Vol.2011-HPC-130 No.25
2011/7/28

プログラムの処理速度調整に基づいたデータセンタ向け省電力タスクスケジューリング法

脇坂 洋祐[†1] 北道 淳司[†2] 柴田 直樹[†3]
安本 慶一[†1] 伊藤 実[†1]

近年，データセンタではプロセッサおよび空調設備の消費電力の増加が問題になっている．本稿では，この問題解決のためにプロセッサの動作周波数および電源電圧を下げることでプロセッサと空調設備の消費電力削減を図るタスクスケジューリングアルゴリズムを提案する．提案アルゴリズムでは入力プログラム内の各タスクをプロセッサに割り当て，予測した処理完了時刻がデッドラインより早い場合には，全てのタスクの中で最も消費電力の大きなタスクが割り当てられているプロセッサの電源電圧と動作周波数を下げ処理時間を延ばすことで消費電力を下げる．評価実験により，提案アルゴリズムは電源電圧と動作周波数を低下させない場合と比べて，電力コストを 9% から 47% 削減できることが判った．

Energy-efficient task scheduling method for the data center based on adjustment of program's execution time

Yosuke WAKISAKA,[†1] Junji KITAMICHI,[†2]
Naoki SHIBATA,[†3] Keiichi YASUMOTO[†1]
and Minoru ITO [†1]

Recently, many data centers have been suffering from increase of their running cost. In this paper, we propose a task scheduling algorithm that reduces the power consumption of processors and air conditioning by decreasing a frequency and source voltage of processors for solving this problem. In the proposed algorithm, first, all tasks of the input program are allocated to processors. Next, we estimate the completion time when running the actual processing. Then, if estimated completion time is earlier than the program's deadline, this algorithm reduces the power consumption by deferring the execution time of tasks in the decreasing order of the power consumption and decreasing the frequency and source voltage of the corresponding processor. Through simulations, we confirmed that our proposed algorithm reduces power consumption by 9% to 47% compared with an existing algorithm without execution time adjustment.

1. はじめに

近年，クラウドコンピューティングの普及や膨大なデータを扱うプログラムの増加に伴い利用者のデータセンタに対する性能要求が急速に高まっている．データセンタはこれらの要求に応えるために高性能な演算装置やネットワークの導入を行うことで大規模化し，データセンタのプロセッサと空調システムの電力コストは運営コスト全体の 63% に達している．そうした中でデータセンタの市場規模は 2008 年には 1 兆 2,400 億円となり，2013 年までに平均 5.6% ずつ増加すると予想されている．一方，消費電力量は 2008 年に 65 億 kWh であり，2013 年までに平均 10% ずつ増加することが見込まれており，データセンタの市場の伸びを上回る勢いで増加している[8]．

高性能な演算装置などを導入したことで発生したデータセンタの運営コストの増加問題に対してデータセンタの消費電力を削減するためのタスクスケジューリングアルゴリズムが数多く提案されている[2)9)10)11)]．これら既存研究は入力されたプログラムを速く処理することを前提としてタスクの割り当てやプログラムの実行で使用する必要のない計算機器の電源を落として消費電力を削減するものが多い．しかし処理されるプログラムはそれぞれが許容できる処理完了期間（以下ではデッドラインと呼ぶ）を持っており，デッドライン内であればプログラムの実行時間が遅くても問題はない．また一般的にプロセッサの消費電力はプロセッサの電源電圧の二乗および動作周波数に比例して増加し，空調システムの消費電力はプロセッサからの発熱量に比例して増加するので，タスクの処理時間を延ばしてプロセッサの電源電圧および動作周波数を低下させることで，消費電力を削減することが可能である．一般的に消費電力を最小にするスケジュールを見つけるスケジューリング問題は NP-困難であるため本稿では発見的問題解決法を用いて近似解を見つける．

本稿では空調コストとプロセッサの消費電力に注目し，プロセッサの動作周波数と電源電圧を下げ，タスクの処理完了時刻をデッドラインまで延ばすことによって消費電力を削減するタスクスケジューリングアルゴリズムを提案し，評価する．提案手法ではまずプログラム

†1 奈良先端科学技術大学院大学 情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
†2 会津大学大学コンピュータ理工学研究科
School of Computer Science and Engineering, The University of Aizu
†3 滋賀大学経済学部情報管理学科
Faculty of Economics, Shiga University

内の全てのタスクを最短の処理時間で終わるように各プロセッサに割り当てる．次に，もしスケジュールされたプログラムの終了時刻とプログラムのデッドラインの間に余裕が有れば，全てのタスクの中で消費電力が最大となるタスクが割り当てられているプロセッサの電源電圧および動作周波数を下げることによってタスクの処理時間を延ばし消費電力の削減を試みる．

