設計自動化 68－6
(1993. 10. 28)

デザインルールチェック並列処理化の一手法 — 並列スケジューリングによる手順割り当て —

河野 一郎　小野寺 秀俊　田丸 啓吉

京都大学工学部電子工学科
〒606 京都市左京区吉田本町
TEL: 075-753-5313

あらまし

本稿では、単一バス結合されたマルチプロセッサシステム上で、デザインルールチェック (以下 DRC) の検証手順を並列化するスケジューリング問題を提起する。この問題では、通信路で接続された EWS(Engineering Work Station) をマルチプロセッサシステムのモデルとしている。この問題の一解決法として、検証手順をタスクグラフで表現し、そのタスクをレベル分けして、同一レベルのタスクごとにプロセッサに割り当てる操作を繰り返すという手法を提案する。本手法の応用例として、ランダムタスクグラフと実 DRC に適用し、有用性の評価を行なった。

和文キーワード　デザインルールチェック, マルチプロセッサ, 並列化, スケジューリング

Parallel Processing of Design Rule Checking with A Multiprocessor Scheduling

Ichiro KOHNO　Hidetoshi ONODERA　Keikichi TAMARU

Faculty of Engineering, Kyoto University,
Sakyou-ku Kyoto
TEL: 075-753-5313

Abstract

In this paper, we propose a multiprocessor scheduling problem for parallel processing of design rule checking(DRC). An example of the multiprocessor system under consideration is a set of engineering work stations(EWS) connected by a network. We propose a method that considers the level of each task and repeats assignment of the tasks at the same level onto processors. We apply this method to randomly generated task graphs as well as task graphs derived from actual DRC. Experimental results demonstrate effectiveness of the proposed method.

英文 key words　design rule checking, multiprocessor, parallel processing, multiprocessor scheduling

1 まえがき

マルチプロセッサシステムの能力を十分に引き出すためには、対象とする処理を構成するタスク集合を各プロセッサに割り当て、各プロセッサ上での実行順序を決定することによって、処理全体の完了時間を最小化することが重要な問題となる。しかしこのようなマルチプロセッサスケジューリング問題は、そのほとんどがNP困難であることが知られており[1]、一般に最適解を求めることは極めて難しい。そのため従来からヒューリスティックによって近似解を求める手法が提案されてきた[2]~[5]。

通信路で結合されたEWS(Engineering Work Station)システムのような、バス結合されたマルチプロセッサシステムでは、多数のプロセッサを接続可能であり、またシステムの構築および変更が比較的容易であるため広く利用されてきている。しかしシステムが柔軟性に富む反面、処理に並行した通信が困難であるため、通信コストや通信路競合などの通信オーバヘッドが全体の処理時間に大きな影響を及ぼす。

これまでに各プロセッサが共有メモリを持つバス結合マルチプロセッサシステムを対象としたスケジューリング手法が提案されている[4]。このような共有メモリ型のシステムでは、処理に並行して通信を行なうことが可能である。そのため、処理に並行した通信が困難なシステムに対してこれらの手法を適用した場合には通信オーバヘッドが大きくなることが予想される。

本稿では、処理に並行した通信が困難である単一バス結合マルチプロセッサシステムにおいて、より通信オーバヘッドの削減に重点をおいたスケジューリング手法を提案する。また提案手法により、LSI設計の検証工程の1つであるデザインルールチェック(以下DRC)のスケジューリングを行なう。

本稿では、まず**2.**で対象としているマルチプロセッサシステムおよびスケジューリング問題を定義する。**3.**では、スケジューリング問題と並列DRCとの対応について述べる。**4.**では、通信オーバヘッドの最小化を図るスケジューリング手法を提案する。**5.**では、提案するスケジューリング手法をランダムタスクグラフとDRCに適用し、性能評価を行なう。

