4H−5

実行タイミングの動的変動に強い静的スケジュール法「CP-DTSP」の特性評価

高木 浩光　　有田 隆也　　曽和 将容
名古屋工業大学 電気情報工学科

1.はじめに

並列計算機のアーキテクチャを実行順序制御の観点から分類すると、命令のプロセッサ割り当てを実行時に行なう動的順序制御方式と、実行前に行なう静的順序制御方式とに大別することができる。後者は前者に比較して、実行時にスケジュール処理を行なう必要がない、先行制御が容易であるなどの理由により、高速な順序制御が可能であるという特長をもっている。ただし静的順序制御方式では、その性能を最大限に引き出すために、実行前に最適な命令スケジュールを決定する必要がる。この最適化は、命令の処理時間の予測をもとに行なわれるが、この処理時間の予測が適確でない、もしくは、キャッシュ・ミスやネットワーク遅延などの不確定要素によって処理時間が実行時に変動するような場合、実行前の最適化では十分に良い性能が得られない場合がある。これが静的順序制御方式の動的順序制御方式に対する欠点のひとつとなっていた。我々は、従来のスケジュール法を改良することによって、処理時間の予測どおりに実行された場合の性能は従来のままに保ち、かつ処理時間が予測から変動した場合の性能を向上させるスケジュール法: DTSP(Dependent Tasks Same Processor)法を提案した[1]。本スケジュール法を用いることにより、従来の方法に比較して5 ～10% 程度の性能向上が得られることが示されている[1]。この性能向上率は、タスクグラフの形状、タスク（命令）の処理時間のバラツキ、プロセッサ数、タスク数などによって変化するものであった。特にタスクの処理時間のバラツキに対しては特徴的な変化がみられた。本稿では、この特徴について明らかにし、DTSP法がどのような環境において特に有効であるか考察する。

2.CP-DTSP 法

本稿で扱うスケジュール問題は、ノンプリエンプティブな決定性スケジュール問題として古くから様々な分野で研究されており[2),3)]、最適解を求めることは特別な場合を除いて強ＮＰ困難となることが知られている[4)]。これに対し近似解ではあるがタスク数の２乗程度のオーダで高速に求めることのできるヒューリスティック・アルゴリズムがいくつか知られている。このうちリスト・スケジューリングの一種であるクリティカル・パス(CP)法は比較的高い率で最適解を得ることができ[5)]、十分に実用的な近似解を得ることのできるものとして知られている。

リスト・スケジューリングは、タスクの処理時間予測にもとづいて実行タイミングのトレースを行いながら、空きプロセッサに実行可能タスクを割り当てていく方法である。空きプロセッサ数より実行可能タスク数の方が大きい場合には、タスクに適当な優先順位（CP法ではクリティカル・パス長を用いる）を設けて、空きプロセッサ数分だけのタスクを選択し割り当てる。このとき従来の方法では、複数の実行可能タスクを複数の空きプロセッサに割り当てる際に、どのタスクをどのプロセッサに割り当てるかが規定されておらず、実際にはタスク番号順などのランダムな因子によって決定されている。

このことは、実行タイミングが変動しない状況下では、全体の実行時間に何ら影響を与えない。しかし、実行タイミングが変動する場合には、処理時間予測が外れたことによる実行時間延長の度合いが、この割り当ての組合せによって異なってくる。DTSP法は、これをある適当な規則に基づいて決定することによって、実行タイミングの動的変動に強いスケジュールを得ようとするものである。DTSP法は、「先行制約の距離」（その先行制約によって順序付けられている２つのタスクの実行終了・開始時刻の差）の短いものから優先的に、その先行制約によって順序付けられている２つのタスクを、同一プロセッサに割り当てるものである。これによって、プロセッサ間をまたがる先行制約数が削減され、実行タイミングが変動した場合の待ち発生率を低く抑えられると期待される。DTSP法のアルゴリズムの詳細については文献1)を参照されたい。

CP-DTSP 法は、CP法をベースに、DTSP法による拡張を施したものである。同様に、その他のリスト・スケジューリング（例えばCP/MISF 法[6)]など）にDTSP法を組み合わせて拡張することも可能であるが、本稿ではその一例としてCP-DTSP 法を評価の対象とした。

