングは行われていない．

本稿では，クリティカルパス上のタスクをできるだけ高速なコンピュータに割り当てることで，タスクセットのスケジュール長を短くし処理完了時刻を早くする HEFT 法を拡張し，タスク処理のエネルギーを削減する手法を提案する．HEFT 法は，他の手法に比べタスクセットのスケジュール長を短くできることが知られている[8)]．スケジュール長が長いと，処理が行われていないプロセッサのアイドル時の消費エネルギーが増加し，省エネルギー化効果が小さくなりがちである．そこで，スケジュール長が短い HEFT 法の特徴を維持したままタスクセット処理の省エネルギー化を行う事で，既存手法の利点を失う事無くタスクセット処理の省エネルギー化が行えると考え，本研究では HEFT 法をベースのスケジューリング手法として採用した．そのため，次節では HEFT 法について詳述する．

2.2 Heterogenious Earliest Finish Time (HEFT) 法

HEFT 法は，優先度ランク作成フェーズとプロセッサへのタスク割り当てフェーズからなる．以下にそれぞれについて説明する．

優先度ランク作成

タスクの処理優先度を決定するために，各タスクをランク付けする必要がある．そこで，あるタスク n_i について，終端タスク n_{exit} までの最長パス長である $rank_u(n_i)$ を以下のように定義する．

$$rank_u(n_i) = \overline{w_i} + \max_{n_j \in succ(n_i)} (\overline{c_{i,j}} + rank_u(n_j)) \quad (1)$$

$$rank_u(n_{exit}) = \overline{w_{exit}} \quad (2)$$

ここで，$\overline{w_i}$ はタスク n_i の各プロセッサ上での処理時間の平均，$succ(n_i)$ はタスク n_i に直接の依存関係 (先行制約) を持つ後続タスクの集合，$\overline{c_{i,j}}$ はタスク n_i から n_j へデータを転送する場合の通信時間の平均を表す．$rank_u(n_i)$ は再帰的に求められ，n_{exit} は終端のタスクを意味する．このように，$rank_u(n_i)$ は各タスクの処理時間とタスク間の通信時間を含む当該タスクから終端タスクまでの最長パス長となる．

例として，文献 5) で用いられているのタスクセットを**図 1** に示す．図 1 では，ノード内に書かれた数字がタスク番号を，エッジに付けられた数字が通信時間を示している．またプロセッサ proc.1〜proc.3 における各タスクの処理時間を**表 1** に示す．なお，表には $rank_u(n_i)$ の値も示している．この例において，$rank_u(n_7)$ は，タスク n_{10} の各プロセッサでの処理時間の平均とタスク n_7 からタスク n_{10} までの通信時間，そしてタスク n_7 自身の処理時間の平均を合算したものとなる．

なお，文献 5) では前述の CPOP 法のために，先頭のタスクから当該タスクまでの最長パスを表す $rank_d$ も定義されている．$rank_d$ は HEFT 法には用いられないが，本稿での提案手法で利用するため，以下にその定義を述べる．

$$rank_d(n_i) = \max_{n_j \in pred(n_i)} (rank_d(n_j) + \overline{w_j} + \overline{c_{j,i}}) \quad (3)$$

$$rank_d(n_{entry}) = 0 \quad (4)$$

ここで，$pred(n_i)$ はタスク n_i が直接の依存関係 (先行制約) を持つ先行タスクの集合であ

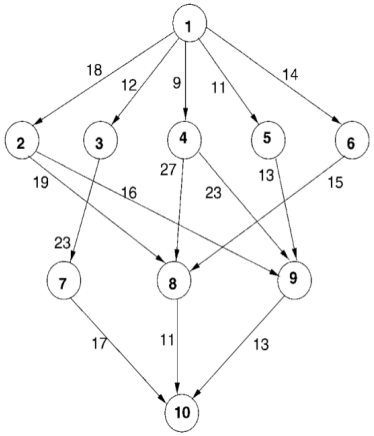

図 1 サンプルのタスクセット (出典: 文献 5))

