分散メモリシステム上でのマクロデータフロー処理のためのデータ到達条件

本多弘樹[†] 上田哲平[†,☆]
深川保[†] 弓場敏嗣[†]

並列性表現として実行開始条件用いた粗粒度処理手法では，粗粒度タスクを動的にプロセッサに割り当てることによりマクロデータフロー処理を共有メモリシステム上で実現している．このマクロデータフロー処理手法を PC クラスタなど分散メモリシステム上で実現するためには，異なるプロセッサに割り当てられたタスク間での明示的なデータ通信のための機能が新たに必要となる．その機能の 1 つとして実行時に決定される変数の定義・使用関係に基づく送信元・送信先の決定が重要となるが，実行開始条件だけではそのための情報が不足している．本論文では，従来の到達する定義の概念を拡張することにより，分散メモリシステム上でマクロデータフロー処理を実現する際に必要となるマクロタスク間データ通信の送信元・送信先を実行時に決定する方式として「データ到達条件」を提案する．

The Data Reaching Condition for Macro-data-flow on Distributed Memory System

Hiroki Honda,[†] Teppei Ueda,[†,☆] Tamotsu Fukagawa[†] and Toshitsugu Yuba[†]

The coarse grain task parallel processing scheme using “execution start condition”, which represents the parallelisms among coarse grain tasks, realizes a macro-dataflow processing by dynamic-assignment of tasks onto processors on a shared memory system. An implementation of the macro-dataflow processing on a distributed memory system, such as PC-cluster, requires new functions for explicit data transfer between tasks assigned to distinct processors. As one of the functions, it is vital to make a sender- receiver pair based on the use-definition chain determined at run-time; however, the execution start condition has no information needed for the function. This paper proposes “data reaching condition”, a method to make a sender-receiver pair of a data transfer for the macro-dataflow processing on a distributed memory system, by extending the concept of the conventional reaching definition.

1. はじめに

ループ並列処理の限界を超えるための粗粒度並列処理の 1 つとして，粗粒度タスク（マクロタスク）レベルでのマクロデータフロー処理（MDF）が注目されている[1)]．MDF では，ループや基本ブロックなどのマクロタスク間の並列性をコンパイル時に実行開始条件[2)]（もしくはそれと同等のもの[3)]）として求め，並列実行時には実行開始条件が成立した（すなわち使用する変数にアクセスが可能になった）マクロタスクを動的に順次プロセッサに割り当てることで並列実行を進める．

これまでの MDF の実装は共有メモリ型並列計算機システム（共有メモリシステム）を対象としていた．よって，実行開始条件が成立したマクロタスクで使用する変数の値は共有メモリ上に存在することを前提とすることが可能で，プロセッサ間での明示的なデータ通信を考慮する必要はなかった．

一方，近年普及してきている PC クラスタなど，分散メモリ型並列計算機システム（分散メモリシステム）上で MDF を実現するには，異なるプロセッサに割り当てられたマクロタスク間のデータ授受をプロセッサ間通信により行う必要がある．

分散メモリシステム上で MDF を実現する方法の選択肢としては，分散メモリシステム上にまずソフト

† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications
☆ 現在，株式会社日立製作所
Presently with Hitachi, Ltd.

ウェア分散共有メモリ（SDSM）を実装しそれを用いてMDFを実現する方法とSDSMを用いずにメッセージパッシング通信による方法とが考えられる．

SDSMを用いた実装ではSDSMにより仮想的に共有メモリ空間が提供されるため，MDFの実装レベルでの明示的なプロセッサ間通信の必要はなく，従来の実装方式とのギャップを少なくでき実装が容易であるという利点があると考えられる．一方この方法では，その性能が各種SDSMでの一貫性制御方式などに依存するため，適切なSDSMの選択やそのSDSM向きの実装上のチューニングなどの課題を解決する必要があると考えられる．また，各SDSMの実装可能なOSに制限がある場合もある．

メッセージパッシング通信による方法では，MDFの実装レベルで明示的にプロセッサ間通信を行わなければならない点が大きな課題となる．一方，明示的なデータ通信によりその動作を直接制御しやすく性能のボトルネックなどの見通しが良いなどの利点があると考えられる．また，MPI（Message-Passing Interface）など広く用いられている通信ライブラリを用いればその実装の適応範囲も広いと考えられる．

最終的なシステムとしてどちらの方式が優位であるかは，性能，ポータビリティやスケーラビリティなどをトータルに勘案して判断しなければならないが，どちらも実装が行われていない現状ではどちらの方式が優位であるかは明確ではなく，両方式の実装とその比較評価が必要である．

そこで，我々はまずメッセージパッシング通信による方法での実装から試みることとし，本論文ではMDFにおける明示的なプロセッサ間通信に関する課題について取り組む．

明示的なプロセッサ間通信のためには，どのマクロタスク間でどの変数に対してのデータの授受が必要となるかを実行時に判断しデータ授受の送信元・送信先の関係を決定しなければならないが，これまでのMDFではそのための方法がなかった．

これに対し本論文では，マクロタスク間でのデータ授受の関係をコンパイル時に「データ到達条件」として求め，実行時にこの条件を検査することによりデータ授受の送信元・送信先の関係を決定する方法を提案する．またデータ到達条件の概念により，MDFにおけるプリロードを分類整理し，新たなプリロードの方法「投機的プリロード」が可能となることを示す．さらに本論文では分散メモリシステム上でのデータ到達条件を用いたMDFの予備的な実装を行いその動作の検証を行う．

2. マクロタスクの並列実行と実行開始条件

本論文が前提とするMDFの並列実行方式とこれを実現するためのマクロタスクの実行開始条件[2]の概要を述べるとともに，3章のデータ到達条件の定義で用いる概念を説明する．

