計算機アーキテクチャ　83－9
（１９９０．７．１８）

並列実行性に着目したプログラム分割と構造解析

村田 英明　小林 真也　中西 暉　手塚 慶一
大阪大学工学部

並列処理システムにおいてジョブの高速な処理を実現するためには，ジョブをいくつかのタスクに分割し，ジョブの持つ並列性を抽出するジョブ分割法が必要となる．また，ジョブ分割時には，プロセッサのタスク割当てやプロセッサ間の同期実行のためにタスク間の処理依存関係を検出しなくてはならない．本稿では，タスクとメッセージ通信による計算モデルに基づき，ジョブ分割によって抽出される並列性について考察する．次にジョブより高い並列性を抽出することができるメッセージ依存分割法を提案し，そのジョブ分割アルゴリズムとタスク間の処理依存関係の検出アルゴリズムを示す．

Program partitioning algorithms for a parallel processing system

Hideaki MURATA　Shin-ya KOBAYASHI　Hikaru NAKANISHI　Yoshikazu TEZUKA
Faculty of Engineering Osaka University
2-1, Yamadaoka, Suita-shi, Osaka, 565, Japan

A job partitioning method plays an important role in parallel computing systems, for the purpose of executing a job at a high speed. Concurrency and dependency between tasks have to be detected in dividing a job into tasks. In this paper, we propose a job partitioning method, that can extract high concurrency from jobs considering the message passing between two tasks. And we present a job partitioning algorithm and a dependency detecting algorithm.

1．まえがき

現在，計算機技術の重要な分野として並列処理システムが活発に研究されている．並列処理システムとは，いくつかのプロセッサにジョブを分割・配分し，各プロセッサが割り当てられた処理を同時に進めていくシステムである．並列処理システムに期待されるものとしてはスループットの向上，高信頼性，価格性能比の改善等があり，一部ではその研究成果が実用に供せられている．

現在実用化されている並列処理システムの多くは，元来複数でしかも独立しているようなジョブを複数のプロセッサに割り当てることで高速化を実現している．しかしながら，単一ジョブを分割してプロセッサ群に割り当てて実行することによって処理時間の短縮を可能としている並列処理システムは研究段階にある．

このような並列処理システムにおいては，ジョブをタスクと呼ばれる相互に並列実行可能な処理単位に分割するジョブ分割法が必要となる．ジョブ分割法を用れば，見かけ上並列性の現れていないプログラムであっても，適切な分割を施すことによって並列性を抽出し生成することができ，得られたタスク群を複数のプロセッサに割り当て高速な処理が可能となる．また逐次処理言語によって記述された膨大なプログラムの蓄積を活用するためにも，こうしたジョブ分割法は有効であり重要である．

ジョブ分割法には次の２つの点が要求される．

・効率の良い並列性の抽出

分割を行うことによって，タスクの並列実行の可能性を最大限に引き出す．その際，並列処理システムでの高速な処理を妨げる要因となる分割時間，分割回数，並列実行のために各タスクに付加されるオーバヘッドを最小限に抑える．

・タスク特性の抽出

プロセッサへのタスク割当てやタスクの同期実行に必要な情報を得るために，プログラムの構造を解析し，タスクの特性として検出する．

そこで本稿では，まずジョブ分割によって抽出可能なタスクの並列実行性とタスク間の処理依存関係について考察する．次に効率の良い並列性の抽出を行えるジョブ分割アルゴリズムとしてメッセージ依存分割法を提案する．最後にタスクの特性として特に処理依存関係に着目しこれを検出するアルゴリズムについて述べる．

2．ジョブ分割による並列性の抽出

図１に示すようなタスクを並列実行の処理単位とする並列計算のモデルを考える．タスクを各々自立した処理単位とみなし，各タスクはそれぞれ内部手続きと保持すべき内部状態を持つものとする．これらのタスクは，完全に並列実行が可能であることはなく処理の順序に関して制約を持つためタスク全体は処理順序に関して半順序集合をなす．この順序関係を定めるのはタスク間の処理依存関係である．処理依存に関する情報は，タスク間でのメッセージ通信という形で交換される．全体としてのジョブの実行は，ジョブ分割により生成されたこれらタスク間でのメッセージ通信によるタスクの起動やデータの受渡しと，個々のタスクの内部手続きの実行によって構成される．