表 1 各タスクの各プロセッサでの処理時間と処理優先度ランク

	タスク処理時間			優先度ランク		
Task	Proc. 1	Proc. 2	Proc. 3	$rank_u$	$rank_d$	$rank_u + rank_d$
n_1	14	16	9	108.000	0.000	108.000
n_2	13	19	18	77.000	31.000	108.000
n_3	11	13	19	80.000	25.000	105.000
n_4	13	8	17	80.000	22.000	102.000
n_5	12	13	10	69.000	24.000	93.000
n_6	13	16	9	63.333	27.000	90.333
n_7	7	15	11	42.667	62.333	105.000
n_8	5	11	14	35.667	66.667	102.334
n_9	18	12	20	44.333	63.667	108.000
n_{10}	21	7	16	14.667	93.333	108.000

る．タスクセットの先頭タスクの $rank_d$ の値，すなわち $rank_d(n_{entry})$ は 0 である．表 1 には図 1 中の各タスクの $rank_d$ の値も示している．

タスク割り当てフェーズ

まず，$rank_u$ の降順にタスクをソートする．この順で処理を行うプロセッサを決定していく．ここで，どのプロセッサにタスクを割り当てるかは，当該タスクを割り当てた場合に，通信時間も考慮してタスクの処理完了時刻ができるだけ早い時刻で完了できるプロセッサである．この $rank_u$ の値の降順で割り当てることで，処理時間を長く要するタスクがより高速なプロセッサへと割り当てられることになり，タスクセットの処理を高速化できることになる．

3. 省エネルギー化スケジューリング手法

本章では，提案手法であるヘテロジニアスなサーバ計算機環境を考慮した低消費エネルギー化タスクスケジューリング手法について述べる．なお，近年の計算機は通常複数のプロセッサコアを持ち，タスクの割り当てをプロセッサ単位で行うことが考えられるため，以降ではタスクの割り当ての単位はプロセッサとして説明する．

3.1 問題定義

以下に，本稿で提案する低消費エネルギー化タスクスケジューリングアルゴリズムにおける前提，および入出力について述べる．

前　　提

- 各プロセッサの処理性能，および消費電力は異なる
- スケジューリング対象は先行制約付きタスクのタスクセットとする
- タスクセット中の各タスクの各プロセッサにおける処理時間と処理にかかる消費エネルギーはあらかじめ与えられている
- プロセッサの DVFS 機能は使用せず，全てのタスクを最大周波数を用いて処理するものとする
- 異なるプロセッサ上で実行されたタスクからのデータ依存が存在し，データ転送が必要な場合はその通信時間を考慮する
- スケジューリングは静的スケジューリングとし，スケジューリング自体の消費エネルギーは考慮しない
- 各プロセッサにはスリープモード (スタンバイモード) があり，スリープモードへの移行，および復帰には時間的，および消費エネルギー的なオーバーヘッドが存在するものとする
- スリープモードへの移行は，時間的なオーバヘッドを考慮し，移行および復帰にかかる時間以上の連続した空き時間が存在している場合を対象とする
- その上でスリープモードへの移行は，連続した空き時間をスリープモードにすることで削減できるエネルギーよりもエネルギーオーバーヘッドが小さい場合にのみ行う

入　　力

スケジューリングアルゴリズムの入力は次の通りである．

- タスク情報 (タスク間の依存関係，各タスクの各コンピュータにおける処理時間と消費エネルギー，先行タスクから後続タスクへの通信時間)
- 使用できる最大のプロセッサ数
- 各計算機のスリープモードへの移行・復帰にかかる時間，および消費エネルギー

出　　力

スケジューリングアルゴリズムの出力は次の通りである．

- 各タスクの割り当て先プロセッサと，処理開始時刻，処理完了時刻
- タスクセットの処理にかかる時間 (スケジュール長)、および消費エネルギー

3.2 スケジューリング手法

本研究で提案するスケジューリング手法は，HEFT 法によるタスク割り当て結果に対して，タスクセット処理にかかる消費エネルギー削減のため，タスクの再割り当てを行う．再割り当て対象のタスクはクリティカルパス上にないタスクである．再割り当て先の候補となるプロセッサは，対象タスクの最早処理開始可能時刻から最遅処理完了可能時刻までの間に，当該タスクの処理時間分他のタスクが割当たっていないプロセッサである．候補として選択されたプロセッサのうち，タスクセット全体の消費エネルギーが最も少なくなるプロセッサにそのタスクを再割り当てる．以下に，本スケジューリング手法の手順の詳細を述べる．