2.1 マクロタスク

MDFはプログラムをマクロタスクの集合としてとらえ，各マクロタスクをプロセッサへの割当て単位として並列処理するものである．マクロタスクはプログラム中の一部分で，その部分の実行を開始する文がその部分の先頭の行であるものである．たとえば，ループ，基本ブロック，サブルーチンがマクロタスクとなりうる．マクロタスクの処理はノンプリエンプティブに行われる．

コンパイル時には，プログラムをマクロタスクに分割し，マクロタスク間の制御フローとデータ依存をマクロフローグラフ（制御フローグラフとデータ依存グラフを統合したグラフ）で表現する．さらにマクロフローグラフから，次節で述べる実行開始条件を求める．

マクロタスク分割に際しては，マクロタスクのサイズ（粒度）をどのように設定するかが重要な課題であり，各種オーバヘッドとの大きさの比率やマクロタスク相互の大きさのばらつき，またマクロタスク間でのデータ通信量などを考慮してマクロタスクを分割または融合し，さらには階層的にマクロタスクを生成すること[1),4),8)]も必要であるが，詳細は参考文献にゆずり本論文ではこれについてはこれ以上議論しない．

マクロフローグラフの例を**図1**に示す．図中方形で表すノードがマクロタスク，横の数字がマクロタスク番号，円形が分岐，破線で表すエッジが制御フロー，実線で表すエッジがデータ依存を表している．

マクロタスクへの分割にあたっては，制御フローグ

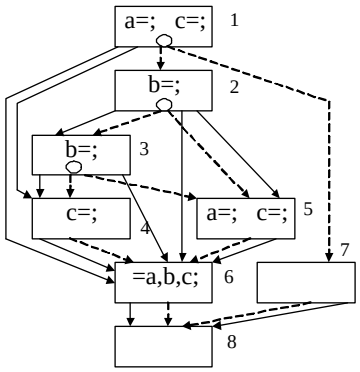

図1 マクロフローグラフの例
Fig. 1 Example of macro-flow graph.

ラフが非循環となるように分割する．

2.2 実行開始条件

実行開始条件とは，あるマクロタスクの実行開始が可能となるための条件で，他のマクロタスクの実行状況を項とした論理式で表現したものである．この論理式は「マクロタスク終了」と「分岐方向決定」の 2 種類の原子条件および論理演算子∨ (論理和) と∧ (論理積) で構成される．マクロタスク MTa の終了条件は，MTa の実行が終了したときに True となる条件で，論理式中では a と表記する．マクロタスク MTb から MTc への制御フローエッジへの分岐による分岐方向決定条件は，条件分岐の条件式の評価によってこの制御フローエッジへの分岐が確定したときに True となる条件で，論理式中では b-c と表記する．

マクロタスク MTi の実行開始条件は式 (1) のように定義される．

(MTi の実行確定条件) ∧
(MTi のデータアクセス可能条件) (1)

式 (1) 中の (MTi の実行確定条件) は，MTi を実行することが確定するための条件で，MTi が制御依存[5] するマクロタスクから MTi が逆支配するマクロタスクへの分岐による分岐方向決定条件の論理和で構成される．

式 (1) 中の (MTi のデータアクセス可能条件) は，MTi で使用する変数へのアクセスが可能となる条件で，MTi がデータ依存する全マクロタスクのそれぞれのマクロタスクに対するデータアクセス可能条件の論理積で構成される．

MTi がデータ依存するマクロタスク MTj に対するデータアクセス可能条件は式 (2) のように定義される．

(MTj の終了条件) ∨
(MTj の非実行確定条件) (2)

式 (2) 中の (MTj の非実行確定条件) は MTj が実行されないことが確定するための条件で，MTj が補制御依存[2] するマクロタスクから MTj へ至らないパスへの分岐による分岐方向決定条件の論理和で構成される．

MTj が補制御依存するマクロタスク MTk とは以下の性質を満たすマクロタスクである．

(1) 制御フローグラフの入口ノードから MTj に至るパス上にあるすべてのマクロタスクの集合を X とするとき，MTk は X に含まれ，かつ
(2) MTk は X 内のマクロタスク以外への分岐を持つ．

図 1 中のマクロタスク 6（MT6）を例にとるとその実行開始条件は**表 1** のとおりとなる．

表 1 図 1 中の MT6 の実行開始条件
Table 1 Execution start condition for MT6 in Fig.1.

実行確定条件	1-2	
データ	MT1	1
アクセス	MT2	2 ∨ 1-7
可能条件	MT3	3 ∨ 1-7 ∨ 2-5
	MT4	4 ∨ 1-7 ∨ 2-5 ∨ 3-5
	MT5	5 ∨ 1-7 ∨ 3-4
実行開始条件	1-2 ∧ 1 ∧ (2 ∨ 1-7) ∧ (3 ∨ 1-7 ∨ 2-5) ∧ (4 ∨ 1-7 ∨ 2-5 ∨ 3-5) ∧ (5 ∨ 1-7 ∨ 3-4)	

なお，上記の定義での実行開始条件中には冗長な分岐方向決定条件が含まれており，実装においては条件評価のオーバヘッドを削減するため，冗長な条件を除去した条件[2] を用いる．

2.3 マクロタスクの並列実行管理

MDF では，スケジューラがマクロタスク実行プロセッサに順次マクロタスクを割り当てることで処理を進める．スケジューラの実装には，集中型として特定のプロセッサで実行する方式と各プロセッサに分散させて実行する方式がありうるが，ここでは集中型について説明する．