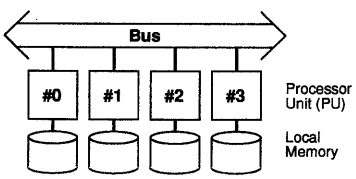

図1: 対象マルチプロセッサシステム

2 データ転送時間を考慮したスケジューリング問題

2.1 対象とするシステム

対象とするマルチプロセッサシステムは、図1に示すように各々のプロセッサを単一バスで結合したシステムである。本システムでは、プロセッサ間のデータ転送はすべてバスを介した1対1通信によって行われる。各プロセッサは記憶装置を持っており、処理の結果得られたデータおよび他のプロセッサから転送されたデータは全体の処理が完了するまで保持される。2台のプロセッサ間のデータ転送中にはバスが一括して占有され、他のプロセッサ間は通信を行なうことができない。また、データの授受を行なっているプロセッサは、並行して処理を行えないものとする。このシステムは、EWSを通信路で結合したマルチプロセッサシステムをモデルとしている。

2.2 データ依存のあるタスク集合の表現形式

処理すべきタスク集合のデータ依存関係は、非サイクル有向グラフ(DAG)で表される。図2に示すように各ノードがタスクを表し、ノード内の数字はタスク番号、左側の数字は実行時間、またタスク間のエッジはデータ依存を表している。各エッジにはバスを介したデータ転送時間が付加されている。以下では図2に示したグラフをタスクグラフと呼ぶ。

図2の例では、タスク5はタスク処理時間20[単位時間]を持ち、タスク3からのデータ転送時間10を要するデータと、タスク4からのデータ転送時間20を要するデータを実行開始時に必要とすることを表している。またタスク5で使用あるいは定義したデータ(データ転送時間10)を、タスク6およびタスク7に送る必要がある

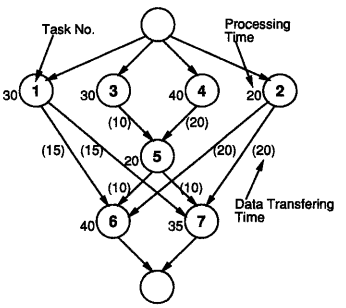

図2: タスクグラフ

ことを示している。あるタスクの実行に必要なデータを供給するタスクをそのタスクの先行タスクと呼ぶ。また、逆に先行タスクから見た当該タスクを後続タスクと呼ぶ。

先行制約 (データ依存) のある 2 つのタスクが異なるプロセッサに割り当てられた場合には、先行タスクを実行したプロセッサから後続タスクを実行するプロセッサにデータを転送するので、エッジに記されたデータ転送時間が必要となる。一方、それらのタスクが同一プロセッサに割り当てられた場合には、データの授受はプロセッサの記憶装置を介して行なわれるものとし、データ転送時間はゼロとする。

2.3 問題の定義

本稿で取り扱うスケジューリング問題は、タスク間にデータ依存に起因する先行制約が存在する n 個のタスクを、バス結合された m 台のプロセッサ上で処理する際に、データ転送を考慮して処理全体の完了時間を最小にするスケジュールを求める問題である。ここではすべてのプロセッサは同じ性能を持ち、タスク処理速度およびデータ転送速度について等しいものとする。また、どのタスクについても一度処理を開始すると中断することができないものとする。

2.4 従来の研究

本研究で取り扱う問題と類似した問題として、各プロセッサが共有メモリを持ち、タスク処理に並行したデータ転送が可能なバス結合マルチプロセッサシステムを対象としたスケジューリング問題が研究されている [3],[4]。本稿ではタスク処理に並行したデータ転送が不可能なシステムを対象としており、先に対象とされたシステムに比べ通信オーバーヘッドの影響が大きくなる。