タスクの処理優先度ランクの作成

タスクの処理優先度ランク作成のため，2.2 節で述べた $rank_u$，および $rank_d$ を求める．さらに，タスクセットのクリティカルパスとして $rank_u$ と $rank_d$ の値を合算したものを用いる．これは，クリティカルパス上のタスクを把握することに利用される．

表 1 には，$rank_u$ と $rank_d$ を合算した値も示している．これらの値がもっとも大きい 108.0000 となっているタスクがこのタスクセットにおけるクリティカルパス上のタスクと考えられる．

HEFT 法によるタスク割り当て

2.2 節で述べた，HEFT 法によるタスク割り当てを行う．

タスクの再割り当て

クリティカルパス上にないタスク n_i に対し，$rank_u$ の値の降順，または昇順にタスクの再割り当てを行う．降順あるいは昇順で再割り当てを行うかについては，後に比較評価を行う．以下に，具体的な再割り当ての手順を述べる．

(1) タスク n_i を p_k に割り当てた場合の最早処理開始可能時刻を求める:
タスク n_i と処理に使用可能なプロセッサ集合 Q 中のプロセッサ p_k の組みに対して，次式によりタスク n_i の最早処理開始可能時刻 ($EST(n_i, p_k)$) を求める．

$$EST(n_i, p_k) = \max_{n_j \in pred(n_i)} (Fin(n_j) + C_{j,i}(p_k)) \quad (5)$$

ここで，$Fin(n_j)$ はタスク n_j の処理完了時刻，$C_{j,i}(p_k)$ はタスク n_i をプロセッサ p_k に割り当てた場合のタスク n_i から n_j へのデータ通信時間を表す．なお，n_j と n_i が同一プロセッサに割り当てられる場合は通信時間は 0 である．

(2) タスク n_i を p_k に割り当てた場合の最遅処理完了可能時刻を求める:
タスク n_i と処理に使用可能なプロセッサ集合 Q 中のプロセッサ p_k の組みに対して，次式により最遅処理完了可能時刻 ($LFT(n_i, p_k)$) を求める．

$$LFT(n_i, p_k) = \min_{n_j \in succ(n_i)} (Start(n_j) - C_{i,j}(p_k)) \quad (6)$$

ここで，$Start(n_j)$ はタスク n_j の処理開始時刻を表す．他は (5) 式と同様である。

(3) タスク n_i の再割り当ての候補プロセッサ集合を求める:
集合 Q のプロセッサのうち，次の条件を満たすプロセッサ p_k の集合 Q_r を求める．p_k 上に時刻 $EST(n_i, p_k)$ から時刻 $LFT(n_i, p_k)$ の期間中に p_k 上での n_i の処理時間以上の連続したアイドル時間 (他のタスクが割り当てられていない間) がある．

(4) タスクの再割り当て:
集合 Q_r 中の各プロセッサに，タスク n_i を割り当てた場合のタスクセットの処理にかかる消費エネルギー (タスク処理の消費エネルギー，全プロセッサのアイドル時の消費エネルギー，スリープモード移行・復帰の消費エネルギーの合計) を算出し，その値が最も少なくなるプロセッサ p' へタスクの再割り当てを行う．この際，p' において EST と LFT 間でタスク n_i の処理が行える時間帯中，どの時刻でタスク n_i の処理を開始するかを決定する必要がある．本稿では，できる限り EST に近い時刻を開始時刻とする場合 (以下，上詰め)，できる限り LFT に近い時刻を開始時刻とする場合 (以下，下詰め) の両者を検討する．なお，4 章では両者を比較評価する．