提案アルゴリズムによる消費電力の削減を確認するために 2 つの並列度の異なるタスクグラフを用いてシミュレーションによる比較実験を行った．実験では各タスクグラフを既存の手法でスケジューリングした時の処理完了時刻を 1.5 倍に延長した時刻をデッドラインとして用いた．その結果，全体の消費電力を 9% から 47% 削減できることを確かめた．

以降 2 章では，関連研究について述べる．3 章ではタスクスケジューリング問題について述べ，入出力および出力が満たすべき制約を定義し，問題の定式化を行う．4 章では提案するタスクスケジューリング手法の詳細を述べる．5 章では提案したアルゴリズムを既存の手法と比較するために行ったシミュレーションについて述べる．最後に 6 章では本研究のまとめを述べる．

2. 関 連 研 究

データセンタの消費電力削減のため数多くの手法が提案され実装されている．データセンタの消費電力を削減する方法としては主に不必要な計算機器の電源を落とす方法[4)7)9)]，空気の循環を考慮した方法[6)11)]，タスクの割り当てを考慮した方法[2)3)5)10)] などがある． 文献 9) では，サーバ群とネットワーク機器が設置されたデータセンタを対象に，それらの機器の設定を管理する省電力サーバを新たに設置することで各サーバの使用状況を監視し，使用状況の少ない機器を休止する手法を提案している．この手法では複数のサーバでまばらに動作している仮想マシン群を一つのサーバにまとめることで，使用する必要がなくなったサーバとそのサーバのみに関係するネットワーク機器，空調システムを休止することで省電力化を行っている．プロセッサの状態は, 動作するか休止するかの 2 状態であり, 本稿で目的としているプロセッサの動作周波数及び電源電圧を変化させる方法とは異なる． 文献 10) ではタスクの割り当てとサーバの状況（ON/OFF の状態割り当て）を同時に考え, タスクの割り当てとサーバの状況を整数線形計画問題として定式化し，多項式時間で解くことのできるヒューリスティックアルゴリズムを提案している．この手法は実際のタスクの割り当てやサーバの状態管理を行っているが，サーバの状態は ON あるいは OFF のみであり, 本稿で目的としているプロセッサの動作周波数及び電源電圧を変化させる方法とは異なる． 文献 11) では，サーバに吸気する冷気の温度をサーバを冷却するために最低限必要な温度にすることによって必要な空調システムのコストを抑える手法を提案している．この手法ではサーバに吸気する冷気の温度制御を対象としており，プロセッサへのタスクの割り当てや，プロセッサの動作周波数および電源電圧については考えられていない．したがって，プロセッサの動作環境を変更させる我々の方法とは異なる． 文献 2) ではシステムレベルで発熱を制御するための温度に注意した作業負荷の配置を提案している．この手法では定常状態ホットスポットやコールドスポットについての情報を用いて論理熱力学に基づく定式化を行い，スケジューリングアルゴリズムを工夫することで省電力化を図っている．これはプロセッサの動作環境を変更させる我々の方法とは異なる．

以上のように，様々なアプローチの省電力化が提案されているがプロセッサの動作周波数および電源電圧を制御による省電力化は提案されていない．

3. 電力消費を最小化するタスクスケジューリング問題

本稿では，タスクスケジューリング問題に対して, プロセッサの発熱およびそれを冷却するための空調設備の電力コストを考え, 電力コストを最小にするスケジューリング問題を扱う．以下ではプロセッサの消費電力モデルと空調システムの電力モデルを定義し，電力消費を最小化するタスクスケジューリング問題を定義する．以下の議論で用いる記号を表 1 で定義する．

3.1 電力モデル

各プロセッサの消費電力と発熱量のモデル，空調システムの消費電力モデルを定義する．プロセッサの消費電力は，プロセッサの動作周波数と電源電圧の二乗に比例して増加する．一般に電圧と動作周波数の間には関係があり，また，CPU だけでなくハードディスクなど周辺機器の消費電力を含めると，複雑な関係が存在するが，本研究では簡単のため，動作周波数と消費電力のみの関係を考え，発熱係数倍を乗じた簡単なモデルを用いる．空調システムの消費電力はデータセンタ室温が部屋の熱許容量以下であるとき自然放熱によって自然に冷やされるので空調システムを稼働させる必要がなく 0 となる．しかし，室温が熱許容量を超えた場合は空調システムの稼働が必要になるため空調に必要なコストは室温の高さに応じて増加する．

定義 1 プロセッサの消費電力: 各プロセッサ $p_i \in Proc$ において， 動作周波数が hp の時の消費電力を決定する関数 $Pow(i, hp)$ を以下のように定義する．ここで，To は外気温（室温）であり，Rc は部屋の熱許容量である（T_O と部屋の容積から計算される）．

