5G－7 OSCARアプリケーション専用目的コンパイラにおける超階層マクロデータフロー処理手法

黒田 泰†　田村 光雄‡　前川 仁孝†　笠原 博徳†
†早稲田大学理工学部　‡松下電器産業（株）

1 はじめに

近年、各種科学技術計算の計算時間短縮のために並列計算機の利用が一般的になっている。それに伴い、並列処理に全く興味のないユーザが、並列処理マシンを使用する機会が増えている。これらのユーザにとっては、並列処理のためのプログラムのチューニング、さらには高級言語を使ったプログラミングすらわずらわしい場合も少なくない。このような状況を考慮すると並列処理マシンを使う際に、ユーザには並列処理を意識させず、ユーザが理解しやすく使いやすい方式、例えば各アプリケーションにおいて通常使用されている形態でのグラフィック入力等が望ましいと考えられる。このようなユーザフレンドリな並列システム開発のためには各アプリケーション特有の入力形式より自動的に並列化プログラムを作成し、マルチプロセッサシステム上で効率良い処理を行なうことを可能とするコンパイラの開発が必要となる。

本稿では、このような機能を持った専用目的コンパイラを各アプリケーションに対して開発することを容易にする並列化中間言語処理系について述べるとともに、並列化中間言語で定めるマクロタスク [1] 内部の粗粒度並列性を階層的に利用する超階層型マクロデータフロー処理を提案する。階層型マクロデータフロー処理手法としては、従来 Fortran 自動並列化コンパイラによるマルチグレイン並列処理 [2] などが提案されているが、この方式では実行時スケジューリングオーバーヘッドの最小化のためにループ内外の基本ブロックを異なる階層として取り扱っているため、並列性の抽出に限界があった。これに対し本稿では、ループ内外の基本ブロック、さらにはネストされたループ内基本ブロックも階層 (ループバウンダリ) を越えて同一レベルの粗粒度タスクとしてスケジューリングを行ない、マクロタスクの階層 (ループネストレベル) を越えた粗粒度並列処理を可能とする超階層マクロデータフロー処理を提案する。

2 OSCARのアーキテクチャ

提案する超階層マクロデータフローはマルチプロセッサシステム OSCAR にインプリメントされている。本章では OSCAR のアーキテクチャについて述べる。

OSCAR は 16 台のプロセッサエレメント (PE) と 1 台のコントロール&I/O プロセッサ (CIOP) とが 3 つの集中型共有メモリ (CM) モジュールと各 PE 上の分散共有メモリに 3 本のバスを介して接続された、共有メモリ型のマルチプロセッサシステムである。本超階層マクロデータフロー処理では、OSCAR の PE16 台のうち、コントロールプロセッサ (CP) として PE を 1 台使用し、残りの PE をプロセッサクラスタ (PC) として用いる。また，OSCAR の 3 本のバスには，それぞれ高速なバリア同期を実現するためのハードが用意されているため、最大 3PC のマルチプロセッサクラスタシステムを効率良くシミュレートできる。

A Super Hierarchical Macro-dataflow Computation Scheme in an Application Specific Parallelizing Compiler for OSCAR
Yasushi KURODA†, Mitsuo TAMURA‡, Yoshitaka MAEKAWA†, Hironori KASAHARA†
†Waseda Univ.
‡National/Panasonic

3 アプリケーション専用目的コンパイラ

図 1に専用目的コンパイラの構成を示す。まず、制御系シミュレーション [6]、電子回路シミュレーション [4]、電力潮流

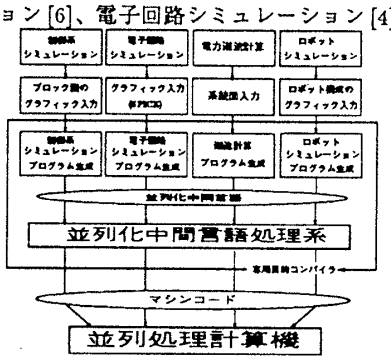

図 1: 専用目的コンパイラ

計算 [5] などの各種アプリケーション特有 (グラフィック入力、系統図入力) の形式で入力を行なう。次にそれぞれの入力に対応した並列中間語プログラムを専用目的コンパイラ内のアプリケーションプログラム生成系が自動生成する。このプログラムはコンパイラ用の専用目的中間言語 (並列化中間言語) で記述される。

各種アプリケーション特有の入力形式により自動的に作成された並列中間語プログラムから、それぞれのマシンコード生成部を別々に作成するのでは、プログラムの開発効率は非常に悪い。そこで、並列化プログラムを専用目的中間言語で記述すれば中間言語処理系を共通にすることが可能となり、新たなアプリケーションへの対応も容易となる。

この並列化中間言語処理系は制御フロー解析、タスク生成、データ依存解析、スケジューリング、並列化マシンコード生成などを行い、マルチプロセッサ・システム OSCAR[3] 上で後述するマクロデータフロー処理を行う。