図 2 は再割り当て対象タスク n_i のあるプロセッサ上での EST と LFT の例を示している．図は HEFT 法によるスケジューリング結果の一部を示しており，タスク n_1 および n_2 はタスク n_i の先行タスク，タスク n_7 および n_8 はタスク n_i の後続タスクであることを示している．図中の実線の矢印はそれらの依存関係を示しており，データを通信するためにそ

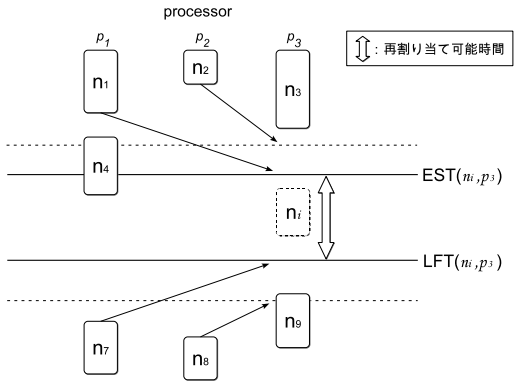

図 2 最早処理開始可能時刻 EST と最遅処理完了可能時刻 LFT

の分の通信時間が必要であることを矢印の高さで表している．

図 2 において，タスク n_1 の完了時刻から n_1 から n_i への通信時間を加えた時刻がタスク n_i のプロセッサ p_3 上での最早処理開始可能時刻 $EST(n_i, p_3)$ となる．また，タスク n_7 の開始時刻より、n_i から n_7 への通信時間分を引いた時刻がタスク n_i のプロセッサ p_3 上での最遅処理完了可能時刻 $LFT(n_i, p_3)$ となる．この場合，$EST(n_i, p_3)$ と $LFT(n_i, p_3)$ の間の時間帯において，プロセッサ p_3 でのタスク n_i の処理時間分の空きがあれば，タスク n_i をプロセッサ p_3 へ移動させることが可能となる．

まとめとして，提案するスケジューリング手法の一連のアルゴリズムを**図 3** に示す．

4. 評　　価

4.1 評価の仮定

提案するスケジューリング手法を，タスクセットの処理時間 (スケジュール長) と消費エネルギーに関して，HEFT 法のスケジューリング結果と比較しつつ評価を行う．

1 つのタスクセット中のタスク数は 100 個とし，ランダムタスクセット 180 個[10] を使用して実験を行った．

タスクの処理時間は，最も高速なプロセッサで実行した場合に最小処理時間になり，最も低速なプロセッサで実行した場合に最大処理時間になるとし，それ以外の各プロセッサで実

1. Set the computation costs of the tasks and communication costs of the edges with means values.
2. Compute $rank_u$ for all tasks by traversing graph upward, starting from the exit task.
3. Sort the tasks in a scheduling list by nonicreasing order of $rank_u$ values.
4. **while** there are unscheduled tasks in the list **do**
5. Select the first task, n_i from the list for scheduling.
6. **foreach** processor p_k in the processor-set ($p_k \in Q$) do
7. Compute Ealiest Finish Time $EFT'(n_i, p_k)$ value.
8. Assign task n_i to the processor p_j that minimizes EFT of task n_i
9. **endfor**
10. **endwhile**
11. Compute $rank_d$ for all tasks by traversing graph downward, starting from the entrance task.
12. Compute $rank_u$ value plus $rank_d$ value.
13. Set the tasks that have the largest value of ($rank_u + rank_d$) to a criticapath list L_c.
14. Sort the tasks in a reassigne list by nonicreasing order of $rank_u$ values.
15. **while** there are un-reassigned tasks in the list **do**
16. Select the first task, n_i from the list for reassignment.
17. **if** task n_i is not criticalpath tasks ($n_i \notin L_c$) **then**
18. set $Q_r = \phi$
19. **foreach** processor p_k in the processor-set ($p_k \in Q$) do
20. Compute $EST(n_i, p_k)$ value.
21. Compute $LFT(n_i, p_k)$ value.
22. **if** p_k has un-assigned time duration for n_i from $EST(n_i, p_k)$ to $LFT(n_i, p_k)$ **then**
23. $Q_r = Q_r + p_k$
24. **endif**
25. **endfor**
26. Select processor $p' \in Q_r$ which minimize total energy consumption of the taskset
27. Reassign n_i to the processor p'
28. **endif**
29. **endwhile**

