階層的マクロデータフロー処理のインプリメント手法

岡本　雅巳+　合田　憲人+　尾形　航+　吉田　明正+　本多　弘樹++　笠原　博徳+

+早稲田大学理工学部　++山梨大学工学部

2D−9

1．はじめに

マルチプロセッサシステム用Fortran並列化コンパイラでは，従来，シーケンシャルループ内部やループ外の基本ブロック内部での細粒度並列性，あるいは，ループ，サブルーチンおよび基本ブロック間の粗粒度並列性を，効果的に利用することが出来なかった．そこで，筆者らは，シーケンシャルループ，基本ブロック内部の並列性を使用するためにスタティックスケジューリングを用いたステートメントレベルの細粒度並列処理[2]，Do-all等のループイタレーションをタスクとする従来の中粒度並列処理，ループあるいはサブルーチンなどをタスクとする粗粒度タスク（マクロタスク）の並列処理（マクロデータフロー処理）[3]とを階層的に組み合わせたマルチグレイン並列処理手法[4]を提案している．

しかし，従来までに実現してきたマルチグレイン並列処理では，マクロデータフロー処理によりプロセッサクラスタに割り当てられた大規模シーケンシャルループやサブルーチンに対して，クラスタ内部でマクロデータフロー処理を階層的に適用することができなかった．本稿では，このような階層的マクロデータフロー処理のインプリメント手法について述べる．

2．ＯＳＣＡＲのアーキテクチャ[1]

今回，階層的マクロデータフロー処理のインプリメントを行ったマルチプロセッサシステムＯＳＣＡＲについて説明する．ＯＳＣＡＲは図１に示すように各々がローカルメモリと分散共有メモリを持つ１６台のRISCプロセッサ(PE)，１台のコントロール&I/Oプロセッサ(CIOP)と３つの集中型共有メモリ(CM)を３本のバスで接続したマルチプロセッサシステムである．ＯＳＣＡＲは，全てのプロセッサが共有メモリに平等に結合された構成となっているが，各バスのバリア同期機構を用い，１から３プロセッサクラスタ(PC)までのマルチプロセッサクラスタシステムとして使用することができる．また，３階層以上の階層的マクロデータフロー処理を行う場合には，各PCを構成するPEをさらに複数のサブプロセッサクラスタに分割し，階層型プロセッサクラスタシステムをシミュレートする．

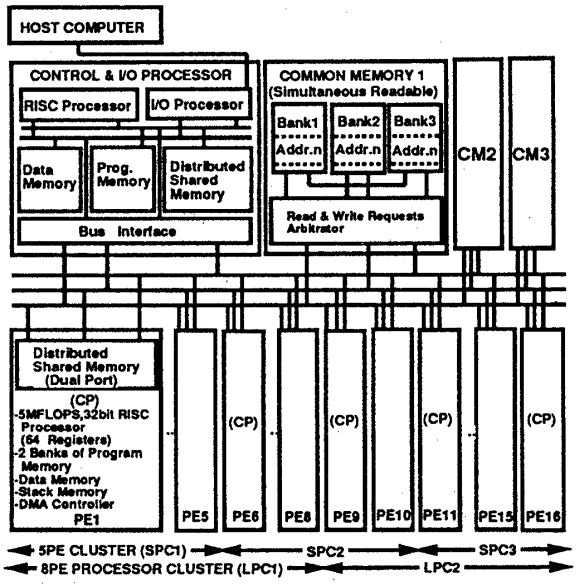

図１．ＯＳＣＡＲのアーキテクチャ

3．階層型マクロデータフロー処理のインプリメント

ＯＳＣＡＲ Fortranコンパイラではマクロデータフロー処理を主に次のような手順で実現している．

1)マクロタスク生成
2)マクロフローグラフ生成
3)マクロタスクグラフの生成
4)ダイナミックスケジューリングルーチンの生成

ここではステップ1)-3)でマクロタスク間の並列性を抽出する過程，および4)に関連しマクロタスクの並列実行方式についての説明をする．

3.1　マクロタスク生成

Fortranプログラムは，次のような３種のマクロタスク（ＭＴ）に分類される．

1) ＢＰＡ(Block of Pseudo Assiginment statement)
　基本ブロックまたは複数基本ブロックを融合したブロック
2) ＲＢ(Repetetion Block)
　最外側ナチュラルループ
3) ＳＢ(Subroutine Block)
　インライン展開が有効に適用できないサブルーチン

