Vol.2014-MPS-99 No.10
2014/7/21

マルチコアプロセッサの動的周波数変更を考慮したタスクスケジューリング

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

概要：マルチコアプロセッサの処理性能を向上させるため，ターボブースト及びハイパースレッディングと呼ばれる処理高速化及び並列処理技術が利用されるようになった．本研究では，これらの技術による動作速度の動的変更とネットワークコンテンションを考慮したタスクスケジューリングアルゴリズムを提案する．実機実験を通して動作周波数モデルを作成し，これを利用してタスクのスケジュールを行う．実機での評価の結果，最大で 36% 処理時間を短縮できた．

1. 序論

近年，マルチコアプロセッサを搭載した計算機が広く利用されており，ターボブースト [1] と呼ばれる処理高速化技術が採用されている．これは，一部のコアのみが使用されている場合，使用中のコアの動作周波数を向上させる技術である．また，ハイパースレッディング [2] と呼ばれる技術も採用されており，これは一つの物理プロセッサコアで複数のスレッドを並行に実行することでハードウェア資源の効率的な使用を可能にしている．本稿では，ネットワークの帯域制約に加え，ターボブースト及びハイパースレッディングによる動作周波数の動的変更を考慮し，全体の処理時間を最小化するタスクスケジューリング手法を提案する．提案手法を評価するため，シミュレーションと実機実験を行った．提案手法は比較手法に比べシミュレーションで最大 43% ，実機実験で最大 36% 処理時間を短縮できることを確かめた．

2. 関連研究

リストスケジューリングは古典的なタスクスケジューリング手法であり，タスクを定められた優先度に従って、最も早く完了できるプロセッサに割り当てていくヒューリスティックな処理を行う．

Sinnen らは，ネットワーク帯域やネットワークコンテンションが生じない環境を仮定したリストスケジューリングを拡張し，ネットワークの遅延やコンテンションを考慮したスケジューリング手法の提案をしている [5]．実環境におけるネットワークの遅延やコンテンションの発生を考慮しつつ，タスクの処理時間が最小となるようにタスクのスケジュールを行い，同時にタスクの処理に必要なデータ転送に使用する通信経路のスケジュールも行なっている．

著者らの所属する研究グループでは，文献 [4] において，複数のマルチコアプロセッサからなる環境での単一ノード故障時における回復時間を最小化するタスクスケジューリングを提案している．この手法ではネットワークコンテンションを考慮しつつ，単一故障が発生した場合での全体の処理時間を最小化している．

3. ターボブースト及びハイパースレッディングのモデル化

ターボブースト及びハイパースレッディングを搭載したマルチコアプロセッサでは，動的に動作周波数が変更される．プロセッサの性能を発揮するためには，各プロセッサの使用状況に応じた動作周波数を推定する必要が生じる．両技術がどのように動作周波数を変化させるかは，一部公開されているものの，この情報から動作周波数を完全に求められるわけではない．本研究では，実機上で負荷を評価するためのプログラムを実行し，動作周波数を計測することにより，動作周波数モデルを作成する．

ターボブーストは，ダイの使用状況を監視し，処理状況に応じて動的に動作周波数を変更する．この技術は，ダイ全体の温度，供給される電流及び電力が仕様上設定されている限界と比べ余裕がある場合，アクティブとなっているプロセッサ数に応じて決められた動作周波数の上限まで自動で動作周波数を引き上げることでダイ全体の処理性能を

[1] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
[2] 会津大学
The University of Aizu

向上させる．ハイパースレッディングは，1 物理プロセッサを複数の論理プロセッサに見せ，1 物理プロセッサ上で複数の論理プロセッサを実行することで，各論理プロセッサに割り当てられたスレッドを同時に実行する．ハイパースレッディングにより，1 つの物理プロセッサ上で複数スレッドを動作させる場合，単一スレッドの場合に比べ，複数スレッドでリソースを共有しているため，各スレッドの実行速度は低下する．本研究では，この速度低下を動作周波数の低下としてモデル化する．各スレッドの実行速度の低下を加味した動作周波数を**実効動作周波数**と呼ぶ．