図 3 提案する省エネルギー化スケジューリング手法のアルゴリズム

行した場合は，最大処理時間と最小処理時間を線形補間した実行時間となるように仮定した．また，タスク処理時の消費電力は，最高速のプロセッサで実行した場合の消費電力を最大消費電力，最も低速のプロセッサで実行した場合の消費電力を最小消費電力とし，それ以外の各プロセッサの消費電力は最小消費電力と最大消費電力を線形補間した値とした．その

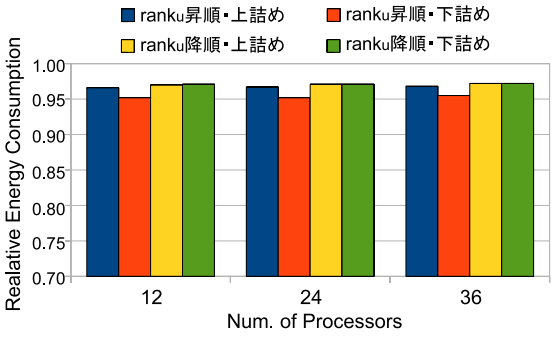

図 4 HEFT 法に対する相対消費エネルギー (タスク処理時間の最大・最小比=1.5，電力の最大・最小比=1.5)

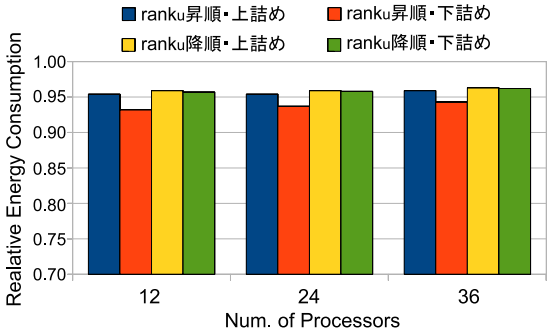

図 5 HEFT 法に対する相対消費エネルギー (タスク処理時間の最大・最小比=2.5，電力の最大・最小比=1.5)

上で，各タスクの消費エネルギーは各プロセッサでの当該タスクの消費電力と処理時間を乗じたものとした．なお，各タスクの処理時間と消費エネルギーは，それぞれのタスクセット中の 100 タスクを 4 つのグループに分け，上記の基準に従ってそれぞれのグループ毎に処理時間と消費エネルギーを決定した．

また，使用可能なプロセッサ数は 12，24，36 の 3 通りを評価した．各タスクの通信時間は，当該タスクに先行制約を持つ各先行タスクの処理時間に比例するものとし，その比率は 0.5 とした．また，タスクセット内のタスク処理時間の最大・最小比は 1.5 と 2.5，タスクセット内のタスク処理にかかる消費電力の最大・最小比も 1.5 と 2.5 の各場合を評価した．

3.1 節の問題定義の前提で述べたスリープモードへの移行，および復帰に要する時間はそれぞれ 5 単位時間とし，各動作ともに，各プロセッサの最小負化時 (アイドル時) の 2.5 倍の消費電力を要するものとした．そのため，スリープモードへの移行と復帰に合計 10 単位時間が必要であることから，プロセッサに 10 単位時間以上の連続した空き時間が存在し，かつスリープモードに移行するエネルギーオーバーヘッドを考慮しても消費エネルギーが削減できる場合にのみ，スリープモードへ移行することになる．なお，スリープモードで消費される電力は 0 とした．

4.2 評価結果

図 4 に，HEFT 法に対する提案手法の相対消費エネルギーをプロセッサ数が 12，24，36 のそれぞれの場合について示す．図の各棒グラフは再割り当ての際に $rank_u$ の昇順/降順で行うか，また EST 側に詰めて配置するか LFT 側に詰めて配置するか (上詰め/下詰め) のそれぞれの評価結果を示している．なお，タスク処理時間のプロセッサ間の最大・最小比，電力の最大・最小比ともに 1.5 の場合の結果である．