マルチグレイン並列処理におけるマクロデータフロー並列処理では，これらのマクロタスクは，コンパイラが生成するダイナミックスケジューリングルーチンにより実行時にプロセッサクラスタに割り当てられる．また，それらのマクロタスクは，プロセッサクラスタ内のPEにより，基本ブロック（ＢＰＡ）あるいは小規模シーケンシャルループ（ＲＢ）であれば細粒度で，Do-allループあるいはDo-across ループ（ＲＢ）であれば中粒度で並列処理される．しかし，マクロタスクが，多重に不完全ネストされた大規模シーケンシャルループ（ＲＢ）の場合や，大規模サブルーチンブロック（ＳＢ）である場合には，そのマクロタスク内部を階層的にサブマクロタスクに分割し，プロセッサクラスタ内のサブプロセッサクラスタ上で階層的にマクロデータフロー処理を適用する必要がある．

3.2　マクロフローグラフの生成

次にコンパイラはマクロタスク間のコントロールフロー及びデータ依存を解析し，図２のようなマクロフローグラフを生成する．階層的マクロデータフロー処理を適用すべきＲＢやＳＢに対しては，内部のサブマクロタスクをノードとするサブマクロフローグラフを階層的に生成する．この際，最上位階層のマクロフローグラフやＳＢ内側で生成

An Implementation Scheme of Hierarchical Macro-dataflow Computation
Masami OKAMOTO+, Kento AIDA+, Wataru OGATA+, Akimasa YOSHIDA+, Hiroki HONDA++, Hironori KASAHARA+
+WASEDA Univ.　++YAMANASHI Univ.

されるサブマクロフローグラフは，常に無サイクル有向グラフとなる．また，ＲＢ内側で生成されるサブマクロフローグラフは，各ＲＢが最外側ナチュラルループであるという定義から，サブマクロフローグラフの出口ノードから入口ノードへのバックエッジが存在するが，それ以外のバックエッジはサブＲＢ内部に含まれる．サブマクロタスク間の並列性を抽出（最早実行可能条件解析[5]）する際には，サブマクロフローグラフは無サイクル有向グラフでなければならない．そのため，繰り返し条件の成立を示すノード(repeatMT)とイタレーションの終了を示すノード(exitMT)をサブマクロフローグラフの出口に仮想的に配置し，バックエッジを除去し最早実行可能条件解析を行なう．この時，exitMTには全てのサブマクロタスクからデータ依存があるものとして解析を行なう．

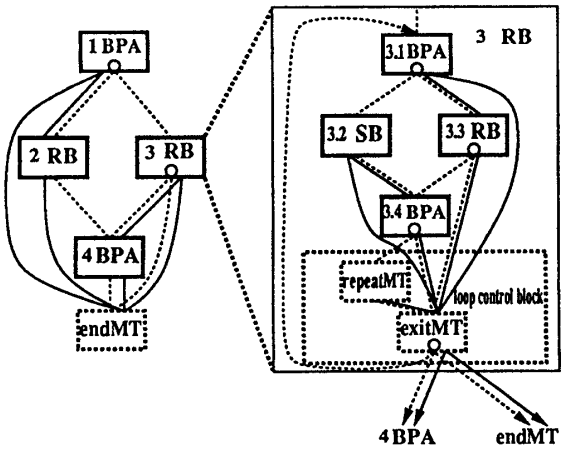

図２．マクロフローグラフ

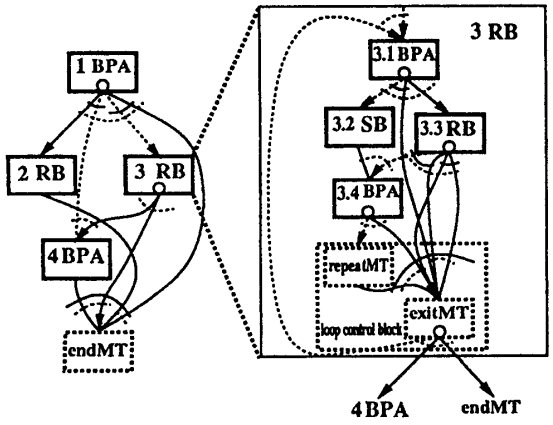

図３．マクロタスクグラフ

3.3 マクロタスクグラフの生成

求められた各マクロタスクの最早実行可能条件を図３のようなグラフで表現したものがマクロタスクグラフである．階層的マクロデータフロー処理を適用すべきＲＢやＳＢに対しては，サブマクロタスクグラフを階層的に生成する．