3.評価と考察

評価は次の手順で行なった。まず、与えられたタスクグラフとタスクの予測処理時間をもとに、CP法によるスケジュールとCP-DTSP 法によるスケジュールを求める。次に、これらをある変動した処理時間のもとで実行した場合の全体の実行時間を計算する。そして、両者の比(CP-DTSP/CP)をもってDTSP法による性能向上率とする。この値は、タスクグラフの形状に強く依存すると考えられるので、この実験を300 のランダムに生成されたタスクグラフを用いて繰り返し、その平均値をとることによって評価の信頼性を高めている。

評価のパラメタには、処理時間が変動するタスク数と処理時間の「バラつきパターン」を用いた。スケジュール時の予測処理時間の決定はランダムに行ない、そのうちのいくつかのタスクの処理時間をランダムに変動させて実行時の処理時間とするのであるが、この変動するタスク数を0,5,10,....,70と変化させて性能向上率の変化を調べた（総タスク数は70に固定）。処理時間のバラつきパターンには、表１に示すA,B,C,D の４つのパターンを用いた。パターンA は、タスクの処理時間は1 ～15のいずれかの時間をとることを意味し、パターンD はすべてのタスクの処理時間が8 であることを意味している。スケジュール時の予測処理時間の決定に、これらそれぞれのパターンを用い、実行時の変動したタスクの処理時間はパターンA の中から選択するものとした。

この評価結果を図１に示す。パターンA ではほぼ、変動するタスク数が増大するにつれてDTSP法の効果が大きくなっているが、それ以外のパターンでは、頭打ちもしくは逆に変動タスク数が大きすぎると効果は小さくなる傾向にある。このことは、

Performance Evaluation of Static Scheduling Method "CP-DTSP" Resistive to Dynamic Fluctuation of Execution Timing

Hiromitsu TAKAGi, Takaya ARITA, Masahiro SOWA
Nagoya Institute of Technology

変動するタスクが多すぎると、実行タイミングのずれが互いのタスクによって相殺されてしまうからであると考えられる。

また、パターンD,C,B,A の順にDTSP法の効果が大きいということが読み取られる。この原因は次のように推察できる。パターンD ではすべてのタスクの処理時間が等しいため、スケジュール時の実行トレースではすべてのプロセッサが同時に実行を終了する。したがって、すべてのプロセッサに同時に実行可能タスクを割り当てる。これに対し、パターンA では、処理時間が分散しているため、同時に終了するプロセッサ数は少なくなる。したがって、少数のプロセッサにしか同時にタスクを割り当てない。このことは、どのタスクをどのプロセッサに割り当てるかの組合せ数が少なくなることを意味し、DTSP法によるスケジュールの自由度が狭められていることを意味する。このことを裏付けるために、スケジュール時の同時に割り当てるタスク数の平均を、各パターンごとに求めてみた。この結果を図2に示す（図中の値は、プロセッサ数で除算されている）。パターンD,C,B,A の順に大きな値となっていることが確認される。

この評価結果から次のことがいえる。一般に粗粒度の並列処理では、タスクの処理時間のバラつきは大きい（例えば、2つのタスクの処理時間比が517:298 であるといったように）。このような環境下では、同時に割り当てられるタスク数はきわめて小さく、DTSP法の効果はあまり期待できないと考えられる。これに対し、細粒度の並列処理、特に命令単位の並列処理では、処理時間は1 ～8 程度の小さなバラつきにおさまっている。特にRISC型プロセッサでは、ほとんどの命令の実行が1サイクルで行なわれるなどこの傾向は顕著なものとなる。このような環境下では、パターンD の例で示したように、スケジュールの自由度が大きくなり、DTSP法の性能改善効果は大きいと考えられる。ただ、粗粒度の並列処理においても、タスクの処理時間がまったく予測不能であるような場合、すべてのタスクの処理時間を等しいと仮定してスケジュールを行なう場合があり、そのような場合にはDTSP法は有効なものとなる。また、DTSP法の効果を上げるために故意にタスクの処理時間を離散化することにも効果があると考えられる。