表 1 記号表
Table 1 Used Notations

$Proc$	プロセッサ全ての集合
p_i	i 番目のプロセッサ
$Pow(i, hp)$	プロセッサ p_i が動作周波数 hp で動作するときの消費電力
B_i	i 番目のプロセッサのアイドル状態の消費電力
$A(x, To)$	室内で x の発熱があった時の空調システムの消費電力
To	外気温
Rc	部屋の熱許容量
P	入力として与えられるプログラム
V	全タスクノードの集合
E	タスク間のエッジの集合
n_i	i 番目のタスク
$w(n_i)$	タスク n_i の仕事量
e_{ij}	タスク n_i からタスク n_j への通信
$c(e_{ij})$	エッジ e_{ij} のコミュニケーションコスト
N	入力として与えるネットワーク
Net_{ID}	各スイッチごとのネットワーク ID
$Proc_{ID}$	1 つのスイッチに接続されているプロセッサの番号
$Bandwidth$	ネットワークの転送速度
PF_i	i 番目のプロセッサの処理能力
dl	入力として与えられるプログラムの許容できる処理期間
$proc(n_i)$	タスク n_i が割り当てられているプロセッサ
$t_f(n_i, p_i)$	タスク n_i がプロセッサ p_i 上で実行された時の終了時間
$t_{ef}(e_{ij})$	タスク n_i からタスク n_j への通信完了時刻
$t_s(n_i, p_i)$	タスク n_i がプロセッサ p_i 上で実行された時の開始時間
$t_d(e_{ij})$	エッジ e_{ij} の通信終了時間
$etime(n_i, p_i)$	タスク n_i をプロセッサ p_i で実行したときの処理時間
$t_{dr}(n_j, p_i)$	タスク n_j の実行を開始するまでに掛る時間
$pred(n_i)$	タスク n_i の直接先行のタスクの集合
$succ(n_i)$	タスク n_i の直接後任のタスクの集合
$bl(n_i)$	タスク n_i の全ての先祖ノードをたどった中で最大の処理時間
c_{cap}	空調システムの処理能力

$$Pow(i, hp) = hp^2 + B_i \tag{1}$$

α は各プロセッサの発熱量の係数とすると，プロセッサ p_i の発熱量は式 (2) で与えられる．

$$\alpha Pow(i, hp) = \alpha(hp^2 + B_i) \tag{2}$$

定義 2 空調の消費電力: 全てのプロセッサ $Proc$ からの発熱量の総和を $h = \Sigma_{p_i \in Proc} \alpha Pow(i, hp)$ としたときに．空調の消費電力を次のように定義する．

$$A(h, To) = \begin{cases} 0 & h \leq Rc \\ \frac{(h-Rc)}{c_{cap}} & h > Rc \end{cases} \tag{3}$$

また，データセンタ全体の消費電力 h_{total} は以下の式によって表わされる：

$$h_{total} = h + A(h, To) \tag{4}$$

3.2 タスクスケジューリング問題

ここでは本研究で使うタスクスケジューリングへの入力と出力，出力が満たすべき制約について定義する．入力によって与えられるプログラムはタスクの集合である．以下でプログラムについて定義をし，例を 1 つ示す．

定義 3 プログラム：プログラム P=(V, E, ω , c) は Directed Acyclic Graph(DAG) によって表現する．

入力として与えられるプログラムの例を図 1 に示す．図 1 の各ノードはタスクを表し，ノードに書かれている値はそのタスクの仕事量を表す．エッジはそれぞれのタスクの依存関係を表現している．また 1 つのタスク内での全ての命令は逐次実行され，タスク内に並列性はない．ノードは全ての入力が到着した後処理を開始し，計算が全て終了した後で出力を開始する．さらに出力を行うときは全ての出力を同時にネットワークを介して送る．例えば，タスク e は直接の先行のタスクであるタスク b とタスク c の処理が完了しそれぞれからのデータ通信が完了しない限り処理を開始出来ない．

入力として与えられるネットワークの構成は以下のように定義する

定義 4 ネットワーク：ネットワーク N は $N = (Net_{ID}, Proc_{ID}, Bandwidth, PF_i, \alpha)$ によって定義される．α は各プロセッサごとの発熱量の係数を表している．