この並列計算モデルは，例えばこれらタスクをローカルメモリを持つプロセッサ群に割り当て，メッセージ通信をプロセッサ間通信に対応させることによって実現できる．

こうしたタスクとメッセージによる並列計算モデルに基づいて，逐次処理言語で記述されたジョブの高速な並列処理を行うためには，ジョブを分割しタスク相互の並列実行性を最大限に抽出するジョブ分割法が必要となる．

ここでタスク間の処理依存関係について更に分類を行うと，処理依存関係には制御依存関係とデータ依存関係の２種類が存在する．

制御依存関係は，先行するタスクの処理結果に後続のタスクの処理開始が決定される場合に生じる．条件分岐や関数呼出しは，制御依存関係である．制御依存関係の例を図２に示す．

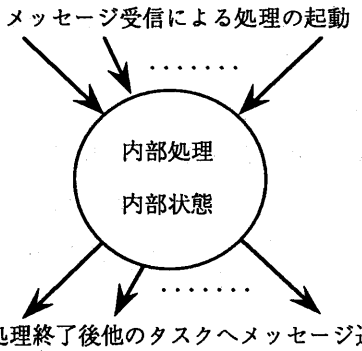


図１ タスクのモデル

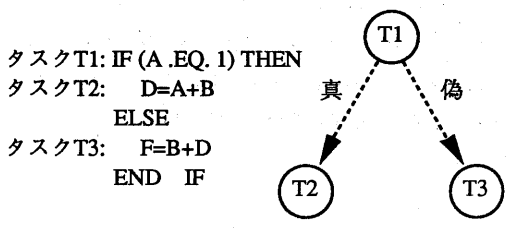


図２ 制御依存関係

データ依存関係は共有変数の書込み・参照関係によって発生し，処理順序を規定する．データ依存関係の例を図３に示す．

このデータ依存関係は，図４に示した従来のジョブ分割法におけるデータ依存関係の概念では図４（ａ）のフロー依存関係に相当する．しかしながら，フロー依存関係以外の逆依存関係及び出力依存関係については，処理順序を規定する関係とはならない．なぜならば，この並列計算モデルにおいては，逆依存関係及び出力依存関係の存在するタスク間ではメッセージの送信は起こらず，相互に並列実行が可能となるからである．

ここでタスクとメッセージの並列計算モデルに基づいたジョブ分割による並列性の抽出と処理依存関係の検出の例を図５に示す．図５（ａ）のソースプログラムを分割し，図５（ｂ）のタスクを生成したとする．ここでＴ１とＴ２，Ｔ３とＴ４はそれぞれ共有変数によってデータ依存関係にある．また従来のデータ依存の概念によればＴ２とＴ３の間は逆依存関係にあることがわかる．これらのデータ依存関係の制約から，逐次処理計算機で実行した場合には図５（ｃ）のような実行の流れになる．一方，並列計算を行った場合には図５（ｄ）に示すように，Ｔ１とＴ２，Ｔ３とＴ４に関してはそれぞれデータ依存関係により処理順序の制約が存在するが，Ｔ１－Ｔ２とＴ３－Ｔ４は互いに並列実行が可能となる．これは各タスクが内部に変数の値を保持し内部の処理を終えた後データを他のタスクに送信する機能をもつために，Ｔ２とＴ３の逆依存関係による処理順序が不必要となるためである．すなわちタスクとメッセージの並列計算モデルに基づいたジョブ分割法によって，逐次計算における共有変数の操作順序から発生する処理依存関係が取り除かれ，潜在的な並列実行性を引き出している．

このようにして，ジョブの持つ並列性を抽出

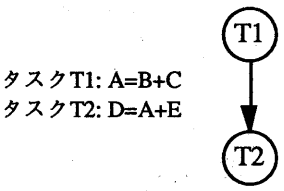


図３ データ依存関係

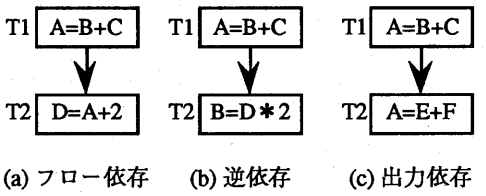
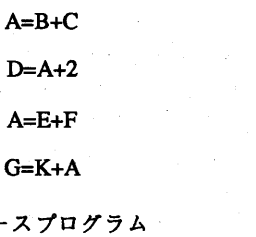