本研究では，ダイ上の全論理プロセッサの使用状況のみから，各論理プロセッサの実効動作周波数が一意に定まると仮定してモデルを構築した．

4. 問題定義

タスクスケジューリングへの入力は後述するタスクグラフとプロセッサグラフであり，出力は各タスクノードをプロセッサノードに割り当てたスケジュール結果である．本研究では，タスクグラフ全体の処理時間の最小化を目指す．

スケジュールされるプログラムを DAG によって表現し，タスクグラフ G と呼ぶ．タスクグラフのスケジュールとは，各タスクノードの処理開始時刻とプロセッサノードの関連付けである．タスクグラフ内の各頂点をタスクノードと呼び，各タスクノードは，1 つのプロセッサで逐次的に実行するための命令の集合を表す．タスクノード n の処理量を $C_{comp}(n)$ で示す．タスクノード間の有向枝をタスクリンクと呼び，ノード間の依存関係及び必要な通信を表す．タスクリンク e での通信に用いられるデータ量を $C_{comm}(e)$ と表す．タスクノード，タスクリンク全ての集合及び開始ノードをそれぞれ V, E 及び v_{start} とするとき，タスクグラフは $G = (V, E, v_{start}, C_{comp}, C_{comm})$ と表現される．

プロセッサグラフは，ネットワークトポロジを表現したものであり，各頂点をプロセッサノード，各辺をプロセッサリンクと呼ぶ．リンクを 2 つ以上持つプロセッサノードはネットワーク上のスイッチあるいはマルチコアプロセッサを搭載した計算機のネットワークインターフェイスであり，それらでタスクノードの処理はできないものとする．リンクを 1 つのみ持つものはマルチコアプロセッサ内のプロセッサを示している．リンクは，通信経路を表し，接続される 2 つのプロセッサ間は双方向に通信を行う事ができる．プロセッサノードの集合を P，プロセッサノード間のリンクの集合を R とする．3 章で述べたターボブースト・ハイパースレッディングのモデルを用いて，各状態において，ダイの使用率が状態 s の時の実効動作周波数を返す関数を $freq(s)$ とする．この時，プロセッサグラフを $N = (P, R, freq)$ と表す．

また，スイッチにおけるデータの転送処理時間はタスクの処理に比べて非常に短いため無視できるものとする．

本研究では，マルチコアプロセッサ内のプロセッサ間でのデータ転送に掛る時間はネットワークを介したデータ転送と比較すると小さく，0 とする．ネットワークを介した通信は方向を問わず同一リンク上で複数の転送は同時に行えないものとし，リンクの帯域は全て同じとする．マルチコアプロセッサの動作周波数モデルは 3 章での実験で得られたものを用いる．

本研究で扱うスケジューリング問題は Sinnen らのモデル [5] をベースにしており，以下の 3 つを制約条件とする．(1) 同じプロセッサノードに割り当てられた 2 つのタスクノードは，片方の処理が終了するまでもう一つの処理を開始できない．(2) 各タスクノードの処理は親ノードの処理が全て完了し，全ての親ノードからのデータ転送が完了するまで開始できない．(3) タスクノードの処理をスケジュールする際，その処理はプロセッサノードの空き時間内に終了しなければならない．これらは，文献 [5] の Condition1〜3 に該当する．

ネットワークリソースは有限であり，ネットワークコンテンションが発生するとする．そのため，ネットワークコンテンション回避のためネットワークに関して，Sinnen[5] らのネットワークコンテンションのモデルをマルチコアプロセッサへ対応させたモデルを用いる．制約条件は以下の 3 つである．(1) タスクノード間のデータ転送は同時に行えない．(2) データ転送の開始時刻を先行のデータ転送の開始時刻より前にすることはできない．(3) データ転送をスケジュールする時，利用するネットワークが使用されていない空き時間内にデータ転送が終了しなければならない．これらは，文献 [5] の Condition4〜6 に該当する．以下では上記の条件をマルチコアプロセッサ向けに拡張する．