各種制約の定義は以下で行う．スケジュールは各ノードの開始時間とプロセッサの関連付

Vol.2011-HPC-130 No.25
2011/7/28

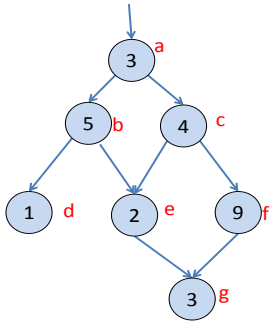

図 1 タスクグラフの例
Fig. 1 An example of the task graph

けであり．スケジューリングでは以下で定義するプロセッサ制約（5），先行制約 (6) とスケジューリング条件 (7) を満たすようにスケジュールを行う．

条件 1 プロセッサ制約：どの 2 つのノード $n_i, n_j \in V$ に対しても，

$$\{proc(n_i) = P_i \wedge proc(n_j) = P_i\} \Rightarrow \left\{ t_f(n_i, P_i) \leq t_s(n_j, P_i) \vee t_f(n_j, P_i) \leq t_s(n_i, P_i) \right\} \quad (5)$$

条件 2 先行制約：$e_{ij} = (n_i, n_j) \in E$(以降，$e_{ij} = (n_i, n_j)$ とする)，$n_i, n_j \in V$, $P_i \in Proc$ に対して，

$$t_f(e_{ij}) \leq t_s(n_j, P_i) \quad (6)$$

条件 3 スケジューリング条件：$G = (V, E, \omega, c)$ において，$[A, B], A, B \in [0, \infty]$ をプロセッサ $P_i \in P$ のアイドル時間インターバルとする． 割り当て可能なノード $n_i \in V$ は $[A, B]$ の中でプロセッサ P_i にスケジュールされる．

$$Max(A, t_{dr}(n_i, P_i)) + etime(n_i, P_i) \leq B \quad (7)$$

プロセッサ制約はプロセッサ内で並列性がないことを意味し，先行制約は直接の後継のタスクが直前のタスクからのすべてのデータ転送を完了したとき処理を始めることを意味している．

各ノード n がプロセッサ P_i で処理を終了する時間は $t_f(n_i, P_i) = t_s(n_i, P_i) + etime(n_i, P_i)$ によって求められる． ノード $n_j \in V$ がノード n_j のエッジの入力によって決められたプロセッサ $P_i \in Proc$ で実行を開始することができる最も早い時間を Data Ready Time（DRT）といい，以下の関数 t_{dr} として定義する．

$$t_{dr}(n_j, P_i) = Max_{e_{ij} \in E, n_i \in pred(n_j)}(t_{ef}(e_{ij})) \quad (8)$$

よって，以下が成り立つ．

$$t_{dr}(n, P_i) \leq t_s(n, P_i) \quad (9)$$

このシステムモデルではエッジの終了時間はオリジナルノードの終了時間とコミュニケーション時間に依存する．エッジの終了時間は以下のようにして求められる．

定義 5 エッジの終了時間：エッジ $e_{ij} \in E$ の終了時間は以下の関数によって与えられる．

$$t_{ef}(e_{ij}) = t_f(n_i) + \begin{cases} 0 & if\ proc(n_i) = proc(n_j) \\ c(e_{ij}) & otherwise \end{cases} \quad (10)$$

エッジの終了時刻は後続のタスク n_j が同じプロセッサ上で実行されるならばプロセッサ間の通信は行われないためエッジによる遅延は 0 であり先行のタスク n_i の終了時間と等しい．しかし実行されるプロセッサが異なる場合はデータの転送が発生するためエッジの通信時間が加算さる．

3.3 問題の定式化

この節では，本稿で扱う問題の定式化を行う．本問題の入出力，制約，目的関数としてはそれぞれ以下のものを与える．

入力データ

(1) 入力プログラム P
(2) ネットワーク構成 N
(3) 空調システムの特性 To, Rc
(4) デッドライン dl

出力データ

(1) プロセッサへのスケジューリング結果
(2) 全体の消費電力値

制約

(1) プロセッサの制約 (5)
(2) 先行制約 (6)
(3) スケジューリング条件 (7)
(4) デッドライン dl 内でプログラムの実行を終了する

目的関数は以下の式で定義する

$$minimize\,(h + A(h, To)) \tag{11}$$

4. 提案する消費電力を考慮したタスクスケジューリングアルゴリズム

本章では，提案アルゴリズムについて述べる．入力のプログラムを実行する上で，先行するタスクが複数ある場合，すべての先行するタスクが終了しなければ後続のタスクは実行を開始できない．そのためスケジューリングによっては，先行する一方のタスクが終了した後，プロセッサには他のタスクが割り当てることの出来ない時間が発生する場合がある．以降この時間を**アイドル時間**と呼ぶ．また，プログラムがデッドラインより早く終了した場合も，アイドル時間が発生する場合がある．提案するアルゴリズムでは，このアイドル時間に着目し，デッドラインまでにプログラムを終了させるという制約が満たされる範囲内で，プロセッサの電源電圧および動作周波数を下げることによってプロセッサおよび空調設備の消費電力を削減する．省電力化は，タスクのプロセッサへの割り当てを変更することによっても実現できるが，本研究では対象としない．提案するアルゴリズムは，二段階の手順によって行われる．まず入力プログラムの全てのタスクを Sinnen らのスケジューリングアルゴリズム[1] を簡単化したものにより各プロセッサに割り当てる．このスケジューリングでは，最高の電源電圧および動作周波数で，実行可能なタスクをプロセッサに割り当てるので，アイドル時間が発生する場合がある．その後，2 段階目として消費電力が大きいタスクから順にデッドラインまでに処理を終了するという制約を満たしながら，タスクの実行時間を延ばすことによって，消費電力を削減していく．各段階のアルゴリズムを **4.1** および **4.2** で述べる．