図４ 従来のデータ依存関係の概念



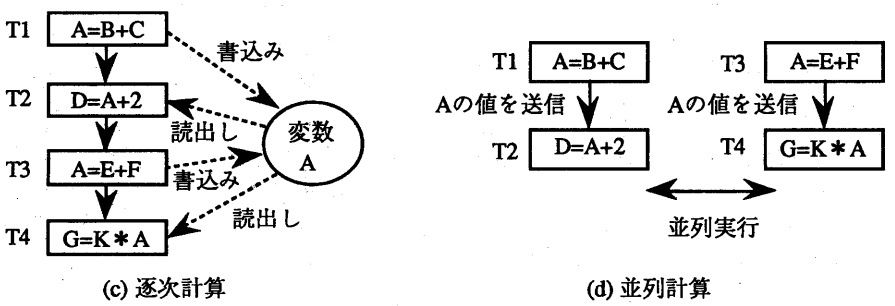


図５ ジョブ分割による並列性の抽出

し，同時に並列実行に必要な処理依存関係を検出することがジョブ分割法の重要な役割である．

図１の並列計算モデルに従うようなタスクとメッセージを，実際のプログラムから生成するために，ここで改めてタスクとメッセージの定義を行う．またジョブ分割法がどの程度効率よく並列性を抽出しているかを判断する基準として理想的な分割の定義も行う．

（定義）

ジョブＪ≡｛処理依存関係による半順序関係を持った処理要素の集まり｝

タスク集合Ｔ≡｛ジョブの部分集合Ｔｎ．ただしｎ≠ｍに対してＴｎ∩Ｔｍ＝φ，∪Ｔｎ＝Ｊ｝

（定義）

あるタスク集合Ｔ＝｛Ｔｎ｝において任意のタスクについて次の関係が満たされているとき，このタスク集合Ｔを得るような分割を理想的な分割という．

（関係１）タスクＡの開始以前にタスクＢは処理不可能であるとき，タスクＢの実行は必ずタスクＡの終了後である．

（関係２）あるタスクの終了後に実行される可能性のあるタスクが複数個存在するか，またはそのタスクの実行開始以前に実行が終了しなくてはならないタスクが複数個存在する．

理想的な分割は，ジョブのタスクへの分割回数を最小に抑えながら最大の並列性をジョブから抽出する．

（定義）

ジョブＪの実行において，理想的な分割の行われたタスク集合Ｔのタスク間で起こる通信をメッセージと呼ぶ．

以上の定義の下では，理想的な分割によって得られるタスク集合において開始タスク及び終了タスクを除いた全てのタスクは，内部の処理の開始時に他のタスクよりメッセージを受信し，処理中にはメッセージの送受信を行わず，処理の終了時に他のタスクへメッセージを送信する．

理想的な分割と理想的でない分割の違いを示す例を図６にあげる．図６（ａ）ではＴ１－Ｔ２－Ｔ５，Ｔ１－Ｔ３－Ｔ５，Ｔ１－Ｔ４－Ｔ５の３つの互いに並列実行可能な系列を抽出していることがわかる．一方図６（ｂ）においてもやはり３つの並列実行可能な系列を抽出しており，並列性の抽出という観点からはどちらの分割も同等と言える．しかしながら図６（ａ）におけるＴ２及びＴ４は図６（ｂ）では更に分割されている．これらの分割は並列性の抽出には寄与せず，しかも並列処理システムでの実行においてはオーバヘッドにつながる．このことから，図６（ａ）の分割は並列性を十分抽出し，しかも分割によるオーバヘッドの少ない理想的な分割であることが言える．

３．メッセージ依存分割法

ジョブを記述するプログラムを分割する際，メッセージ通信の発生する可能性の高い位置で分割を行えば，より理想的な分割に近いジョブ分割が実現できる．このようなタスク間のメッセージに着目した分割法として，メッセージ依存分割法を提案する．

このメッセージ依存分割法は，プログラム中のメッセージを明らかにすることによって分割を行うため，特定の言語には依存しない．しかしジョブがＣやＦＯＲＴＲＡＮ等の言語で記述されている場合，全てのメッセージの送受信を明らかにすることは容易ではない．従って必要な分割が行われない，あるいは無駄な分割が行われるといったことが起こりうる．