スケジューラのコードはコンパイル時に実行開始条件をもとに生成される．スケジューラは以下の動作を繰り返す[6]．

(1) 実行開始条件の検査：各マクロタスクから通達されるマクロタスク終了および分岐方向決定の情報をもとに実行開始条件を検査し，条件が成立したマクロタスクをレディマクロタスクとする．
(2) プロセッサへの割当て：所定の割当戦略に従ってレディマクロタスク中のどのマクロタスクをどのプロセッサに割り当てるかを決定する．
(3) 各プロセッサに実行すべきマクロタスク番号を通知する．

マクロタスクを実行する各プロセッサのコードには，すべてのマクロタスクの実行コードに加えて，スケジューラからのマクロタスクの割当て・実行指示を受け取るコードとマクロタスク終了と分岐方向決定をスケジューラへ通達するコードが含まれる．

3. データ到達条件

3.1 マクロタスク間でのデータ授受

実行開始条件を用いた並列実行にあたっては，マクロタスク MTi の実行開始条件が成立した時点で，MTi のデータ依存マクロタスクのいずれもその実行が「終了している」か「実行されないことが確定している」ことが保証されている．このため共有メモリシステム

上での実装では，たとえデータ依存するマクロタスクが異なるプロセッサで実行されていたとしても，特にマクロタスク間のデータ授受を明示的に行う必要はなく，原理的には（キャッシュコヒーレンスにかかる問題は別として）MTi で使用する変数の値は共有メモリ上に存在しているとして MTi の実行を開始できる．よって，同一変数に関して複数のデータ依存マクロタスクが存在しても，どのデータ依存マクロタスクとの間でデータ授受が行われるのかは考慮する必要はない．

一方分散メモリシステムにおいては，データ依存関係にあるマクロタスクが異なるプロセッサで実行される際には，明示的なデータ通信が必要となる場合があり，そのためにはどのマクロタスク間でどの変数に関するデータ授受を行うかが明確でなくてはならない．

しかしながらあるマクロタスク MTi で変数 V の使用（MTi 外で定義された値の使用）があり，V への定義（代入などの変数への値の設定）が複数のマクロタスクで行われうる際，そのうちのどのマクロタスクで定義された V の値を MTi で使用すべきなのかは一般に実行時にならなければ決定できない．

よって，どのマクロタスク間でどの変数に関する通信を行うかの判断も実行時に行うこととなるが，実行開始条件だけではそのための情報は不十分で，通信相手と変数を決定することができないという問題がある．

3.2 データ到達条件

データ到達条件とは，マクロタスク MTi（の先頭の文）に到達する[7] 変数 V への定義を持つマクロタスクの集合を S_V^i としたとき，MTi（の先頭の文）での V の値が，MTj $\in S_V^i$ で定義した値となることが確定するための条件で，実行開始条件と同様に他のマクロタスクの実行状況を項とした論理式で表現する．

MTi での V に対する MTj $\in S_V^i$ のデータ到達条件は，MTj から MTi へのパス上において，MTj での V への定義を kill する定義（ここでは確実な定義による kill のみを念頭に置く）を持つすべてのマクロタスクの集合を K としたとき，式 (3) のように定義する．

(MTj の実行確定条件) ∧

(K 中の全マクロタスクの非実行確定の条件) (3)

式 (3) 中の (MTj の実行確定条件) の定義は前述の実行開始条件の場合と同様である．

式 (3) 中の (K 中の全マクロタスクの非実行確定の条件) は，MTj から MTi へのパス上で MTj での V への定義を kill する定義を持つマクロタスクは 1 つも実行されないことが確定する条件で，そのようなマクロタスクの非実行確定条件の論理積で構成される．マクロタスクの非実行確定条件の定義は前述の実行開始条件の場合と同様である．

表 2 図 1 中の MT6 に関するデータ到達条件
Table 2 Data reaching condition for MT6 in Fig.1.

変数	定義を行うマクロタスク	データ到達条件
a	MT1 MT5	True ∧(1-7 ∨ 3-4) (2-5 ∨ 3-5) ∧ True
b	MT2 MT3	1-2 ∧ (1-7 ∨ 2-5) 2-3 ∧ True
c	MT4 MT5	3-4 ∧ Ture5 (2-5 ∨ 3-5) ∧ Ture

並列実行中に MTi での V の使用に関するデータ到達条件が成立するのは S_V^i 中のただ 1 つのマクロタスクとなる．

図 1 の MT6 を例にとると，MT6 で使用される各変数とこれに定義を行う各マクロタスクの組に対するデータ到達条件は**表 2** のとおりとなる．

3.3 データ到達条件の求め方

マクロタスク MTi で使用する変数 V と MTi へ到達する V への定義を持つマクロタスク MTj の組に対するデータ到達条件は以下のように求めることができる．

(1) MTj の実行確定条件を求める．
(2) 集合 K ={MTk | MTk は MTj から MTi へ至るパス上にあり MTj での V の定義を kill する定義を持つマクロタスク } を求める．
(3) K 中の各マクロタスクに対し非実行確定条件を求める．
(4) (1) と (3) で求めた条件の論理積をデータ到達条件とする．

ただし，式 (3) によるデータ到達条件の定義と上記の求め方では，データ到達条件中に冗長な条件を含んでいる．具体的には式 (3) 中の（K 中の全マクロタスクの非実行確定の条件）を構成する分岐方向決定条件うち，次の 2 種の条件が冗長な条件となる．

(1) MTj から MTi に至るすべてのパスで構成される部分グラフ中に含まれない制御フローエッジによる分岐方向決定条件：このような制御フローエッジへの分岐は MTi と MTj のどちらか片方か両方が実行されない場合に生じる分岐であり冗長である．
(2) K 中のマクロタスクの実行確定条件中に含まれる分岐方向決定条件と同一のもの：このような分岐方向決定条件は MTj での V への定義を kill するマクロタスクの実行を確定するものあり冗長である．