図を見ると，全ての場合で提案するスケジューリング法により，タスクの再割り当てを行うことで，HEFT 法の場合に比べ，消費エネルギーが削減できていることがわかる．これは，再割り当てにより，タスクセットのスケジュール長を延ばさない範囲でより省エネルギーなプロセッサへできる限りタスクを配置することで，実際に消費エネルギーが削減できたためである．このことから，提案手法はヘテロジニアスなクラスタプロセッサシステムにおいて，先行制約を持つタスクセットをスケジューリングする上で消費エネルギー削減に有効な手法であると考えられる．

次に，タスク再割り当ての際の順番と配置側について考察する．図を見ると，全てのプロセッサ数の場合で $rank_u$ の昇順・下詰めで再割り当てを行う場合が，消費エネルギー削減効果が大きいことがわかる．プロセッサ数が 12 の場合，$rank_u$ の昇順・下詰めで再割り当てを行うことで，4.8%消費エネルギーが削減できている．これは，$rank_u$ の昇順でタスクの再割り当てを行うことで，タスクセット後方からの再割り当て順序となり，移動できるタスク候補が多くなったためと考えられる．また，その際にタスク下詰めで再割り当てを行うことで，先行するタスクにも移動可能候補の余地が多くなり，再割り当て可能なタスクが全体として増加したため，消費エネルギー削減効果が大きくなったものと考えられる．

図 5 にタスク処理時間のプロセッサ間の最大・最小比を 2.5 にした場合，および**図 4.2** にタスク処理にかかる電力の最大・最小比を 2.5 にした場合の結果を示す．これらの図を見ると，あるタスクの処理時間や処理にかかる電力のプロセッサ間でのばらつきが大きくなる，すなわちヘテロジニアス性が拡大すると，提案手法による消費エネルギー削減効果が大きく

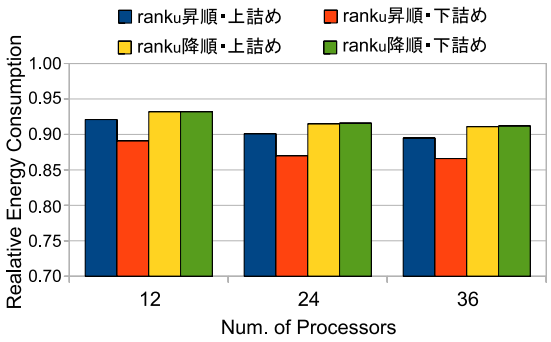

図 6 HEFT 法に対する相対消費エネルギー (タスク処理時間の最大・最小比=1.5，電力の最大・最小比=2.5)

表 2 スケジューリング自体に要する時間 [秒]

プロセッサ数	12	24	36
HEFT 法	0.32	0.60	0.94
提案手法	1.44	3.19	5.74

なることがわかる．処理時間のヘテロジニアス性が大きい場合タスク再割り当ての移動可能候補が増加し，また電力のヘテロジニアス性が大きい場合は，タスクを再割り当てした際に，より消費電力が小さなプロセッサに割り当てることができる可能性が増大するためと考えられる．提案手法を用いることにより，図 4.2 のプロセッサ数 36 における $rank_u$ の昇順・下詰めの結果において，最大で 13.4%もの消費エネルギーが削減できている．

また，評価結果全体を通して言えることとして，使用可能なプロセッサ数が多くなるほど消費エネルギーが削減効果が増加していることがあげられる．これも，タスク再割り当ての移動可能候補の増加と，より消費電力が小さなプロセッサに割り当てができる可能性が増大することが理由である．

4.3 スケジューリング自体に要する時間の評価

表 2 に HEFT 法と本提案手法でタスクスケジューリングに要する時間を示す．今回の評価では Core2 Duo 1.42GHz，主記憶 2GB の計算機を使用し，スケジューリングを行うプログラムを C++で記述した場合の結果である．また，表中の値は，評価に用いた 100 タスクからなるランダムタスクセット 180 個のスケジューリング時間の平均を示している．