本問題への入力としてタスクグラフ $G = (V, E, v_{start}, C_{comp}, C_{comm})$，プロセッサグラフ $N = (P, R, freq)$ を与える．出力は各タスクノードをプロセッサノードへ割り当てたスケジュール S である．本問題の目的関数は式 $minimize\,(lt(S))$ で示す．ただし，$lt(S)$ はスケジュールされた結果 S に対する最遅処理完了時刻を表す．

5. 提案手法

本研究で扱う問題は NP 困難に属する組み合わせ最適化問題であり，タスクグラフのサイズが大きくなると最適解を短時間で求めることは困難である．提案手法では，ネットワークの遅延及び競合を考慮するために，既存のネットワークコンテンションを考慮した手法をベースに拡張する．割り当て対象とするタスクノードの処理期間中のダイの使用状況を調べるために，並列に処理が行われるタスクノードを判定する必要がある．そのために，後続タスクノードを Sinnen らのネットワークコンテンションを考慮した手法を用いて空いているプロセッサノード群の中から最も早

く処理完了できるプロセッサノードへ仮割り当てを行う．その後，生成したスケジュール結果に対して，本研究で作成したターボブースト及びハイパースレッディングによる動作周波数の変更モデルを用い，両技術によるプロセッサノードの利用率に応じた動作周波数の変更による処理時間の変動及び，それに伴う処理開始時間等の調節を行う事でタスクノードの処理時間等を推定する．これをプロセッサノード全てに対して行い，本来割り当てたいタスクノードを各プロセッサノードに割り当てた場合の全体の処理時間を推定していく．

また，本手法では，タスクノードの処理時間の推定を全てのプロセッサノードに対して割り当てた場合を考えており，その中からスケジュールに採用するプロセッサノードへの割り当てパターンを選択する必要がある．選択の基準として，各パターン毎の仮割り当て時の結果から推定した全体の処理完了時刻を利用し，全てのパターンの中で最良と思われる割り当てパターンを採用する．その後，後続のタスクノードのスケジュールへ移行する．

全てのタスクノードに対してこの処理時間推定と割り当てるプロセッサの選択処理を繰り返し行うことで最終的なスケジュールを生成する．

5.1 古典的なリストスケジューリング

提案するタスクスケジューリングアルゴリズムは，古典的なリストスケジューリングアルゴリズムを拡張している．リストスケジューリングでは，先行制約及び問題に依存する優先度 (例：残りのスケジュール長の大きい順) に従ってタスクノードをプロセッサノードに割り当てる．タスクノードが割り当てられるプロセッサノードは，そのタスクノードを最も早く処理完了出来るプロセッサノードである．

5.2 ネットワーク特性の考慮

ネットワークの特性を考慮するスケジューリング手法では，通信帯域やスイッチでの遅延，転送するデータの競合などを考慮しつつ，各手法の目的を達成するためにタスクノードの割り当てを行う．提案手法のベースとなる Sinnen らの手法では，タスクノードを割り当てるプロセッサノードを選択する部分においてネットワークコンテンションを考慮している．