4.1 簡易版 Sinnen らのアルゴリズム

ここでは Sinnen らのスケジューリングアルゴリズム[1] の簡易版について述べる．このアルゴリズムは，3 章で述べたプロセッサ制約（5）と先行制約（6）を満たすように各ノードと各プロセッサをスケジュールを行う．ノードは優先度に従ってソートされる．優先度は開始ノードから，各ノードに到達するまでの最長経路を表す**ボトムレベル** (bl) を用いる．bl は，以下のように再帰的に定義される．

$$bl(n_i) = \omega(n_i) + Max_{n_j \in succ(n_i)}\{c(e_{ij}) + bl(n_j)\} \tag{12}$$

$succ(n_i)$ はタスク $n_i \in V$ の直接の後続全てのタスクを表す関数である．

リストスケジューリングは，以下のようにソートされたタスクを，実行可能なプロセッサに順に割り当てる．

リストスケジューリングアルゴリズム：

(1) リスト L の中に V の全ノードが先行制約とボトムレベルに従って昇順にソートされる
(2) リスト L の各要素に対し先頭から順に以下の処理を行う
  2-a n_i の処理をもっとも早く処理完了できるプロセッサ $P_j \in Proc$ を探す
  2-b P_j に n_i をスケジュールする

ノードの開始時刻を決めるために，もっとも早いインターバル $[A, B]$ が条件 3 に従って各プロセッサで調べられる． インターバル $[A, B]$ を見つけるために，ノード n_i の開始時刻は以下のように決められる

$$t_s(n_i, P) = Max(A, t_{dr}(n_i, P)) \tag{13}$$

図 2 を用いてアルゴリズムの動作を説明する．図 2 は Sinnen らのスケジューリングアルゴリズムを用いて図 1 のタスクグラフを 2 つのプロセッサにスケジュールした結果である．例えば，タスク a はプロセッサ 1 に割り当てられ，処理時間は 3ms である．次にタスク c はプロセッサ 1 に割り当てられ，依存関係に従って 3ms から処理を開始する．

4.2 各タスクの実行時間を遅延させるアルゴリズム

ここでは，まず例を用いて提案アルゴリズムの説明を行う．提案アルゴリズムでは，**4.1** で求められた，各タスクの各プロセッサへの割り当てを元に，プロセッサの電源電圧および動作周波数を下げることによって各タスクの処理時間を延ばしていく．まず，4.1 の Sinnen らの簡易版スケジューリングの結果に対して，各プロセッサの実行するタスクが割り当てられていないアイドル時間を求める．次に最大の消費電力のタスクを求め，そのタスクの処理時間に，アイドル時間を考慮してタスクの処理時間を延ばす．この変更に伴いタスクの依存関係と全てのタスクのパラメータは調整され，処理時間が延ばされたタスクが割り当てられていたプロセッサの電源電圧および動作周波数を低下させる．延ばされた結果がデッドライン内にプログラムが終了するという制約条件をみたせば，結果をスケジュール結果に反映し，そうでなければこの変更をスケジュール結果に反映させない．これらの処理は全てのタスクに対して繰り返し行う．もし，全てのタスクに対して実行した後，アイドル時間を

Vol.2011-HPC-130 No.25
2011/7/28

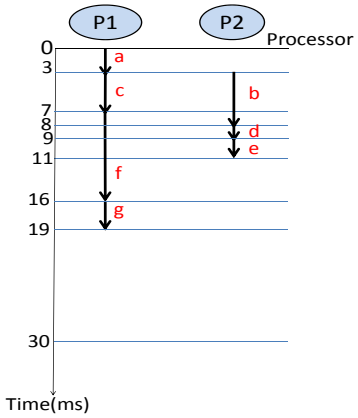

図 2 簡易版 Sinnen らのスケジューリングアルゴリズムによるスケジューリング結果
Fig. 2 Scheduled result by simply list scheduling algorithm