表 2 のように，HEFT 法と本提案手法でタスクスケジューリングに要する時間を比較すると，プロセッサ数 36 の場合では，提案手法は HEFT 法の約 6 倍のスケジューリング時間がかかってしまっている．しかし，一般的には 100 個タスクセットの処理に要する時間はさらに長く，また一度スケジューリングした結果を何回か利用することも考えられるため，タスクスケジューリングに要する時間としては，大きな問題とならないと考えられる．

5. ま と め

本稿では，先行制約を持つタスクセットを対象に，ヘテロジニアスなサーバ計算機環境を考慮した低消費エネルギー化タスクスケジューリング手法を提案した．提案手法は，従来の (HEFT 法のスケジューリング結果を基に，サーバへのタスク再割り当てを行うことで，タスク処理のエネルギーを削減するものであり，タスクセットのスケジュール長を延ばさずに，消費電力を削減することができる．

本提案手法を評価したところ，従来の HEFT 法に比べ，最大で 13.4%ほど消費エネルギーを削減できることがわかった．評価の結果より，提案手法はヘテロジニアスなクラスタ計算機システムにおいて，先行制約を持つタスクセットをスケジューリングする上で消費エネルギー削減のために有効な手法であると考えられる．

今後の課題としては，より多くのケースについて評価をし，提案手法の有効性を確かめることがあげられる．また，さらに消費エネルギーを削減できるようにスケジューリング手法を改良することの他，スケジューリング自体に要する時間を短縮できるアルゴリズムの開発なども今後の課題である．

謝辞 本研究の一部は，文部科学省科学研究費補助金 (基盤研究 (C) No.22500043)，および科学技術振興機構戦略的創造研究推進事業 (CREST) の研究プロジェクト「革新的電源制御による超低電力高性能システム LSI の研究の支援によって行われた．

参 考 文 献

1) Report to Congress on Server and Data Center Energy Efficiency, Public Law 109-431, U.S. Environmental Protection Agency and ENERGY STAR Program, August 2, 2007
2) Kwok, Y.-K. and Ahmad, I.: Exploiting duplication to minimize the execution times of parallel programs on message-passing systems, Proc. Int ’l Par. Proc. Symp., pp. 426-433 (1994).
3) Kwok, Y.-K. and Ahmad, I.: Link contention- constrained scheduling and mapping of tasks and messages to a network of heterogeneous processors,Proc. Int ’l Conf.

Par. Proc., pp. 551-558 (1999).
4) Topcuoglu, H., Hariri, S.and Wu, M.-Y.: Task scheduling algorithms for heterogeneous processors, Proc. Hetero. Comp. Workshop ,pp.3-14 (1999).
5) H.Topcuouglu et.al ”Performance Effective and Low Complexity Task Scheduling for Heterogeneous Computing ”, IEEE Trans. Parallel Dist. Systems, vol.10, no.8 pages 795-812(1999).
6) Young Choon Lee et al. ”Minimizing Energy Consumption for Precedence-constrained Applications Using Dynamic Voltage Scaling ”, Zomaya, 9th IEEE/ACM International Symposium on Cluster Computing and the Grid(CCGrid), Pages 92- 99(2009)
7) K.H.Kim et al. ”Power Aware Scheduling of Bag- of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters ”, International Symposium on Cluster Computing and the Grid IEEE/ACM, pages 541-548(2007).
8) Dusit Niyato et al. ”Optiomal Power Management for Server Farm to Support Green Computing ”, International Symposium on Cluster Computing and the Grid IEEE/ACM, pages 84-91(2009).
9) 浜野 智明, 遠藤 敏夫, 松岡 聡ヘテロ並列環境のための省電力タスクスケジューリング 電子情報通信学会技術研究報告. CPSY, コンピュータシステム 108(180), 97-102, (2008)
10) Standard Task Graph Set, 早稲田大学 笠原・木村研究室, http://www.kasahara.elec.waseda.ac.jp/schedule/