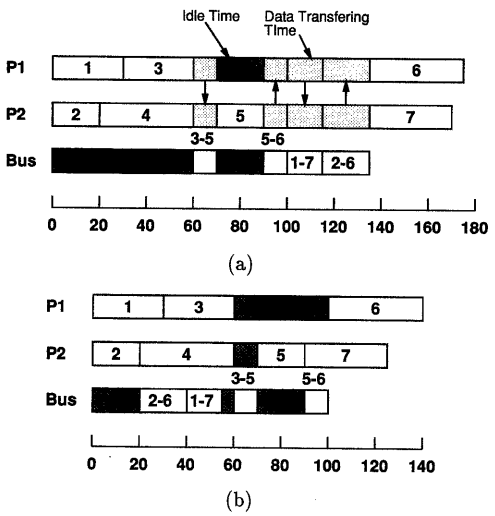

図 3: システムの相違によるスケジュールへの影響

対象とするシステムの相違がスケジュールに及ぼす影響について説明する。図 3(a) は、タスク処理に並行したデータ転送を許さない 2 台のプロセッサ上で図 2のタスクグラフを実行する場合のスケジュールの例を示したものである。図 3の (a) は上から順に、プロセッサ 1, 2 でのタスクの実行状況およびデータ転送状況、データ転送によるバスの使用状況を示したものである。図中ではタスク 7 はプロセッサ 2 の時刻 135 に割り当てられている。タスク 7 の先行タスクはプロセッサ 1 に割り当てられているタスク 1 とプロセッサ 2 に割り当てられているタスク 2, 4 である。そのためプロセッサ 1 から 2 へ時刻 100 から 115 の間にタスク 1 のデータ転送 (Bus の欄の 1–7) を行なっている。図 3(b) は、2 台のプロセッサがタスク処理に並行してデータ転送できる場合のスケジュールを示したものである。(a) の場合と同様にタスク 7 にはタスク 1 からデータ転送が行なわれるが、時刻 30 のタスク 3 の実行と並行してタスク 1 のデータ転送 (時刻 40 から 55) が行なわれている。図 3で示すように、タスク処理に並行したデータ転送を許さないときには許したときに比べて全体の処理時間が 140 から 175 と大きくなっている。

3 スケジューリングによる並列 DRC

3.1 DRC について

LSI の設計情報は、最終的にマスク製作のための幾何学的なパターンデータに集約される。DRC とは、LSI 設計の最終結果であるマスクパターンに対する設計検証であり、製造工程の限界からくるマスクパターン上の幾何学的な設計規則 (最小間隔、最小幅など) について検証を行なうものである。

LSI 製造工程において設計ミスのあるマスクパターンを使用することは、マスク再試作に直結し、開発期間の延長とコストの増大を招くことになる。そのため LSI 製造の前には、DRC によってすべての設計ミスを取り除く必要がある。

3.2 DRC の検証手順

DRC の対象となるパターンデータは、多層のパターンデータ (レイヤ) から成っている。DRC では各層のマスクパターンを単位として検証を行なうことが一般的である。また、与えられたレイヤに対してそのまま検証を行なうだけでなく、与えられたレイヤから別の新たなレ

イヤを作り出した後に検証を行なうことが可能である。

DRC の手順について説明する。DRC の基本となる手順は、まず与えられたレイヤから検証する部分のレイヤを作り、その後に設計規則を満たすか検証するというものである。図 4(a) に CMOS 回路の一部のレイアウトを示す。このレイアウトには、チャネルからのゲートのはみ出し長 (図中 x) とフィールド酸化膜のはみ出し長 (図中 y) についての設計規則が存在する。

この設計規則の検証手順は図 5に示すような形式で記述される。1~5 行目までは与えられたレイヤから検証する部分のレイヤを作る操作である。1, 2, 5 行目では処理するレイヤのパターン図形の和集合 (OR) となるレイヤを作り、3, 4 行目では積集合 (AND) となるレイヤを得ている。例えば、3 行目でゲート層と n+ 拡散層の重なりを ntrch というレイヤとして作り、4 行目でゲート層と p+ 拡散層の重なりを ptrch というレイヤとして作っている。さらに 5 行目で ntrch と ptrch を合わせて channel というレイヤとして定義している。次に 6, 7 行目は作り出したレイヤを用いて設計規則の検証を行なう操作である。図 4(b) は、(a) の n 型トランジスタの部分を拡大し、6 行目の操作について示したものである。6 行目では channel を拡大したものと fl の AND を取ったもの (図 4(b) の太線部) から、ゲートのはみ出し長 x について検証を行なっている。

上述のように DRC の検証手順はレイヤを入出力するオペレーション群から構成され、一般にオペレーション間にはレイヤ (データ) 依存がある。そのため DRC の検証手順はタスクグラフによって表すことが可能である。図 5の検証手順をタスクグラフで表したものが図 2