冗長な条件が存在しても論理的にはデータ到達条件に変化はないが，実装上はオーバヘッド削減のために冗長な条件を除去したほうが望ましい．

上記の冗長さを除去した式 (3) 中の（K 中の全マクロタスクの非実行確定の条件）は次のように求めることができる．

(1) MTj から MTi へ至るすべてのパスで構成される部分マクロフローグラフ上で，MTi から MTj へ向けて次の規則に従いながら制御フローエッジを深さ優先でさかのぼる．
- 分岐を持つマクロタスクに到達した際，そのマクロタスクの出口側のすべての制御フローエッジに到達している場合には，入口側の制御フローエッジからさらにさかのぼる．そうでなければさかのぼりをやめる．
- K 中のマクロタスクまたは MTj に到達した際にはさかのぼりをやめる．

(2) それ以上さかのぼれなくなった時点で，K 中のマクロタスク以外のマクロタスクに到達した分岐エッジによる分岐方向決定条件の論理積が式 (3) 中の（K 中の全マクロタスクの非実行確定の条件）となる．

実装においてはコンパイル時のスケジューラコード生成の際に，各マクロタスクに対してそのマクロタスクで使用するすべての変数とそのマクロタスクに到達する定義を持つマクロタスクの組に対してデータ到達条件を求める．

3.4 スケジューラでの通信管理

分散メモリシステム上でのスケジューラは，実行開始条件の検査とマクロタスクの割当て・実行指示に加えて，データ到達条件の検査によりデータ通信の必要性を判定し，必要であればプロセッサにその指示を与える．

具体的には，2.3 節で述べた動作 (2) でマクロタスク MTi の割当てをプロセッサ Pi に決定した後に，MTi で使用する変数 V ごとに通信の必要性判定と通信指示のための動作を以下のように行う．

(1) データ到達条件の検査：V に関するデータ到達条件を評価し，条件が成立したマクロタスク MTj とそれが割り当てられたプロセッサ Pj を求める．

(2) 通信の指示：MTj が生成した V の値がすでに Pi に存在する場合（Pi と Pj が同一の場合，もしくは Pi ですでに実行されたマクロタスクで使用するために V の値が Pi に通信済みの場合）にはデータ通信の指示は行わない．そうでない場合は，Pj に対して MTj で定義された V の値を Pi へ送信するよう指示し，さらに Pi に対しては MTi の実行に先立ち Pj から V の値を受信するよう指示する．

3.5 不確実（ambiguous）な定義による kill

これまでの議論では，あるマクロタスク MTk で変数 V へ定義が行われるとは，MTk 内で必ず V への「確実な定義[7]」が行われることを前提としていた．一方，実際のプログラムでは，(i) MTk 内の文での V への定義自体が「不確実な定義」である場合，(ii) V への定義を実行するか否かがマクロタスク内の条件分岐方向によって実行時に決定される場合，(iii) V が配列変数でそれぞれの要素に対する定義を実行するか否かが実行時に決定される場合，がありうる．MTk での変数 V への定義が不確実な場合（上記 (ii) と (iii) も含める），V への定義が MTk で kill されるか否かが確定せず，実行開始条件とデータ到達条件を求めることができない．

これに対しては，MTk に到達する V への定義の値を MTk の入口で V へ定義（代入）しなおすこととすれば，MTk で確実な定義が行われるようにできる．この方法では，実際には kill されてしまう定義による値のデータ通信も行ってしまい無駄が生じるが，実装としては最も簡便である．

ポインタ解析，手続き間解析，分岐方向解析，配列添字解析によって不確実さを除去したり，kill を「確実な kill」と「不確実な kill」に区別して実行開始条件とデータ到達条件を求めたりすることにより不要なデータ通信を低減させることも考えられるが，これらについてはここでは議論しない．

4. データ到達条件のプリロードへの応用

ここでは MDF においてどのようなプリロードが可能であるかを議論し，従来のプリロードに加えてデータ到達条件の概念により実現可能となるプリロードを示す．

4.1 マクロデータフロー処理でのプリロード

マクロタスク処理と通信とをオーバラップして実行できる環境では，マクロタスク MTi の実行開始以前に，他のマクロタスクの処理と並行して MTi で必要なデータの通信を行うプリロードによりデータ通信によるレイテンシを隠蔽することが望ましい．

以降の議論では，マクロタスク MTi で使用する変数 V への定義を持つマクロタスク MTj から MTi へのプリロードを行ううえで次の条件を仮定する．

(1) マクロタスクの投機実行（実行開始条件が成立

していないマクロタスクの実行）はしない．

(2) MTi を割り当てるプロセッサが決定している．

(3) MTj が終了している．

また，スケジューラがマクロタスクをプロセッサに割り当てることとプロセッサにそのマクロタスクの実行の指示を与えることは個別の動作であるとする．

4.2 プロセッサ割当てとプリロードの分類

ある時点での，マクロタスク MTi の実行開始条件の状況は**表 3** のように場合分けできる．これをもとに，マクロタスク MTi のプロセッサ割当ての方法を，割当て時点に MTi が表 3 のどの状況にあるかという点で分類すると以下のようになる．

通常割当て： a の状況での割当てで，実行開始条件が成立したレディタスクのみを割り当てる．

データ依存待ち割当て： b，c の状況での割当てで，割当て後にデータアクセス可能条件の成立を待つ．

制御依存待ち割当て： d の状況での割当てで，割当て後に実行するか否かが決定される．

制御・データ依存待ち割当て： e，f の状況での割当て．

上記の (2)，(3)，(4) は，実行開始条件が未成立のマクロタスクに対しても割り当てるプロセッサを決めているもので，今後これを「プリアサイン」と呼ぶこととする．

一方，MTi で使用する変数 V への到達する定義を持つマクロタスク MTj の状況は**表 4** のように場合分けできる．表 4 の各状況で，MTi での V の値と MTj の関係および MTj で定義された値のプリロードの可否は次のようになる．

A： MTj で定義済みの V 値を使用することが決定している．プリロード可．

表 3 実行開始条件の状況
Table 3 Status of execution start condition.