ＲＢ３のように，内部で階層的にマクロデータフロー処理を適用する場合には，現イタレーションの終了条件はexitMTの実行可能条件で表わし，次イタレーションの実行可能条件はrepeatMTへの分岐と，exitMTの実行可能条件の論理積に表わすことができる．

3.4 ダイナミックスケジューリングによるマクロタスクの実行

マクロデータフロー処理では，実行時不確定性（実行時に確定できないマクロタスクの処理時間の変動や，条件分岐方向）に対処するためマクロタスクは実行時にダイナミックスケジューリングによってPCに割り当てられる．ダイナミックスケジューリングアルゴリズムとしては，スタティックスケジューリングアルゴリズムであるCP法を拡張したDynamic-CP法を使用している．このダイナミックスケジューリングではスケジューリングオーバーヘッドを最小化するためＯＳコールなどを用いず，コンパイラが各プログラム専用のダイナミックスケジューリングルーチンを生成する．このスケジューリングルーチンは任意のPE上で実行でき，そのPEはダイナミックスケジューラ(CP: Control Processor)として動作する．

ある階層のプロセッサクラスタ(PC)内の全PEは，上位階層のCPから次に実行すべきマクロタスクの情報を受け取ると，一斉にそのマクロタスクのためのマシンコードが格納されているアドレスにジャンプする．その際，１台のPEがスケジューリングルーチンを実行し，そのプロセッサがその階層のCPとして動作する．次にそのCPはダイナミックスケジューリングを開始し，初期実行可能マクロタスク（その階層におけるマクロタスク）をPCに割り当てる．割り当てられたマクロタスクの実行が終了すると，PCはCPに対してマクロタスクの実行終了信号を送信する．またマクロタスク中の条件分岐文の実行により分岐方向が確定すると，PCはCPに対して分岐信号を送信する．CPは受信した終了信号・分岐信号から実行可能なマクロタスクを求め，優先度の高い実行可能マクロタスクを，アイドルＰＣへ，あるいは先行割当を行なう場合には負荷の軽いPCへ割り当てる．

ＲＢ内部でダイナミックスケジューリングを行なっているCPは，バックエッジ（次イタレーションの先頭）への分岐信号と，現イタレーションで実行する全マクロタスクの実行の終了信号を受信すると，次イタレーションのダイナミックスケジューリングを開始するために必要な初期化等を行ない，次イタレーションのためのダイナミックスケジューリングを開始する．また，マクロタスクのスケジューリングを終了したCPは終了信号を上位CPに送り処理を終了する．

４．まとめ

本稿では，マクロタスクの内側を複数のサブマクロタスクに分割し，階層的にマクロデータフロー処理を行う階層的マクロデータフロー処理のインプリメント手法について述べた．　今後の課題としては，階層的マクロデータフロー処理を適用するマクロタスクの選択及び，その階層の深さを自動的に決定する手法と，最適なプロセッサクラスタリング手法の開発などが挙げられる．

参考文献

[1]笠原,成田,橋本，ＯＳＣＡＲのアーキテクチャ，信学論J71-D(8),8 1988
[2]H.Kasahara,H.Honda,S.Narita, "Parallel Processing of Near Fine Grain Tasks Using Static Scheduling on OSCAR" in Proc. IEEE ACM Supercomputing '90, Nov.1990.
[3]笠原，並列処理技術，コロナ社,1991-06.
[4]H.Kasahara, H.Honda, S.Narita, "A Multi-Grain Parallelizing Compilation Scheme for OSCAR",Proc. 4th. Workshop on Language and Compilers for Parallel Computing,Aug. 1991.
[5]本多,岩田,笠原，Fortranプログラム粗粒度タスク間の並列性検出手法，信学論,J73-D-I(12),12 1990