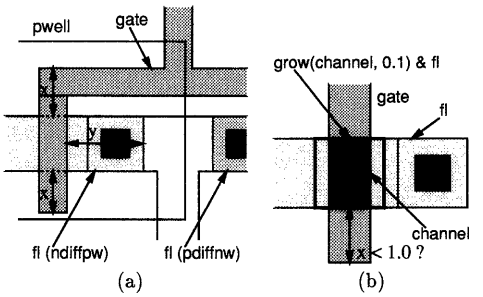

図 4: 設計規則の例

```
1. fl = nfl | pfl;
2. gate = g1ntr | g1ptr | g1sig;
3. ntrch = g1ntr & ndiffpw;
4. ptrch = g1ptr & pdiffnw;
5. channel = ntrch | ptrch;
6. drc(gate, grow(channel, 0.1) & fl, ext < 1.0);
7. drc(fl, grow(channel, 0.1) & gate, ext < 2.0);
```

図 5: 検証手順の記述

であり、検証手順の番号 (行数) がタスク番号に対応している。

3.3 DRC のスケジューリング問題

DRC では一度に行なう検証手順のオペレーション数は数百程度になる。大部分のオペレーションは単層または 2 層のレイヤに対して行なわれるため、先行するオペレーション数の平均は 1 以上 2 以下であると考えられる。また DRC 処理時間についてのシミュレーション [6] によれば、オペレーション実行時間に対するデータ転送時間の比は 10% 程度であると考えられる。したがって DRC の検証手順を表すタスクグラフに対してスケジューリングを行なうことにより、DRC をマルチプロセッサ上での並列処理に展開することが可能である。

近年の LSI の超集積化にともない、大規模なマスクパターンに対して精密な検証を行なう DRC には膨大な計算時間が費やされてきた。そのため DRC を高速処理する必要性が高まり、従来から DRC の並列処理化が提案されてきた。本稿ではスケジューリングによって DRC を並列処理する場合の処理時間について **5.2** で検討する。

4 データ転送時間を考慮したスケジューリング手法

複数のプロセッサに処理すべきタスク集合を割り当て、処理時間の最小化を図るマルチプロセッサスケジューリング問題は、特殊な場合を除いてそのほとんどが NP 困難な問題であることが知られている。このような問題においては、実用的なヒューリスティックアルゴリズムの開発が重要である。

本章では、データ転送時間およびプロセッサの空き時間の最小化を図るヒューリスティックなスケジューリング手法を提案する。

4.1 従来のスケジューリング手法

バス結合されたマルチプロセッサシステムを対象としたスケジューリング手法は、従来から数多く提案されている。その中で、各プロセッサが共有メモリを持ち、処理に並行した通信が可能であるマルチプロセッサシステムを対象とした実用的なスケジューリング手法として CP/DT/MISF 法 [4] が挙げられる。この手法ではタスク数が 40~100、タスク実行時間の総和とデータ転送時間の総和との比が 10~150% の範囲のスケジューリング問題を対象としている。

本稿の対象とするシステムは処理に並行した通信が困難であるバス結合マルチプロセッサシステムである。

CP/DT/MISF 法では割り当て時に選択される個々のタスクの局所的なデータ転送時間に注目しているため、本稿で対象とするシステムについて適用した場合には全体のデータ転送時間が大きくなると考えられる。

そのため、処理に並行した通信が困難であるバス結合マルチプロセッサシステムについては、よりデータ転送時間に重点をおいたスケジューリング手法が必要である。

4.2 スケジューリングの方針

全体の処理時間を構成するものは、(i) タスク実行時間、(ii) データ転送時間、(iii) プロセッサの空き時間の 3 つである。このうち (i) は処理すべきタスクが与えられると決定される。したがって全体の処理時間の最小化を図るためには、(ii) および (iii) の最小化を図ることが必要である。

スケジューリングは、タスクの実行時刻およびデータの転送時刻を考慮してタスクをプロセッサに割り当てる操作である。この操作によって処理全体の完了に要する時間を最小化することは、前述のように NP 困難の問題である。そのため、プロセッサにタスクを割り当てる操作と、タスク実行およびデータ転送の時刻を決める操作に分けてスケジューリングを行なう。