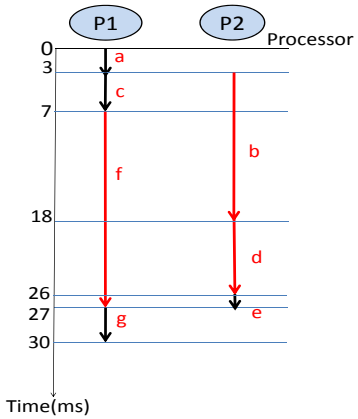

図 3 提案するアルゴリズムによってスケジュールされた結果
Fig. 3 Scheduled result by proposed algorithm

再計算し，全てのタスクに対して再計算されたアイドル時間を用いて再度この処理を行う．このアルゴリズムの疑似コードを Algorithm1 に示す．

図 1 のタスクグラフに Algorithm1 を適用した例を次に示す．デッドラインを 30 として，図 1 を 2–3 行目の簡易版 Sinnen らのスケジューリングアルゴリズムによって初期割り当てしたものが図 2 に相当し，5 行目でこの結果のコピーをリスト C に代入する．この結果の中で消費電力が高いタスクは順に $f \geq b \geq d$ であるとすると，15 行目でまず最初にタスク f が選択され，16 行目でタスク f の処理時間にアイドル時間を加算する．そしてタスク f を延ばしたことで他のタスク間の依存関係が崩れないように 18 行目で依存関係のあるタスク g の開始時間と終了時間の変更を行う．タスク f は割り当てられたプロセッサ上での前後関係もタスク b だけなので 19 行目では何も行わない．最後に延ばした後のプログラムの終了時間が dl を超えていれば結果を破棄してひとつ前の結果に戻る．そうでなければ結果を保存して次のタスクへ処理を移行する．以後同じようにタスク b，タスク d についても同様の処理を行う．このアルゴリズムを図 2 のスケジューリング結果に適用した結果は図 3 になる．

Algorithm 1 プログラムの速度調節に基づいた割り当て

```
1: Define COUNTMAX=10
2: 全てのタスクノード n を先行制約を満たした状態で bl(n) のボトムレベルの順でソートを行いリスト L に格納する．
3: リスト L に格納されたタスク群をタスク間の依存関係を守りつつ最も早く処理されるようにプロセッサに割り当てる．
4: dl と初期割り当てから計算されるプログラムの処理完了時刻の差をアイドル時間とする．
5: 初期割り当ての結果をコピーしてリスト C に代入する．
6: 変数 count, tnum を 0 に初期化しておく．
7: while true do
8:     if count=COUNTMAX then
9:         ループから抜ける．
10:    end if
11:    while true do
12:        if tnum≥ タスクの総量 then
13:            ループから抜ける．
14:        end if
15:        リスト C から消費電力が最大のタスク t を探し出す．
16:        t の処理時間にアイドル時間を加算する．
17:        延長した処理時間に対応するプロセッサの動作周波数および電源電圧を低下させる．
18:        t と依存関係を守るように t の後続のタスクの開始時間と終了時間を調節する．
19:        割り当てられているプロセッサ上での前後関係を守るように t の後続の開始時間と終了時間を調節する．
20:        if アイドル時間を加算したことにより dl をオーバーした場合 then
21:            今回の最適化の処理をクリアしひとつ前の状態に戻す．
22:        end if
23:        処理された結果のコピーを行う．
24:        今回処理したノードをリスト C から外す．
25:        tnum++
26:    end while
27:    アイドル時間を再計算する．
28:    count++
29: end while
30: 最終的な結果をもとに式 4 を用いて消費電力を計算する．
31: 計算された消費電力と最終的なスケジュール結果を返す．
```

5. 実験結果

本章では，提案アルゴリズムを用いて最適化した結果と **4.1** で示した簡易版 Sinnen らのスケジューリングの結果．すなわちタスクの実行時間を延ばす前の最適化が行われていないスケジューリング結果を比較する．比較のため提案手法と簡易版 Sinnen らのスケジューリ

ングアルゴリズムで二種類の並列度の異なるタスクグラフを実行し．全てのプロセッサの消費電力，空調の消費電力とそれらの合計値を求める．

提案するアルゴリズムの評価環境は以下の通りである．今回の実験のではシミュレーションを行った環境としては Intel Core i7 2600K (3.4GHz)，Ubuntu，x86-64 を用いた．実験で用いるタスクグラフのサイズを縮小したものを図 4 と図 5 に示す．実際に使用する二種類のタスクグラフは図 4 の様な依存関係を持ち処理ノードの個数が 160 個のものと，図 5 の様な依存関係を持ち処理ノードの個数が 133 個のものである． 次に評価実験におけるネットワークの構成について述べる．実験で使用するネットワークの構成は 1 つのスイッチにプロセッサを 1, 2, 3, 4 個接続したものを 2 つ繋げたネットワークを想定する．想定するネットワークでのデータ通信は 1ms 毎に 3kB のデータを転送できるものとし，1 つのエッジにおいて送信するデータが複数ある場合，データの通信は最初にデータ転送を開始したノードがエッジを独占できるものとし，そのほかの通信は待たされる．このネットワーク上に接続されている各プロセッサに **4.1** で示した Sinnen らのスケジューリングを用いてタスクを割り当て，提案手法を適用する前の処理終了時間を計測する．計測した処理終了時刻の 1.5 倍に延長した時刻をデッドラインとして提案手法をスケジューリングされた結果に対して適用する．各プロセッサ数ごとの提案手法の適用前と適用後の処理完了時刻を表 3 に示す．また，実験で用いたプロセッサやネットワークなどの値を表 2 に示す．