		データアクセス可能条件		
		全 MT 成立	一部 MT 成立	全 MT 未成立
実行確定条件	成立	a	b	c
	未成立	d	e	f

表 4 データ到達条件とデータアクセス可能条件の状況
Table 4 Status of data reaching condition and data access condition.

		データアクセス可能条件		
		成立（終了）	成立（非実行確定）	未成立
データ到達条件	成立	A	n.a.	B
	未成立	C	D	E

B： MTj で定義される V の値の使用は決定しているが MTj はまだ実行中．プリロード不可．

C： MTj で定義済みの V の値を使用するか否かは未定．プリロードは可能であるがプリロードした変数の値を使用しない可能性もある．

D： MTj は実行されないことが決定している．プリロードは不必要．

E： MTj が実行されるか否か，MTj で定義される V の値を使用するか否かは未定．プリロード不可．

上記の分類から，マクロタスクのプロセッサへの割当てとその割当てでのプリロードには以下の組合せが考えられる．

(1) 通常割当て：到達する定義を持つマクロタスクの状況はそれぞれ A，C，D のいずれかであることが確定しおり，以降変化しない．プリロードの対象となるのは A の状況のマクロタスクだけである．

(2) データ依存待ち割当て：到達する定義を持つマクロタスクの状況は A〜E のいずれかである．どのマクロタスクをプリロードの対象とするかは次の 2 つの方法が考えられる．

(a) 割当て時点で A の状況のもの．もしくは割当て後 A の状況に推移したもの．

(b) 割当て時点で A か C の状況のもの．もしくは割当て後 A か C の状況に推移したもの．

(3) 制御依存待ち割当て：到達する定義を持つマクロタスクの状況とプリロードの対象は通常割当てと同じ．

(4) 制御・データ依存待ち割当て：到達する定義を持つマクロタスクの状況とプリロードの対象はデータ依存待ち割当てと同じ．

上記 (1) は従来の共有メモリシステム上でのマクロデータフロー処理でも実現されており，この場合には，プリロード（共有メモリからローカルメモリなどへのロード）の対象となるマクロタスクの状況が A であるか否かを判断する必要はない．

(2)〜(4) は (1) より積極的なプリロードで，データ到着条件を用いることにより，プリアサインを前提として可能となるものである．なお，(2) の (b)，(3)，(4) の (a) と (b) でのプリロードでは，MTi が実行されなかったり，MTj のデータ到達条件が成立しなかったりして，プリロードした変数の値が不要となってしまう場合がある．このようなプリロードを「投機的プリロード」と呼ぶこととする．

表 5 システム構成
Table 5 System configuration.

	PC クラスタ A	PC クラスタ B
OS	Red Hat Linux 6.2 Kernel 2.2.19	Red Hat Linux 7.1 SCore Version 4.2.1
CPU	Pentium3 800 MHz x2	Pentium3 866 MHz x2
メモリ	128 MB	1 GB
NIC	Ether-100 base	Myrinet-2000
MPI	mpich1.2.1	mpich1.2.0

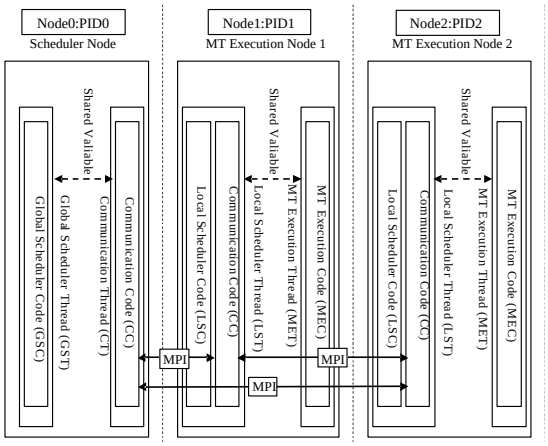

図 2 分散メモリシステム上での実装例
Fig. 2 Example of implementation.

5. 分散メモリシステム上での予備的実装

データ到達条件を用いた分散メモリシステム上でのMDF の動作検証と，マクロタスク割当て・データ通信指示やデータ到達条件評価のためのオーバヘッドにより並列処理効果が阻害されるような状況にならないかどうかを知るために，簡単な実装とベンチマークプログラムによる動作検証を行った．

5.1 実装の概要

対象としたシステムは各ノードに 2 つのプロセッサを備える 2 種類の SMP クラスタ（**表 5**）である．システム A とシステム B ではメモリ量が異なることに加え，ネットワークの性能が大きく異なる．この上で MPI と p-thread を用いて実装した．

ノードのうち 1 つをスケジューラノード（SN），その他のノードをマクロタスク実行ノード（EN）とする．各ノードでは 1 つのプロセスを起動する（**図 2**）．

SN では，全ノードの並列実行を統制するグローバルスケジューラコードと他ノードとの通信を行う通信コードをそれぞれ個別のスレッドとして実行する．

SN のグローバルスケジューラコードでは以下の機能を行う．

(1) 「マクロタスク終了」と「分岐方向決定」の通知にともなう実行開始条件とデータ到達条件の更新

(2) 実行開始条件が成立したマクロタスクのプロセッサへの割当て決定およびマクロタスク実行指示とデータ送信・受信指示の発行

(3) プリアサイン可能なマクロタスクのプロセッサへの割当て決定およびデータ送信・受信指示の発行

SN の通信コードでは以下の機能を行う．

(1) EN からの「マクロタスク終了」と「分岐方向決定」の通知の受信

(2) EN へマクロタスク実行指示とデータ送信・受信指示の送信

EN では，通信とマクロタスク処理を統制するローカルスケジューラコードと他ノードとの通信を行う通信コードをまとめて 1 つのスレッドとして実行し，マクロタスクの処理を行うマクロタスク処理コードを 1 つのスレッドとして実行する．