(ii) はタスクをプロセッサに割り当てる際に決まる量であり、どのタスクをどのプロセッサに割り当てるかに依存する。(iii) はタスクの実行時刻およびタスク実行に必要なデータの転送時刻などの実行タイミングを決めると定まる量である。そのため、タスク割り当て時に正確に推定することが困難である。(ii) の最小化のみを考慮してタスクを割り当てると、タスクが特定のプロセッサに偏って割り当てられ、(iii) が大きくなる可能性がある。そこで (iii) を考慮してタスクを割り当てるために、(iv) 各プロセッサの処理時間を均等にすることが必要である。

本稿では (ii)~(iv) を考慮したスケジューリング手法を提案する。以下では提案手法を TALS(Task Assignment with Level Sort) 法と呼ぶ。

4.3 アルゴリズム

本稿で提案するアルゴリズムは、タスクのデータ依存を表すタスクグラフをもとに各タスクのレベルを求め、同一レベルに属するタスク集合ごとにプロセッサに割り当てる操作を基本とし、これをレベルの小さい順から繰り返すというものである。但しここでいうレベルとは、タスクグラフの先頭タスクから当該タスクまでの経路の最大タスク数である。

アルゴリズムの概要について説明する。まず各タスクについてレベルを計算し、タスク集合を各レベルごとの集合に分ける。

次に以下の処理を、分割した各レベルのタスク集合ごとにレベルの小さい順に実行する。

同一レベルのタスク集合に含まれるすべてのタスクをプロセッサに割り当てる。このようにすると、注目しているレベルより小さいレベルのタスクについての割り当てがすべて完了していることになり、注目するレベルのタスクを割り当てるときにはタスク実行に必要なデータがすべて揃っていることになる。プロセッサへのタスクの割り当ては (ii), (iv) を考慮して行なう。すなわち、注目するレベルの前のレベルまでの処理時間の小さいプロセッサほど、多くのタスクを割り当てる。この際に、注目するレベルのタスク全体の実行に必要なデータの転送時間の最小化を考慮する。最後に、注目するレベルのタスク実行に必要なデータの転送時刻を決定する。これは (iii) を考慮して転送可能時刻の早い順に決める。その後でタスクを逐次実行する。

図 6を例に挙げ、TALS 法のアルゴリズムを説明する。図 6(b) は、(a) で示されるタスクグラフに対してスケジューリングを行なった結果である。以下にアルゴリズム中で用いられる記号の定義を示す。

T_i :	タスク i	S_l :	レベル l のタスク集合
P_j :	プロセッサ j	n_l :	レベル l のタスク数
m :	プロセッサ台数		

(1) 各タスク i のレベル l_i を求める。

(2) レベル l の小さいタスク集合 S_l から順に、実行に必要なデータが揃ったタスク (レディタスク) の集合とし、(3)~(6) を繰り返す。

[例 1] 図 6(a) に示すようにレベル 1 のタスク集合 S_1 は T_1, T_2, T_3 である。他のレベルのタスク集合についても同様である。

(3) 各プロセッサに割り当てられるタスク数を決定する。

まず次の規則でプロセッサの順番を決定する。

(3-1) 処理完了時刻の早い順
(3-2) (3-1) で決まらないときは、後続タスク数の総計の多い順

順番の若い q 台のプロセッサとその他のプロセッサについて、次式によって割り当てられるタスク数を決定する。

$$\text{順番の若い } q \text{ 台} = \lceil n_l/m \rceil + 1$$
$$\text{残りの } m-q \text{ 台} = \lceil n_l/m \rceil \quad (q = n_l \bmod m)$$

[例 2] 図 6(b) において、S_2 のタスクまでの割り当てが終了しており (図中の点線)、次に S_3 のタスクを割り当てるものとする。このとき P_1~P_3 の処理時間は順に 60, 40, 70 である。また S_3 のタスク数 n_3 は 4 である。したがって P_1~P_3 に割り当てられるタスク数は順に 1, 2, 1 となる。

(4) 各プロセッサのタスク要求リストを作成する。

各プロセッサ P_j について、レディタスク割り当て時の優先順位を表すリストを作成する。次の規則によってタスクの優先順位を決める。

(4-1) $T_i \in S_l$ の先行タスク集合が割り当てられたプロセッサの内、P_j を除くプロセッサの数を $g_{i,j}$ とする。また T_i の実行に必要なデータの内、P_j が保持しないデータの転送時間の和を $t_{i,j}$ とする。このとき、転送時間積 $f_{i,j} = g_{i,j} * t_{i,j}$ の小さい順

