2D−6

マルチプロセッサスーパーコンピュータ上でのFORTRANプログラムのマクロデータフロー処理

合田　憲人+　岡本　雅巳+　尾形　航+　本多　弘樹++　笠原　博徳+　成田　誠之助+

+早稲田大学理工学部　++山梨大学工学部

1．はじめに

高性能プロセッサを比較的少数結合した主記憶共有型マルチプロセッサ（マルチプロセッサスーパーコンピュータ）上での従来のFORTRANプログラムの並列処理では，マクロタスキング（サブルーチン並列処理）とマイクロタスキング（ループ並列処理）のみが行われていた[4][7]．また，プログラム中からの粗粒度の並列性の抽出は，多くの場合ユーザーにゆだねられていた．

本稿では，マルチプロセッサスーパーコンピュータ上でのFORTRANプログラムのマクロデータフロー処理手法[1][5][9]を提案する．本手法では，コンパイラがプログラムの粗粒度タスク（マクロタスク）への分割，マクロタスク間の並列性抽出，各Fortranプログラム専用のダイナミックスケジューリングコードの生成等を自動的に行うため，低オーバーヘッドで効率の良い並列処理を行うことができる．

2．マクロデータフロー処理手法[1][5][9]

本節では，Fortranプログラムのマクロデータフロー処理について概説する．本稿におけるマクロデータフロー処理とは，サブルーチン，ループ，基本ブロック等の粗粒度タスク間の並列性を利用する並列処理手法である．

本マクロデータフロー処理手法では，まずはじめにFortranプログラムを，マクロタスクと呼ぶ粗い粒度をもつタスクに分割する．マクロタスクは，基本ブロック（BB）あるいは複数の基本ブロックからなるブロック（BPA），繰り返しブロック（RB），サブルーチンブロック（SB）から構成される．RBは，DOループまたはバックワードブランチによって生成されるループであり，より厳密には最外側ナチュラルループを意味する．サブルーチンに関しては，基本的に可能な限りインライン展開を行うが，コード長等を考慮して，効率的にインライン展開できない場合には，そのサブルーチンは1つのマクロタスク（SB）として定義する．この場合，SBと他のマクロタスクとの並列性を最大限に引き出すためには，強力なインタープロシージャ解析が必要となる．

プログラムの分割終了後，マクロタスク間の制御フロー，データフローを解析することにより，図1のようなマクロフローグラフを生成する．次にマクロフローグラフからマクロタスク間の制御依存，データ依存関係を解析することによりマクロタスク間の並列性を最大限に抽出し，各マクロタスクの最早実行開始条件[6][9]を表現した図2のようなマクロタスクグラフを生成する．さらに生成されたマクロタスクグラフの情報より，マクロタスク間の並列性を高めたり，データ転送およびスケジューリングオーバーヘッドを最小化するために，マクロタスクの再分割・融合[8][9]を行う．

各マクロタスクは，マクロタスクグラフの情報をもとにして，条件分岐や処理時間の変動に対処するために，マルチプロセッサスケジューリング理論を拡張して開発されたダイナミックスケジューリング手法[3]により，実行時にプロセッサに割り当てられる．一般的なOSコールによるダイナミックスケジューリングは実行時のオーバーヘッドが大きいが，本手法では，コンパイラにより生成されたスケジューリングコードがダイナミックスケジューリングを行うため，オーバーヘッドを相対的に小さく抑えることができる．

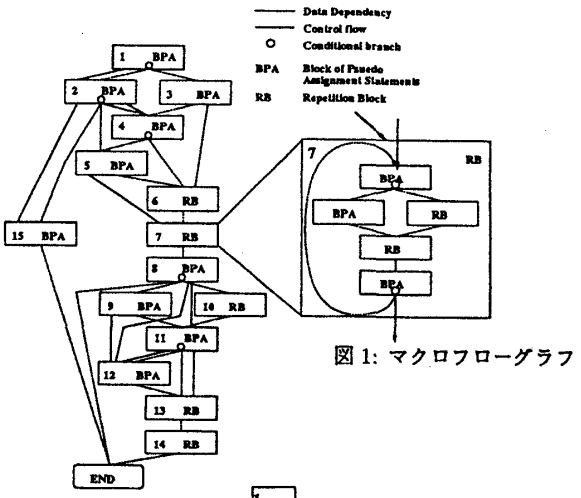

図1: マクロフローグラフ

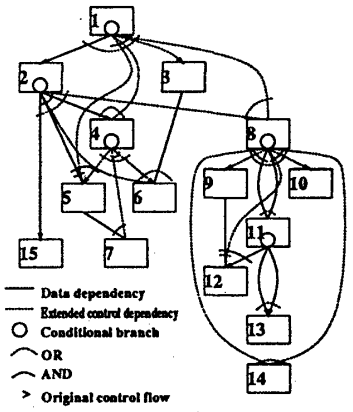

図2: マクロタスクグラフ

Macro-dataflow Computation of FORTRAN Program on Multi-processor Super Computer

Kento AIDA+, Masami OKAMOTO+, Wataru OGATA+, Hiroki HONDA++, Hironori KASAHARA+, Seinosuke NARITA+

+WASEDA Univ., ++YAMANASHI Univ.