EN のローカルスケジューラコードと通信コードは以下の機能を行う．

(1) SN からのマクロタスク実行指示とデータ送信・受信指示の受信

(2) マクロタスク処理コードにデータが揃ったマクロタスクの実行指示

(3) SN への「マクロタスク終了」と「分岐方向決定」の通知の送信

(4) 他の EN とのデータ送信・受信

EN のマクロタスク処理コードは以下の機能を行う．

(1) ローカルスケジューラコードからのマクロタスク実行指示の受付

(2) マクロタスクの実行

(3) (2) にともなう「マクロタスク終了」と「分岐方向決定」の通知の発行

ノード間で行われる通信は，(1) SN から EN へのマクロタスク実行の指示，(2) SN から EN へのデータの送信・受信の指示，(3) EN から SN へのマクロタスクの実行にともなう事象の通知，(4) EN 間のデータの送信・受信の 4 種類である．**図 3** にマクロタスクの割当てと実行の過程を，**図 4** にデータ通信指示と実行の過程を示す．

プリロードに関しては，従来のプリロード（4.2 節の (1)）のみを行うものと，より積極的なプリロード（4.2 節の (2)-(a)）を行う方式を実装した．

5.2 動作検証

動作の検証に用いたプログラムは SPEC CFP95 の tomcatv（データサイズ 257 × 257）と swim（デー

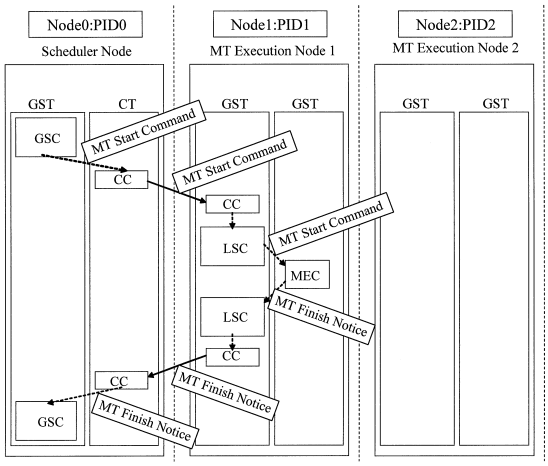

図 3 マクロタスクの割当て実行
Fig. 3 Macrotask assignment and execution.

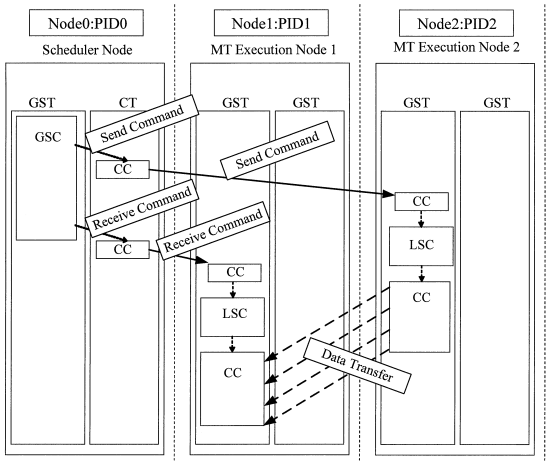

図 4 マクロタスク間データ通信
Fig. 4 Data transfer between macrotasks.

タサイズ 505 × 505）の比較的小規模のプログラムで，この 2 つのプログラムを人手で並列化（マクロタスク生成と並列性検出）し並列実行した．またその実行時間を測定した．

並列化に際しては tomcatv，swim とも主ループの内側の処理をマクロタスクに分割し MDF を行うようにした．それぞれのプログラムの主ループの内側の処理を 4 並列用にマクロタスク分割した場合のマクロフローグラフを**図 5** と**図 6** に示す．マクロタスク（図中の方形）の主要なものは主ループ内部のループを分割または融合したものである．また，マクロタスク間のデータ授受（実線矢印）は配列変数によるものである．図 5 と図 6 では主ループの各イタレーション内でのデータ授受だけを示している．実際にはこのほか

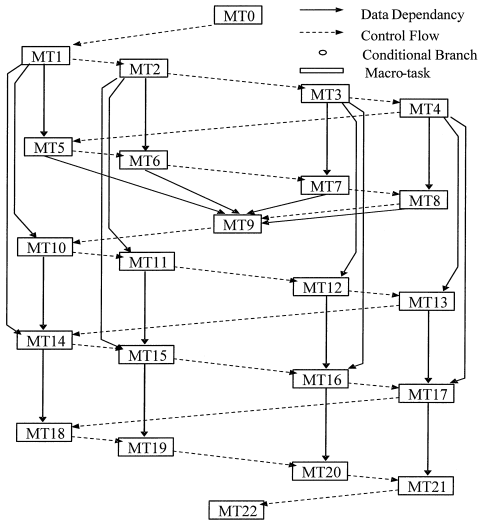

図 5 tomcatv のマクロフローグラフ
Fig. 5 Macro-flow-graph of tomcatv.