5.3 ターボブースト・ハイパースレッディングによる動作周波数の変更を考慮した提案アルゴリズム

タスクノードの処理時間を正確に推定するためには，実際にタスクノードが処理される際のターボブースト及びハイパースレッディングによる動作周波数を知る必要がある．しかし，リストスケジューリングではタスクノードに割り当てられた優先度に基づいてソートされたリストの先頭から順に割り当てを行うため，タスクノードを割り当てる時点では，後続のタスクノードの割り当てによって割り当てたプロセッサノードの使用率が変化してしまう可能性があり，正確な動作周波数を求めることができない．そこで，プロセッサノードの使用状況を把握するために，提案手法では Sinnen らのネットワークコンテンションを考慮した手法を用いて後続のタスクノードの仮割り当てを行い，割り当てたいタスクノードを処理する動作周波数を暫定的に決定し，タスクノードの処理時間を求める．提案手法では，ターボブースト及びハイパースレッディングの動作周波数変更モデルを使用し，両技術が動作している場合の処理時間の再調整を行う．その後，割り当てたプロセッサノードの中から最も処理完了時刻が早い結果と推定された割り当てを採用し，後続のタスクノードの割り当てを行う．この一連の動作を全てのタスクノードに対して行う事でターボブースト及びハイパースレッディングによる動作周波数の動的変更を考慮したうえで全体の処理時間を最小と出来るスケジュール結果を作成する．

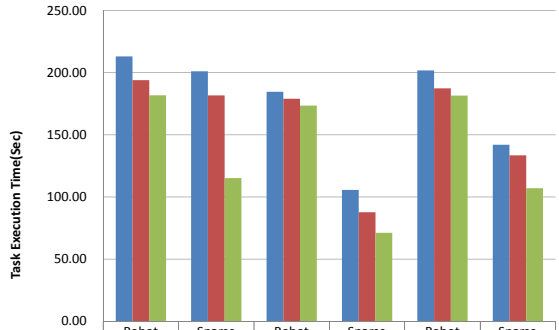
提案手法ではまず，既存のリストスケジューリングと同じように入力として与えられたタスクグラフ内のタスクノードを，暫定的に計算した最も早い実行開始時間の順にソートする．次に，タスクノードの処理時間を正確に推定するために，各プロセッサノードにネットワークコンテンションを考慮してタスクノード割り当てる．その後，各割り当てパターンに対して，処理完了時刻推定タスクスケジューリングを通してタスクノードを処理する際の動作周波数を暫定的に決定するために必要な後続のタスクノードの仮割り当てを行う．そして，仮スケジュールの結果に対して本研究で作成した動作周波数モデルを用いてプロセッサノードの使用状況に応じた動作周波数を決定し，全体の処理完了時刻の推定を行う．

6. 評価実験

処理時間の短縮率及び実機上での有効性と作成した動作周波数モデルの妥当性を評価するため，シミュレーションと実機実験による評価を行った．

既存手法は，ターボブースト及びハイパースレッディングによる動作周波数の動的変更を考慮を行っていない．本研究では比較手法として，既存手法に本研究で作成した動作周波数モデルを組み込んだ SinnenPhysical と SinnenLogical の 2 種類を用意し，それらと比較することでスケジューリングアルゴリズムによる処理時間の短縮率を評価する．

SinnenPhysical はネットワークコンテンションを考慮した Sinnen らの提案したスケジューリングアルゴリズムの割り当て対象を物理プロセッサに限定し，ターボブーストによる動作周波数の変更を考慮するため拡張したものである．ターボブーストによる動作周波数の変更を考慮するため，本研究で作成した動作周波数モデルをタスクノード



	Robot	Sparse	Robot	Sparse	Robot	Sparse
	Tree network		Fully connected		Star network	
SinnenPhysical	213.10	201.16	184.62	105.66	201.84	142.04
SinnenLogical	193.98	181.72	179.02	87.70	187.44	133.50
Proposed	181.80	115.19	173.50	71.10	181.60	107.00