まずメッセージ依存分割法の基本的なアルゴリズムについて述べた後，従来のジョブ分割法と比較した評価結果について述べる．

３．１　分割アルゴリズム

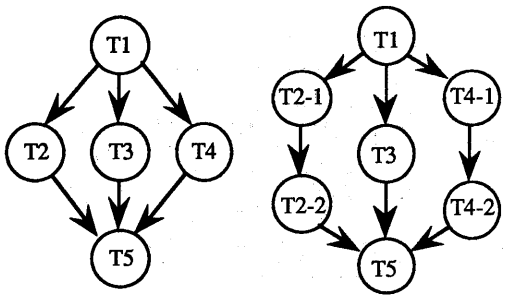
メッセージ依存分割法は，以下のアルゴリズムでジョブ分割を行う．

（ステップ１）ジョブを記述したプログラム中で，メッセージの送受信点を全て見いだす．

（ステップ２）見いだしたメッセージの送受信点の直後を分割位置として前後に分割する．

ここでジョブの記述言語として，特殊なライブラリルーチンを一切用いないＦＯＲＴＲＡＮ７７を仮定する．このとき次のような記述がメッセージとなる可能性が高いと考えられ，実際の分割位置となる．

分割位置：プログラムの流れを制御する実行文



(a) 理想的な分割　　(b) 理想的でない分割

図6.理想的な分割

を含む行．ＣＡＬＬ，ＩＦ（条件式）ＴＨＥＮ，ＥＬＳＥ，ＥＮＤ　ＩＦ，ＤＯ，ＣＯＮＴＩＮＵＥ，ＥＮＤ

またＦＯＲＴＲＡＮ特有の記述に対応して定める細則は次のとおりである．

（細則１）ＤＯ文による繰り返し処理

繰り返し処理単位の末尾はＣＯＮＴＩＮＵＥ文となっているものと仮定し，繰り返し処理の先頭のＤＯ文の直後の行から末尾のＣＯＮＴＩＮＵＥ文までの実行文の集合をＤＯループと呼ぶこととする．そしてＤＯループ内部に分割点が存在しない場合はＤＯループ全体を一つのタスクとする．一方分割点が存在する場合には，ループの繰り返し回数がプログラムの実行開始以前に定まっている場合に限り，繰り返しにおいて異なる要素全てに対してタスクを一つずつ生成する．

（細則２）異なるＣＡＬＬ文による同一サブルーチンの呼び出し

プログラム中で異なった位置にあるＣＡＬＬ文において同一サブルーチンを呼び出している場合は各ＣＡＬＬ文に対応したタスクがそれぞれ別個に生成されるものとする．

３．２　メッセージ依存分割法の評価結果

比較対象として，ベクトル化及びモジュール分割法を選択した．これらの分割法とメッセージ依存分割法とを同一のプログラムに適用し，分割回数と並列性抽出度を評価尺度に用いて比較した．

ここでベクトル化及びモジュール分割法の基本的なアルゴリズムとその特徴について簡単にまとめる．

ベクトル化

ベクトル演算など同時処理可能な計算をベクトルプロセッサで並列に処理するために，プログラムのループ部分において各ループ要素毎の演算を互いに分割する．

・分割アルゴリズムが簡単であり，ベクトルプロセッサというハードウェアに適合した並列性が抽出可能である．

・繰り返しの各要素に対応した処理が互いに依存する実行順序を持つ場合には分割を行っても並列性は得られない．

・ＤＯループというプログラムの一部分に注目して並列化するため，必ずしもプログラム全体の並列処理を考えた場合の最適な分割が行われない．

モジュール分割法

ＦＯＲＴＲＡＮなどの逐次処理型高水準言語による大規模なプログラムを対象とする．プログラムはモジュールと呼ばれる処理単位で構成されているとみなし，このモジュールに着目して分割を行う．

・プログラム記述全体にわたって分割を行うため，ＤＯループにとどまらない広い範囲からの並列性を抽出可能．

・モジュールに対して再帰的に分割を行う点や隣接文で互いに依存関係がなければ分割を行うことから，分割回数が増し数多くのタスクが生成される．

いくつかのプログラムに適用し，次のような結果が得られた．

１）メッセージ依存分割法は他の分割法に比べて分割回数を低く抑え高い並列性抽出度を得ることができ，より理想的な分割を行える．

２）従来の分割法がスケジュール不可能なジョブを分割できないのに対して，メッセージ依存分割法はスケジュール不可能なジョブも分割可能である．

４．処理依存関係の検出

ジョブ分割により生成されるタスクの集合は，並列処理システムにおける次の処理段階であるプロセッサへのタスク割当て及び複数プロセッサでの同期実行へと引き渡されていく．そしてジョブの同期実行段階においては各プロセッサに割り当てられたタスクは相互の処理依存関係に従って必要な情報をタスク間通信によって交換する．このとき，異なるプロセッサに割当てられたタスク間のメッセージによる通信はプロセッサ間通信を引き起こし，並列処理におけるオーバーヘッドとなる．従ってタスクの割当て・同期実行を行うために個々のタスクの持つ処理量やタスク相互の処理依存関係といったタスクの持つ特性を，並列実行性という観点から検出する必要がある．