提案手法を適用した時の結果を図 6 と図 7 に示す．図 6 と図 7 では，縦軸が消費電力量を表し，横軸が各アルゴリズムごとのプロセッサの数を表している．全体として簡易版 Sinnen らのスケジューリングアルゴリズムより消費電力が削減されている．しかし，並列度の高い図 4 のタスクグラフに提案手法を適用した場合ではプロセッサの数を増やすごとに消費電力が減少していくが，並列度の低い図 5 のタスクグラフの場合ではプロセッサを増やすごとに少しずつ消費電力が増加しているのが判る．提案手法では一度スケジューリングした結果に対して各タスクの処理時間延長の処理を行っているためスケジューリングに掛る時間は増加している．しかしスケジューリングに掛る時間は最大でも 0.24 秒であり，その時の入力プログラムの処理のみに必要な時間がおよそ 900ms であるので影響はないと思われる．この実験の結果から提案手法によって全体の消費電力が 9% から 47% 削減出来たことが判る．

6. ま と め

本稿では，プロセッサと空調システムの消費電力モデルを考え，依存関係などの制約を満たしつつプロセッサおよび空調システムの消費電力を最小化する問題を定式化した．また，

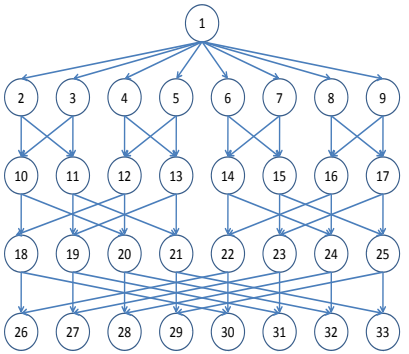

図 4 タスクグラフ 1（詳細省略）
Fig. 4 Task graph 1(details omitted)

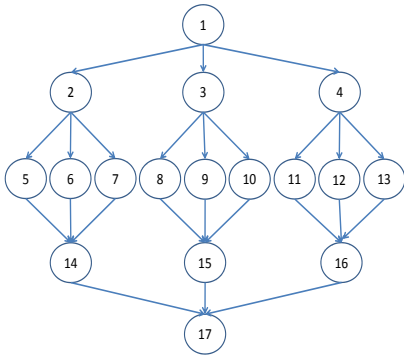

図 5 タスクグラフ 2(詳細省略)
Fig. 5 Task graph 2(details omitted)

表 2 実験で用いたパラメータ
Table 2 Parameter used in this experiment

記号	意味
各タスクの仕事量	データの分割結合をどちらか含む部分は 10，データの分割などを両方含む部分は 30
通信量	10kB
B_i	プロセッサ ID が奇数なら 3.4，偶数なら 4.2
PF_i	10
$Bandwidth$	3kB/ms
Rc	7000
c_{cap}	8

表 3 タスクグラフの処理完了時刻
Table 3 Finish time when executing two task graph

図 4 の処理完了時間			図 5 の処理完了時間		
プロセッサ数 (個)	適用前 (ms)	適用後 (ms)	プロセッサ数 (個)	適用前 (ms)	適用後 (ms)
2	240	360	2	610	915
4	120	180	4	315	473
6	90	135	6	220	330
8	60	90	8	145	218