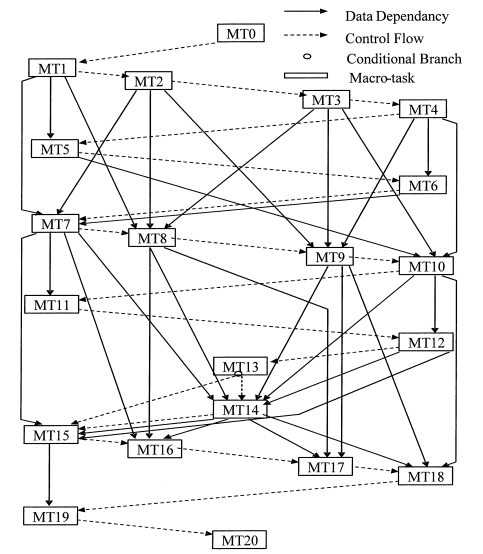

図 6 swim のマクロフローグラフ
Fig. 6 Macro-flow-graph of swim.

に主ループのイタレーション間にまたがるデータ授受が存在する．

この並列化した tomcatv と swim を表 5 のそれぞれのシステムにおいて，逐次の場合，EN 数が 2 の場合および 4 の場合において並列実行しその実行時間を測定した．測定結果を**表 6** と**表 7** に示す．表中「PL なし」とは従来のプリロード（4.2 節の (1)）のみを行うもので，「PL あり」はより積極的なプリロード（4.2 節の (2)-(a)）のプリロードを行うものである．

どの場合も矛盾なく並列実行できていることが確認された．

表 6 実行時間（tomcatv）[sec]
Table 6 Execution time (tomctv) [sec].

システム	ノード数	実行時間		実行時間比	
		PL 無	PL 有	PL 無	PL 有
A	1	100.2	100.2	1.00	1.00
	2	65.8	66.0	0.66	0.66
	4	104.0	103.6	1.04	1.03
B	1	49.3	49.3	1.00	1.00
	2	30.2	30.1	0.61	0.61
	4	15.9	16.4	0.32	0.33

表 7 実行時間（swim）[sec]
Table 7 Execution time (swim) [sec].

システム	ノード数	実行時間		実行時間比	
		PL 無	PL 有	PL 無	PL 有
A	1	75.8	75.8	1.00	1.00
	2	55.1	51.6	0.73	0.68
	4	41.4	38.5	0.54	0.52
B	1	43.0	43.0	1.00	1.00
	2	22.5	22.4	0.52	0.52
	4	10.9	10.9	0.25	0.25

並列処理効果に関しては，システム B 上での swim においてほぼリニアな結果が得られている．これより，MDF を分散メモリシステム上で実装するために発生するマクロタスク割当てやデータ通信指示およびデータ到達条件評価のためのオーバヘッドはさほど大きいものではないと判断できる．

システム A 上では，swim においても EN 数が 4 での処理時間が逐次の場合の約 1/2 程度にとどまっている．これは通信性能がシステム B より劣っていることにより生じているものと考えられる．また，tomcatv においては，システム B では EN 数が 4 での処理時間が逐次の場合の約 1/3 を達成しているにもかかわらず，システム A では逐次処理より実行時間が長くなってしまっている．これは EN 数が増える（マクロタスクの分割数が増える）に従ってマクロタスク間での通信の総量が増えており，その影響が通信性能の低いシステム A で顕著になってしまっているためであると考えられる．

4.2 節の (2)-(a) のプリロードの効果に関しては，本プログラムでは大きな効果が得られていない．これは，(1) tomcatv ではこのプリロードの対象となりうるプロセッサ間データ通信がないこと，(2) swim においてもマクロタスクの処理粒度に比べてデータ通信送量が小さいこと，などの理由によると考えられる．

6. 関連研究

MDF を共有メモリシステム上で実現する場合に，プロセッサのローカルメモリやローカルキャッシュを有効に用いることを目的とし，マクロタスク間データ依存解析をもとにマクロタスクの再分割やスケジューリングを行うことによりデータローカリティを向上させる最適化が可能であり，一部スタティックに割り当てられるマクロタスク間でこのような最適化を行うための手法が提案されている[8),9)]．

データ到達条件による通信管理方式は，分散メモリシステム上での並列処理で必要となる通信を可能な限りコンパイル時の解析によって支援しようというアプローチである．これは，SDSM において，同期区間によって明示的に並列性が記述されているプログラムを対象として，一貫性制御のための実行時オーバヘッドを削減するためにコンパイラの支援によって一貫性制御コードを生成する Shasta などに代表される[10)~12)]アプローチと共通した考え方である．

7. おわりに

本論文では実行開始条件を用いたマクロデータフロー処理（MDF）を分散メモリシステム上でソフトウェア分散共有メモリ（SDSM）を用いずに実現する際に必要となる「データ到達条件」を提案した．データ到達条件を実行時に検査することにより，マクロタスクに到達する定義のうちどの定義による値を使用するかを実行時に動的に決定し，マクロタスク間のデータ授受の関係を求めることができる．またデータ到達条件を用いることにより，新たに可能となる積極的なプリロード方式「投機的プリロード」を示した．データ到達条件を用いた予備的な実装を行い，分散メモリシステム上での MDF の実装においてもマクロタスク割当てやデータ通信送指示およびデータ到達条件評価のためのオーバヘッドは問題となるほど大きくないことが確認できた．一方，現状では大規模なプログラムの実行ができないため，積極的なプリロードの効果を示すには至っていない．

5 章で述べた予備的な実装では実装を容易にするため，(1) スケジューラ用にノードやプロセッサを占有している，(2) 扱えるデータサイズにも制限がある，など実用的な面からは不十分な点があるため，今後これらを改良する必要がある．現在 MPI と OpenMP を用いた実装，ならびに本方式によるコードを逐次プログラムから生成するコンパイラの開発も進めている．今後これらを用いてノード数とノード内プロセッサ数を増やした環境での評価を行う予定である．これにより分散メモリシステム上での MDF におけるプロセッサ間通信の特質などを明らかにしていきたい．