ここではタスク間の処理依存関係をタスク特性として着目し，その検出アルゴリズムを示す．

まず，ジョブ分割によって得られたタスク間の処理依存関係は，制御依存関係とデータ依存関係の２種類に分類できる．

制御依存関係は条件分岐や手続き呼出によって発生する．ＦＯＲＴＲＡＮの場合にはブロックＩＦ文，ＣＡＬＬ文，ＤＯ文において分割・生成されたタスク間には制御依存関係が存在するので，これを検出する．

一方，データ依存関係は次のアルゴリズムによって検出される．

（ステップ１）分割・生成されたタスクに対し，

逐次処理した場合の処理順序に従った番号付けを行う．これを処理順序番号と定義し，タスクTの処理順序番号をｎｕｍ（Ｔ）と表記する．
（ステップ2）タスクTにおいて参照している全ての変数の集合をＶｒ（Ｔ）とする．各変数ｖ∈∈Ｖｒ（Ｔ）について，ｖの書込みを行い，かつｎｕｍ（Ｔｗ（ｖ））＜ｎｕｍ（Ｔ）であるようなタスク集合｛Ｔｗ（ｖ）｝を求める．
（ステップ3）｛Ｔｗ（ｖ）｝の要素の中で最も処理順序番号の大きなタスクＴｄを求める．Ｔｄが存在するとき，ＴｄとＴはデータ依存関係にある．各変数ｖ∈Ｖｒ（Ｔ）に対してＴｄを求める．
（ステップ4）任意のタスクTについて（ステップ2）～（ステップ3）を行う．
タスク集合に対してこのアルゴリズムを適用することにより，データ依存関係を全て検出することができる．
図7を用いてデータ依存検出アルゴリズムの適用を説明する．
（ステップ1）タスクＴ１～Ｔ４の処理順序番号が図のように求められるものとする．
（ステップ2）例えばＴ４についてデータ依存関係を求めると次のようになる．Ｔ４において参照している変数はＡであり，Ａの書込みを行っているタスクであって処理順序番号がＴ４より小さいものはＴ１及びＴ３である．
（ステップ3）Ｔ１とＴ３で処理順序番号が大きいものはＴ３である．よってＴ３とＴ４はデータ依存関係にある．
（ステップ4）Ｔ１，Ｔ２，Ｔ３についても同様にしてデータ依存関係が求められる．

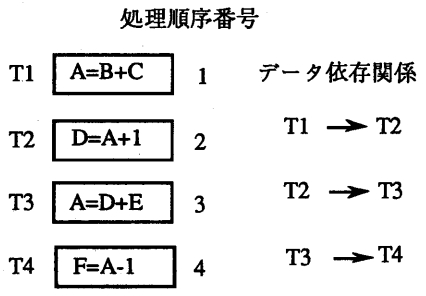


図7 データ依存関係の検出例

5．むすび

タスクとメッセージによる並列計算モデルに基づいたジョブ分割により，従来のジョブ分割における並列性に比べて，より高い並列性を抽出しうることを示した．また，ジョブを分割し高い並列性を持ったタスクを生成するジョブ分割法としてメッセージ依存分割法を提案し，その分割アルゴリズムと処理依存関係の検出アルゴリズムを述べた．今後の課題としては，
・メッセージ依存分割法をより多くのプログラムに適用し，本分割法の有効性とその適用範囲を明らかにする．
・並列処理システムを構成する処理モジュールの１つとしてのメッセージ依存分割法のインプリメントを行う．
こと等が挙げられる．

＜参考文献＞

[1] Padua,D.A.Wolf,M.J.:Advanced Compiler Optimizations for Supercomputers, Commun.ACM,Vol.29, No.12,pp.1184-1201(1986)
[2] 菊池，白鳥，宮崎：逐次型高水準言語プログラムのモジュール分割による並列性の抽出について，信学論(D),J71-D,No.8,pp.1525-1531(1988)
[3] 村田，小林，中西，手塚：プログラム構成単位間通信に注目した並列性抽出に関する研究,信学技報,CPSY89-58,pp.81-86(1989)
[4] 小林，小巻，中西，手塚：タスク間の排反性に注目したタスク割当法に関する研究，SWoPP琉球'90,信学会CPSY(1990)