4.おわりに

実行タイミングの動的変動に強い静的スケジュール法であるCP-DTSP 法について、その性能評価を、変動するタスク数、タスクの処理時間のバラつきパターンをパラメタとして行なった。その結果、処理時間のバラつきが小さいほどDTSP法の効果が大きいということが明らかになった。このことから、一般に処理時間のバラつきが小さい、細粒度の並列処理で特にDTSP法が有効となることが示された。この傾向は、CP/MISF 法をベースとしたCP/MISF-DTSP法[1]についても同様であった。

本スケジュール法は、データの通信時間についてはすべて均一であるとみなしているが、マルチポートのレジスタファイルを共有して動作するような、静的なスーパースカラ・プロセッサ（例えばPNプロセッサ[7]）などでは、そのまま実用的なスケジュール法として用いることができる。この場合、命令レベルの並列処理であるため処理時間のバラつきは小さく、DTSP法の効果は十分期待できるものとなるであろう。

参 考 文 献

1) 高木，有田，曽和，" 実行タイミングの動的変動に強い静的プロセッサ・スケジューリング・アルゴリズム",信学技報，CPSY90-83 (1990).

2) Hu,T.C.,"Parallel Sequencing and Assembly Line Problems", Oper.Res., vol.9, pp.841-848 (1961).

3) Coffman,E.G. and Graham,R.L.,"Computer and Job-shop Scheduling Theory", John Wiley & Sons (1976).


```
A (1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15)
B (  2,    4,    6,    8,   10,   12,   14   )
C (       4,          8,         12,         )
D (                   8                      )
```

表1．タスク処理時間のバラつきパターン

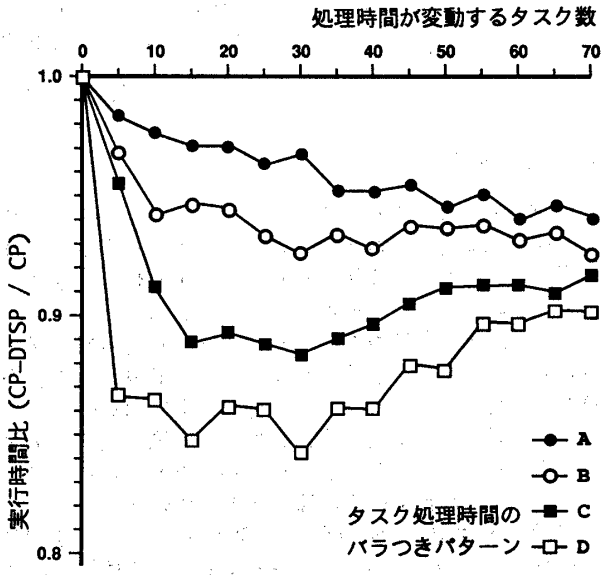

図1．DTSP法による性能向上率

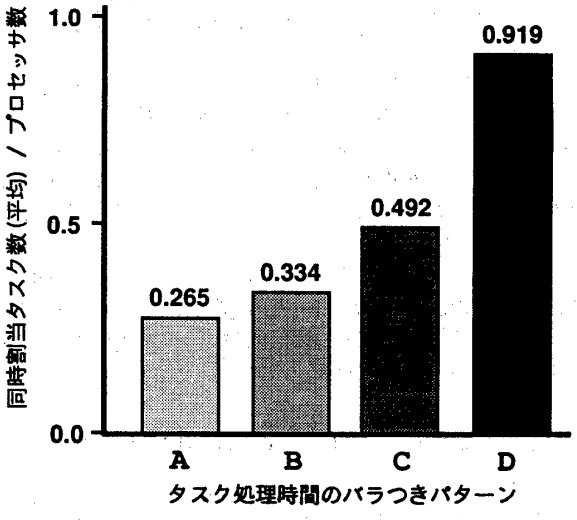

図2．タスク配置の自由度

4) Garey,M.R.and Johnson,D.S.,"Computers and Intractability, A Guide to the Theory of NP-Completeness", Freeman, San Fransisco (1979).

5) Adam, Chandy and Dickson, "A Comparison of List Schedules for Parallel Processing Systems", Commum. ACM, vol.17, pp.685-690 (1974).

6) Kahahara,H. and Narita,S., "Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing",IEEE Trans.Comput., vol.C33, pp.1023-1029 (1984).

7) 曽和，有田，河村，高木,"機能分割型プロセッサによる複数命令流実行方式",信学論文誌D-I, vol.J73-D-I, No.3, pp.280-285 (1990).