(4-2) (4-1) で決まらないときには、$g_{i,j}$ の小さい順

(4-3) (4-1),(4-2) で決まらないときには、T_i の後続タスク数の多い順

[例 3] 図 6(c) は P_1~P_3 のタスク要求リストを示したものである。図中の表は、上にあるタスクほど優先順位が高いことを表している。また表の右の () 内の数字は転送時間積 $f_{i,j}$ である。ここでは、P_2 について T_8, T_7 の転送時間積 $f_{i,2}$ の計算を示す。T_8 が P_2 に割り当てられるとすると、T_8 の先行タスク T_5 のデータ (データ転送時間 10) は P_3 にあるため、$f = 10$ となる。また、T_7 については先行タスク T_4, T_5 のデータがそれぞれ P_1, P_3 にあるため、$g = 2, t = 20$ となり $f = 40$ となる。

(5) タスク要求リストをもとにタスクを各プロセッサに割り当てる。

(5-1) 割り当て済みとなったタスク数が割り当てタスク数に満たず、かつ最小であるプロセッサ P_j をタスク割り当ての対象とする。

(5-2) 割り当て対象のプロセッサの要求リストにおいて最初に示されたタスクが互いに重複しないとき、各々の対象のプロセッサにリストの最初に示されたタスクを割り当てる。

(5-3) 要求リストの最初に示されたタスクが重複するとき、対象のプロセッサの中から次の規則によってそのタスクを割り当てるプロセッサを決定する。

(5-3-1) 重複するタスク T_i についての転送時間積 $f_{i,j}$ が小さいプロセッサ

(5-3-2) (5-3-1) で決まらないときには、$g_{i,j}$ の少ないプロセッサ

[例 4] 図 6(c) に示すように P_1~P_3 の要求するタスクが重複していないので、まずそれぞれに T_6, T_8, T_9 を割り当てる。さらに P_2 の割り当てタスク数が 2 であるので、P_2 に残りの T_7 を割り当てる。

(6) 割り当てられたタスクの実行時刻、およびタスク実行に必要なデータ転送処理の時刻を決定する。

まず (5) で割り当てられたすべての T_i について実行に必要なデータの転送時刻を決定する。このとき T_i の実行に必要なデータの内、既に T_i を割り当てたプロセッサに保持されているデータについては転送を行なう必要はない。データ転送の順序は次の規則によって決定する。

(6-1) データ転送可能時刻の早い順

(6-2) (6-1) で決まらないときには、データ転送時間の小さい順

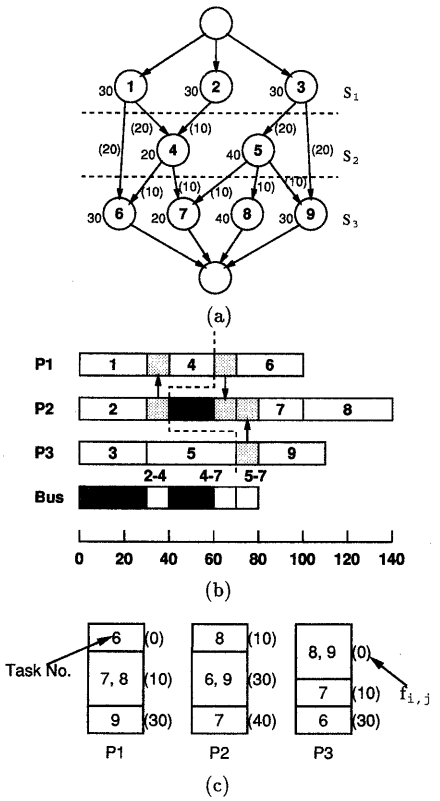

図 6: TALS 法によるスケジューリング

その後でT_iを逐次実行する。

[例5] P_1に割り当てられたタスクはT_6であり、P_2についてはははT_7, T_8、P_3についてはT_9である。図6(b)に示すように、T_7にはT_4, T_5のデータが必要であり、T_8にはT_5のデータが必要である。T_6, T_9の実行に必要なデータはそれぞれの割り当てられたプロセッサに保持されているため転送の必要はない。T_4からT_7へのデータ転送可能時刻は60であり、T_5からT_7へのデータ転送可能時刻は70である。したがってT_4からT_7へのデータ転送を先に行なう。