図 1: タスクグラフの処理時間：シミュレーション

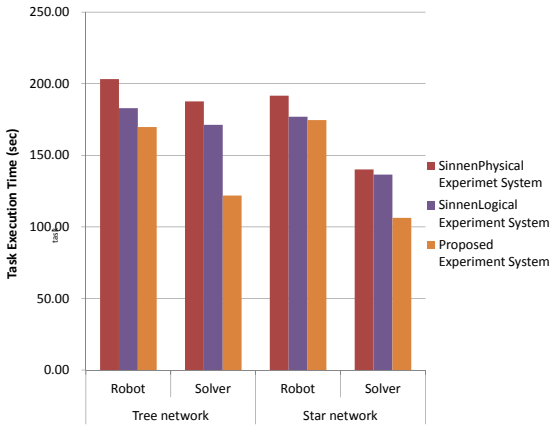


図 2: タスクグラフの実行時間：実機実験

を割り当てるプロセッサノードを決定する関数に組み込み，各タスクノードを割り当てる際にプロセッサノードの使用状況に応じて変化する動作周波数を考慮するように拡張した．**SinnenLogical** は，同様に Sinnen らの提案したスケジューリングアルゴリズムをターボブースト及びハイパースレッディングによる動作周波数の変更を考慮するために拡張したものであり，割り当て対象とするプロセッサが物理プロセッサだけでなく，論理プロセッサにも割り当てを行うことでハイパースレッディングによる動作周波数の変更についても考慮する．

6.1 実験設定

この実験で用いるタスクグラフとして Standard Task Graph Set[6] の Robot Control と Sparse Matrix Solver を用意した．スケジュールした結果を基に，タスクを実行する計算機システムとして，PC4 台を Gigabit Ethernet により接続した物を使用した．計算機のスペックは，CPU:Intel Core i7 3770T(2.5GHz，物理 4 プロセッサ，論理 8 プロセッサ，シングルソケット) である．プロセッサグラフとして，シミュレーションでは上記の計算機を想定し，ツリー型，メッシュ，スター型の 3 つのネットワークトポロジを構築した．

6.2 性能評価実験

提案手法と比較手法 2 種類に対してシミュレーション及び実際の計算機上でタスクノードを実行する実機実験を行いシミュレーションと実機実験のそれぞれでどの程度処理時間を短縮できたのか評価した．シミュレーションによるスケジュール結果の比較を図 1 に，実機実験の結果の比較を図 2 に示す．これらの結果から，提案手法は比較手法と比べシミュレーションでは最大でおよそ 43% ，実機実験の結果では最大で 36% の処理時間を短縮できることを確認した．

7. 結論

本論文では，ネットワークコンテンションが発生し，ターボブースト及びハイパースレッディングが搭載されているマルチコアプロセッサ環境を想定したタスクスケジューリングアルゴリズムを提案した．今後の課題としては，ターボブースト及びハイパースレッディングによる動作周波数モデルの妥当性の向上およびアルゴリズムの性能向上，異なるタスクグラフを用いた評価等があげられる．

参考文献

[1] Intel: “Intel Turbo Boost Technology in Intel Core Microarchitecture (Nehalem) Based Processors,” 入手先 ⟨http://download.intel.com/design/processor/applnots/320354.pdf⟩.

[2] DT. Marr, F. Binns, DL. Hill, G. Hinton, DA. Koufaty, JA. Miller and M. Upton: “Hyper-Threading Technology Architecture and Microarchitecture,” *Intel Technology Journal, Vol. 6, No. 1, pp. 4-15, 2002.*

[3] Anderson, J.H., Calandrino, J.M.: “Parallel task scheduling on multicore platforms, ” *ACM Special Interest Group on Embedded Systems(SIGBED) **3**(1), pp. 1–6, 2006.*

[4] S. Gotoda, M. Ito and N. Shibata: “Task scheduling algorithm for multicore processor system for minimizing recovery time in case of single node fault,” *International Symposium on Cluster, Cloud, and Grid Computing (CCGrid), pp. 260-267, 2012.*

[5] Sinnen, O., Sousa, L.: “Communication contention in task scheduling, ” *Parallel and Distributed Systems, IEEE Transactions on **16**(6), pp. 503–515, 2005.*

[6] Tobita, T., Kasahara, H.: “A standard task graph set for fair evaluation of multiprocessor scheduling algorithms, ” *Journal of Scheduling **5**(5), pp. 379–394, 2002.*