本論文でその可能性を示したプリアサインとプリロードの方式に対しての実装とその有効性に関する研究は今後の課題である．

また，MDF に限らず，たとえば SDSM での一貫性制御コード生成にデータ到達条件を用いることにより一貫性制御の最適化を実現することなど，データ到達条件の並列プログラム解析への応用も今後の課題としたい．

なお，今後 MDF の SDSM 上での実装も進め，各種 SDSM 方式と MDF との親和性の検討や SDSM を用いない方式との比較を行う必要があると考える．この際，SDSM を用いない実装での評価によって得られる分散メモリシステム上での MDF におけるプロセッサ間通信の特質を SDSM 上での効率的な実装に活かせるものと考える．

謝辞 本研究の一部は文部科学省科学研究費（基盤 C，2，12680336）によって行われた．

参 考 文 献

1) 笠原博徳，小幡元樹，石坂一久：共有メモリマルチプロセッサシステム上での粗粒度タスク並列処理，情報処理学会論文誌，Vol.42, No.4, pp.910–920 (2001).
2) 本多弘樹，岩田雅彦，笠原博徳：Fortran プログラム粗粒度タスク間の並列性検出手法，電子情報通信学会論文誌，Vol.J73-D-1, No.12, pp.951–960 (1990).
3) Girkar, M. and Polychronopoulos, C.D.: Optimization of Data/control Conditions in Task Graphs, *Proc. 4th Workshop on Languages and Compilers for Parallel Computing*, pp.Z1–Z15 (1991).
4) 笠原博徳，合田憲人，吉田明正，岡本雅巳，本多弘樹：Fortran マクロデータフロー処理のマクロタスク生成手法，電子情報通信学会論文誌，Vol.J75-D-I, No.8, pp.511–525 (1992).
5) Ferrante, J.F., Ottenstein, K.J. and Warren, D.J.: The Program Dependence Graph and Its Use in Optimization, *ACM Trans. Prog. Lang. and Sys.*, Vol.9, No.3, pp.319–349 (1987).
6) 本多弘樹，合田憲人，岡本雅巳，笠原博徳：Fortran プログラム粗粒度タスクの OSCAR における並列実行方式，電子情報通信学会論文誌，Vol.J75-D1, No.8, pp.526–535 (1992).
7) Aho, A.V., Sethi, R. and Ullman, J.D.: *Compilers Princeiples, Techniques and Tools*, Addison-Wesley (1988).
8) Yoshida, A., Maeda, S., Fujimoto, K.. and Kasahara, H.: Data-Localization for Macro-Dataflow Computation Using Static Macrotask Fusion, *Proc. 5th Workshop on Compilers for Parallel Computers*, pp.440–453 (1995).
9) 中野啓史，石坂一久，小幡元樹，木村啓二，笠原博徳：キャッシュ最適化を考慮したマルチプロセッサシステム上での粗粒度タスクスタティックスケジューリング手法，情報処理学会研究報告，Vol.2001, No.76 (ARC-144), pp.67–72 (2001).
10) Scales, D.J., Gharachorloo, K. and Thekkath, C.A.: Shasta: A Low Overhead, Software-Only Approach for Supporting Fine-Grain Shared Memory, *Proc. 7th Int'l. Conf. on ASPLOS*, pp.174–185 (1996).
11) 丹羽純平，稲垣達氏，松本 尚，平木 敬：非対称分散共有メモリにおける最適化コンパイル技法の評価，情報処理学会論文誌，Vol.39, No.6, pp.1729–1737 (1998).
12) 佐藤茂久，草野和寛，佐藤三久：OpenMP 向けコンパイラ支援ソフトウェア DSM，情報処理学会論文誌：ハイパフォーマンスコンピューティングシステム，Vol.42, No.SIG 12 (HPS 4), pp.788–801 (2001).
13) 上田哲平，本多弘樹，弓場敏嗣：分散メモリシステム上でのマクロデータフロー処理の実現，情報処理学会研究報告，Vol.2002, No.22 (ARC-147/HPS-89:HOKKE), pp.203–208 (2002).

(平成 14 年 1 月 28 日受付)
(平成 14 年 5 月 16 日採録)

本多 弘樹（正会員）

1984 年早稲田大学理工学部電気工学科卒業．1991 年同大学大学院理工学研究科博士課程修了．1987 年より同大学情報科学研究教育センター助手．1991 年より山梨大学工学部電子情報工学科専任講師．1992 年より同助教授．1997 年より電気通信大学大学院情報システム学研究科助教授．並列処理方式，並列化コンパイラ，並列計算機アーキテクチャ，グリッド等の研究に従事．工学博士．電子情報通信学会，IEEE-CS，ACM 各会員．

上田 哲平

2000 年電気通信大学情報工学科卒業．2002 年同大学大学院情報システム学研究科修士課程修了．同年株式会社日立製作所入社．在学時代には分散メモリシステムにおける並列処理に関する研究に従事．工学修士．

深川 保

2001 年電気通信大学電子情報学科卒業．同年より同大学大学院情報システム学研究科修士課程在学中．分散メモリシステムにおける OpenMP による粗粒度並列処理に関する研究に従事．

弓場 敏嗣（正会員）

1966 年神戸大学大学院工学研究科修士課程修了．（株）野村総合研究所を経て，1967 年通商産業省工業技術院電気試験所（現，産業技術総合研究所）に入所．以来，計算機のオペレーティングシステム，見出し探索アルゴリズム，データベースマシン，データ駆動型並列計算機等の研究に従事．その間，知能システム部長，情報アーキテクチャ部長等を歴任．1993 年より，電気通信大学大学院情報システム学研究科教授．並列処理の科学技術一般に興味を持つ．工学博士．電子情報通信学会，日本ソフトウェア科学会，日本ロボット学会，ACM，IEEE-CS 各会員．