5 アルゴリズムの評価

本章では、ランダムに生成したタスクグラフ、および実DRCの検証手順から生成したタスクグラフについてTALS法を適用し、性能評価を行なう。

5.1 ランダムタスクグラフによる評価

ここでは乱数を用いてタスクグラフを生成し、TALS法を適用したときの性能を評価する。

[ランダムタスクグラフについて]

本稿で使用したタスクグラフは、タスク間の先行制約を一様乱数によって生成し、さらにタスクの処理時間、各先行制約のエッジのデータ転送時間を正規乱数を用いて生成したものである。ランダムタスクグラフを生成するために与えるパラメータは、タスク数、平均先行タスク数とその標準偏差、タスク実行時間の平均と標準偏差、データ転送時間の平均と標準偏差である。今回の評価では、タスク数nを50, 100, 200、平均先行タスク数ρを1.0, 2.0, 3.0、タスク実行時間の平均に対するデータ転送時間の平均の比λを10%, 50%, 100%としてランダムタスクグラフを生成した。

[評価]

タスク処理に並行したデータ転送が困難であるバス結合マルチプロセッサシステムについて、提案手法とCP/DT/MISF法をそれぞれ適用したときの性能評価を行なった。但しCP/DT/MISF法では、割り当てられたタスクの実行に必要なデータ転送をタスク実行の前に挿入するものとして評価を行なった。

プロセッサ台数2, 4としたときのタスク数とタスクグラフ処理時間の関係を表1に示す。タスクグラフ処理時間は、プロセッサ1台のときの処理時間を1としたときの相対的な並列処理時間の平均で表している。ρ, λについてはそれぞれ2.0, 50%としている。表1から、すべてのnについてTALS法の処理時間が小さくなることがわかる。またプロセッサ台数が多くいほど処理時間が短縮される。

先行タスク数とタスクグラフ処理時間の関係を表2に示す。n, λについてはそれぞれ100, 50%としている。データ転送時間とタスクグラフ処理時間の関係を表3に示す。n, ρについてはそれぞれ100, 2.0としている。この結果からρおよびλを変化させた場合についてもTALS法の処理時間が小さくなることがわかる。

表1: タスク数とタスクグラフ処理時間の関係

プロセッサ台数m	スケジューリング手法	タスク数n		
		50	100	200
2	TALS	0.707	0.685	0.672
	CP/DT/MISF	0.881	0.883	0.879
4	TALS	0.518	0.482	0.454
	CP/DT/MISF	0.755	0.730	0.729

表2: 平均先行タスク数とタスクグラフ処理時間の関係

プロセッサ台数m	スケジューリング手法	平均先行タスク数ρ		
		1.0	2.0	3.0
2	TALS	0.600	0.685	0.764
	CP/DT/MISF	0.746	0.883	0.985
4	TALS	0.358	0.482	0.599
	CP/DT/MISF	0.534	0.730	0.975

表3: データ転送時間とタスクグラフ処理時間の関係

プロセッサ台数m	スケジューリング手法	データ転送時間の比λ		
		10%	50%	100%
2	TALS	0.567	0.685	0.834
	CP/DT/MISF	0.690	0.883	1.125
4	TALS	0.333	0.482	0.684
	CP/DT/MISF	0.494	0.730	1.060