3．マクロデータフロー処理実行方式

3．1　ターゲットアーキテクチャ

本稿が対象とするマルチプロセッサスーパーコンピュータは，図3に示すように，各々がベクトルおよびスカラユニットとローカルメモリを持つプロセッサ4台を共有メモリに接続した構成である．

3．2　スケジューリングコード

コンパイラが生成するスケジューリングコードは，図3のシステム用にプログラムコード中に埋め込まれており，各プロセッサが割り当てられたタスクの実行を終了する度に，スケジューリングコードの実行を行う．このスケジューリングコードを生成する際には，スケジューリングコードを単プロセッサに実行させる集中型スケジューラ方式[5][8][9]も考えられるが，今回対象としているマルチプロセッサスーパーコンピュータのプロセッサ数が4台程度と少ないので，上述の分散型スケジューラ方式をとっている．

スケジューリングコードの機能は，マクロタスクの実行開始条件の管理とマクロタスクの実行起動である．図3に示すように，各プロセッサはこのスケジューリングコードを埋め込まれた同一のプログラムコードを持っている．またPE1は，このプログラムコードに加えて初期化処理のためのコードを持っている．マクロタスク実行管理のためデータ（MT実行管理変数）としては，マクロタスクの状態（非実行可能，実行可能）を管理するマクロタスク実行状態テーブル，マクロタスクグラフのエッジに相当する終了・分岐信号を管理するマクロタスク信号管理テーブル，実行可能なマクロタスクが登録されるレディーMTキューを用意する．これらのMT実行管理変数は，各プロセッサにより共有され，その更新は排他的に行われる．

3．3　マクロタスク実行手順

ここでは，各プロセッサ上のスケジューリングコードがマクロタスクを実行する手順を述べる．

プログラムの実行開始時には，PE1が初期化処理（MT実行管理変数の初期化，プログラム開始時点での実行可能マクロタスクのレディーMTキュー登録）を行う．初期化処理終了後，各プロセッサは，レディーMTキューから実行可能マクロタスク（MTi ）を取り出し，MTi の実行を開始する．MTi の実行中，MTi からの他のマクロタスクへの分岐があれば，それをマクロタスク信号管理テーブルへ通知する．MTi の実行終了後は，MTi の実行終了をマクロタスク信号管理テーブルへ通知し，次にまだ実行可能でないマクロタスクのうち，MTi の実行終了または他のマクロタスクへの分岐が実行開始条件中に含まれるマクロタスクの実行開始条件を調べる．そして実行可能なマクロタスク（MTj ）が存在すれば，マクロタスク状態テーブル上のMTj の状態を実行可能に変更し，MTj をレディーMTキューへ登録する．

以上の処理を各プロセッサがプログラムの終了まで繰り返す．

4．むすび

本稿では，マルチプロセッサスーパーコンピュータ上でのFortranプログラムのマクロデータフロー処理手法を提案した．今後，実マルチプロセッサスーパーコンピュータ上での本手法の性能評価および実用化のための検討を行っていく予定である．

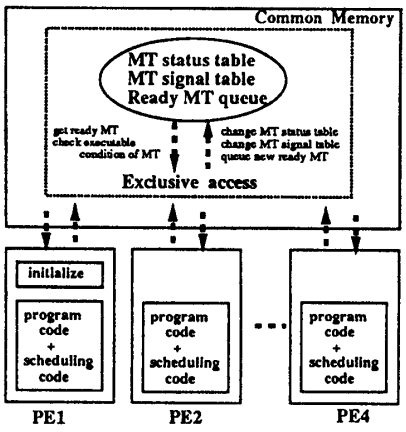

図 3: マクロデータフロー処理実行方式

参考文献

[1] F.Gajski, D.Kuck, D.lawrie, A.Sameh : CEDAR - A Large Scale Multiprocessor, Proc. 1983 International Conference on Parallel Processing, pp114-121, (Aug.1983)

[2] Constantine D. Polychronopoulos : Parallel Programming and Compilers, Kluwer Academic Publishers, (1988)

[3] 岩田，笠原：Fortranマクロタスクグラフのダイナミックマルチプロセッサスケジューリング手法，第36回情報処理学会全国大会3H-8，（昭63-03）

[4] M.Guzzi, D.Padua, J.Hoeflinger, D.Lawrie : Cedar Fortran and Other Vector and Parallel Fortran Dialects, Proc. Supercomputing '88, pp 114-121, (Mar.1988)

[5] 本多，広田，笠原：階層型マルチプロセッサシステムＯＳＣＡＲ上でのFortran並列処理手法，並列処理シンポジウム，JSPP'89, pp251-258（平01-02）

[6] 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D-I, Vol.J73-D-I, No.12, pp951-960, (1990-12)

[7] 笠原：並列処理技術，コロナ社，(1991-06)

[8] 合田，岡本，吉田，本多，笠原：マクロデータフロー処理におけるマクロタスク分割・融合手法，信学技報，CPSY91-30, pp205-212, (1991-07)

[9] H.Kasahara, H.Honda, A.Mogi, A.Ogura, K.Fujiwara, S.Narita : A Multi-Grain Compilation Scheme for OSCAR, Proc. 4th Workshop on Languages and Compilers for Parallel Computing, Santa Clara, (Aug.1991)