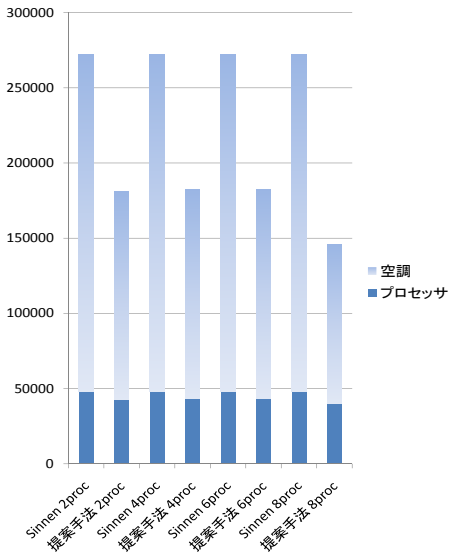

図 6 タスクグラフ 1 に対するシミュレーション結果
Fig. 6 Simulation results for task graph 1

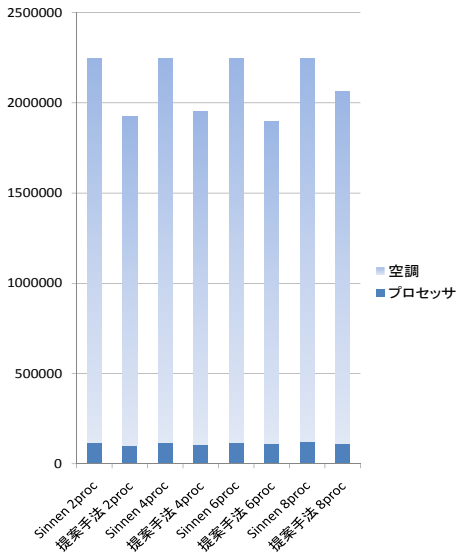

図 7 タスクグラフ 2 に対するシミュレーション結果
Fig. 7 Simulation results for task graph 2

プログラムのデッドライン内でタスクの処理時間を延ばすことでプロセッサの動作周波数および電源電圧を低下させ，プロセッサと空調システムの消費電力を削減するタスクスケジューリングアルゴリズムを提案した．

提案したアルゴリズムを評価するために提案手法と Sinnen らのスケジューリングアルゴリズムを簡単化したものを二種類の並列度の異なるタスクグラフを用いて比較実験をシミュレーションにより行った．この実験を行う際に必要なパラメータのデッドラインは入力として与えられたプログラムを簡易版 Sinnen らのスケジューリングアルゴリズムを用いてスケジューリングした時の処理完了時刻を 1.5 倍に延長した時刻とした．その結果，プロセッサの動作周波数および電源電圧を下げた時にプロセッサと空調システムの消費電力を 9% から 47% 削減することが出来た．

今後の課題としては，本稿ではネットワークの構成を固定としたが，さまざまなネットワーク構成に対する評価実験やネットワークコンテンションを考慮したネットワークモデルの実装，プロセッサや空調システムのパラメータなどをより現実に近づけての検証を行うことである．

参 考 文 献

1) O. Sinnen and L. Sousa, “Communication Contention in Task Scheduling,” *IEEE Trans. Parallel and Distributed Systems*, vol. 16, no. 6, pp. 503-515, June 2005.
2) O. Sinnen and L. Sousa, “Toward a Realistic Task Scheduling Model,” *IEEE Trans. Parallel and Distributed Systems*, vol. 17, no. 3, pp. 263-275, March 2006.
3) J. Moore, J. Chase, P. Ranganathan and R. Sharma, “Making Scheduling “Cool”: Temperature-Aware Workload Placement in Data Centers,” *in 2005 USENIX Annual Technical Conference*, pp. 61-74, April 2005.
4) O. Beaumont, V. Boudet, and Y. Robert, “A Realistic Model and an Efficient Heuristic for Scheduling with Heterogeneous Processors,” *Proc. 11th Heterogeneous Computing Workshop*, 2002.
5) J. Congfeng, X. Xianghua, W. Jian, Z. Jilin, Z. Yinghui, “Power Aware Job Scheduling with Quality of Service Guarantees: A Preliminary Study,” *Proceedings of the Second International Symposium on Networking and Network Security*, pp, 197-200, April 2010.
6) Q. Tang, S. K. S. Gupta, and G. Varsamopoulos, “Thermal-aware task scheduling for data centers through minimizing heat recirculation, ” *IEEE Cluster*, Sept. 2007.
7) H. Barada, S. M. Sait, and N. Baig, “Task matching and scheduling in heterogeneous systems using simulated evolution, ” *10th IEEE Heterogeneous Computing Workshop (HCW 2001)*, in the CD-ROM Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS 2001), paper HCW 15, Apr. 2001.
8) 株式会社ミック経済研究所. “データセンターの消費電力とグリーン IT 化の実態調査, ” http://journal.mycom.co.jp/news/2009/08/07/037/index.html.
9) 荒井 大輔，吉原 貴仁， 井戸上 彰, “データセンタ省電力運用管理手法の提案，”電子情報処理学会.
10) E. Pakbaznia and M. Pedram, “Minimizing data center cooling and server power costs,” *Proc. of International Symposium on Low Power Electronics and Design,* pp. 145-150, Aug. 2009.
11) Q. Tang, S. K. S. Gupta, and G. Varsamopoulos, “Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach, ” *IEEE Transactions on Parallel and Distributed Systems, vol. 19, no. 11,* pp. 1458-1472, 2008.