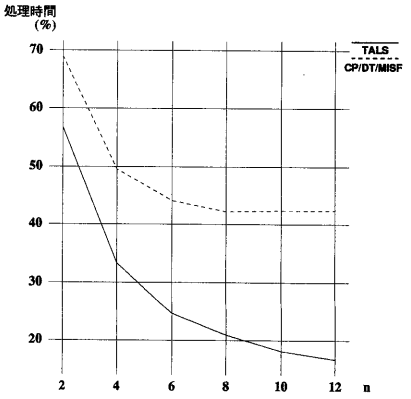

図7: プロセッサ台数とタスクグラフ処理時間の関係

また、ρが3.0のときおよびλが100%のときには、タスク間のデータ転送量が増加するため並列処理の効率が著しく低下している。

次にプロセッサ台数とタスクグラフ処理時間の関係を図7に示す。ここではn, ρ, λをそれぞれ100, 2.0, 10%としたランダムタスクグラフに対して評価を行なった。図7より、両手法ともnが大きくなるにつれて処理時間が減少することがわかる。またすべてのnについてTALS法の処理時間が小さく、nの増加に対する処理時間の減少率についてもTALS法の方が大きくなっている。さらにCP/DT/MISF法ではnが8のときに処理時間の減少が飽和しており、このときの処理時間は42%である。TALS法でも処理時間の減少が飽和する傾向にあるが、nが10のときに18%という処理時間が得られている。

以上により、タスク処理に並行したデータ転送が困難であるバス結合マルチプロセッサシステムを対象とするスケジューリング手法としてTALS法が有効であることが示された。

5.2 DRCによる評価

本節ではDRCにTALS手法を適用し、並列処理の効率の面から評価を行なう。

[DRCの検証手順について]

ここで取り上げるDRCは、繰り返しのない178の検証手順から構成され、1つの検証手順の先行制約となる手順の平均は1.64である。

[評価]

DRCの各検証手順のデータ依存関係は既知であるので、各タスクの実行時間およびデータ転送時間を決定してタスクグラフを生成することで評価を行なう。ここではλを10%とし、正規乱数を用いてタスク実行時間およびデータ転送時間を生成した。

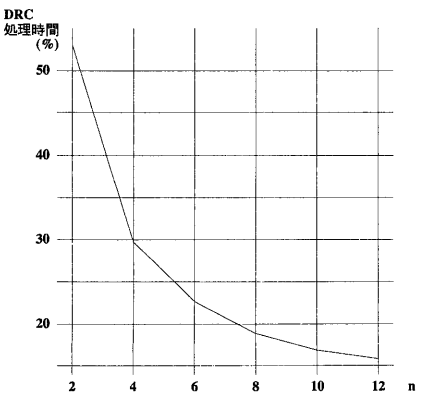

図8: DRC処理時間

図8はDRCのタスクグラフに提案手法を適用して求めたDRC処理時間を示したものである。図中ではプロセッサ1台のときのDRC処理時間で正規化した処理時間が示されている。図8より、DRC処理時間はランダムタスクグラフによる評価の処理時間(図7)と同様な傾向を示していることがわかる。またnが4のときには29%、10のときには16%という処理時間が得られている。

6 むすび

本稿では、処理に並行した通信が困難である単一バス結合マルチプロセッサシステムにおいて、通信オーバヘッドを考慮したスケジューリング手法を提案した。またランダムタスクグラフとDRCに提案手法を適用し、その有用性の評価を行なった。

参考文献

[1] Lenstra J.K. and Kan A.H.G.R.: "Complexity of Scheduling under Precedence Constraints", Oper. Res., 26, 1, pp.22–35 (1978)

[2] Thomas L.Adam, K.M.Chandy and J.R.Dickson: "A Comparison of List Schedules for Parallel Processing Systems", Commun. ACM, Vol.17, 12, pp. 685–90 (1974)

[3] Kasahara H. and Narita S.: "Practical Multiprocessor Scheduling Algorithm for Efficient Parallel Processing", IEEE Trans. Comput., C–33, 11, pp.1023–29 (1983)

[4] 藤原和典, 白鳥健介, 鈴木 真, 笠原博徳: "データプレロードおよびポストストアを考慮したマルチプロセッサスケジューリングアルゴリズム", 信学論 (D–I), J75–D–I, 8, pp.495–503 (1992)

[5] Indurkhya B., Stone H.S. and Lu Xi-cheng: "Optimal Partitioning of Randomly Generated Distributed Programs", IEEE Trans. Software Eng., SE–12, 3, pp.483–95 (1986)

[6] 上坂達生, 池田 浩, 河野一郎, 田丸啓吉: "平面分割によるデザインルールチェックの並列処理方法", 信学技報, VLD93–10 (1993)