3.1 並列化中間言語処理系

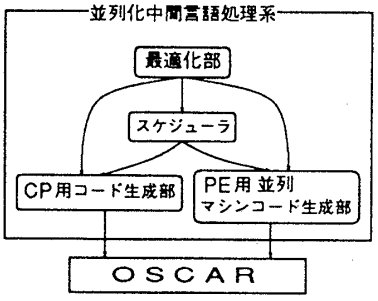

図 2: 中間言語処理系の構成

図 2に並列化中間言語処理系の構成を示す。並列中間語プログラムは「最適化部」で制御フロー解析、データ依存解析、リストラクチャリング、タスク生成を行い、最早実行開始条件解析により得られたマクロタスク間並列性を記述するマクロタスクグラフ [1]、コントロールプロセッサ用マクロタスク実行開始管理テーブル、マクロタスク終了信号管理テーブルなどを出力する。このマクロタスクグラフをもとに「スケジューラ」がプロセッサクラスタ (以下 PC) 内の近細粒度タスクに対するスタティックスケジューリング、またマクロタスク (粗粒度

タスク) のダイナミックスケジューリングのための優先順位づけを行いさらに、「コントロールプロセッサ (以下 CP) 用コード生成部」が CP 上で実行されるダイナミックスケジューリング用コードを生成し、また PC 内における各プロセッサエレメント (以下 PE) 用のマシンコードを「PE 用並列マシンコード生成部」が生成する。

3.2 超階層マクロデータフロー処理

マクロデータフロー処理は粗粒度タスク (マクロタスク) の並列処理を行う手法である。本アプリケーション専用目的コンパイラではマクロタスクの指定はコンパイラ内のアプリケーションプログラム生成系が出力する並列化中間言語で行われる。各マクロタスクはプロセッサクラスタ上で実行されるが、その割当はコントロールプロセッサ CP(PE1 台を専用に使用する) が実行時に行うダイナミックスケジューリングによって行われる。また各プロセッサクラスタは 1 台あるいは複数のプロセッサからなる。プロセッサクラスタ上では近細粒度タスク、粗粒度タスクの 2 つの粒度で並列処理が行われる。

3.2.1 マクロタスクの定義

マクロタスクは 1 つあるいは複数個のマクロタスク要素から成り立っている。マクロタスク要素には基本ブロックの他に、構造化された if 構造、repeat 構造、while 構造の 3 種の制御構造がある。それぞれの構造内には、基本ブロック、if 構造、repeat 構造、while 構造のいずれかを任意回繰り返した複合マクロタスク要素が存在する。

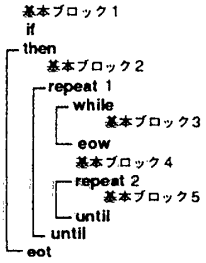

図 3: 並列中間語プログラムの例

例えば図 3のように、if の then 部に基本ブロックと repeat 構造が並列に存在し、さらに repeat 構造の中に while 構造と基本ブロック、repeat 構造が並列に存在する。といったネストされたマクロタスク要素も記述が可能である。これらのマクロタスク要素は専用目的コンパイラ開発者が指定することにより融合されてマクロタスクとなり、指定のないマクロタスク要素はマクロタスク要素 1 つからなるマクロタスクとなる。

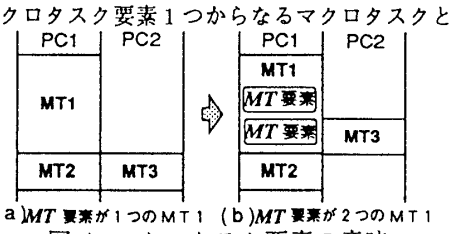

図 4: マクロタスク要素の意味

マクロタスク要素を複数持つマクロタスクの場合、例えば図 4において MT1 から MT2、MT3 へ依存がある場合、MT1 の実行途中に MT3 のデータ依存が満足されたとしても MT1 が実行終了してからでないと MT3 は実行できない。しかし、図 4(b) のように MT1 が 2 つのマクロタスク要素を融合したものであれば、MT3 は MT1 の実行途中に実行開始することができる。これは、マクロタスク要素が終了する毎にマクロタスク要素の実行終了信号を CP に送り、CP がそれによって実行可能となるマクロタスクを PC に割り当てる方式を取っているためである。

3.2.2 超階層マクロデータフロー処理の実行イメージ

次に CP と PC の役割について簡単に説明する。CP は、実行時にマクロタスクを PC に割り当てるダイナミックスケジューリングを行う。これはマクロタスクの並列処理では、条

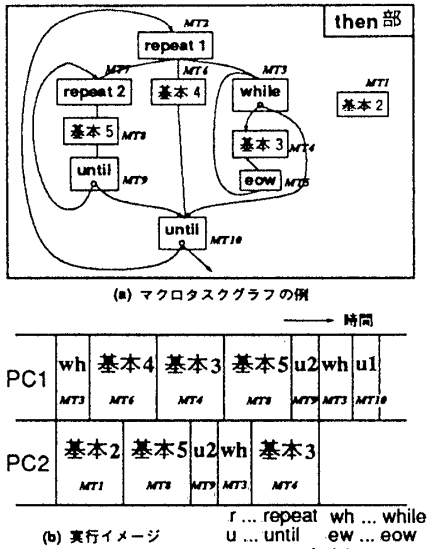

図 5: マクロタスクグラフと実行イメージ

件分岐やループ等の影響による実行時間の変動により、スタティックスケジューリングを有効に利用することが難しいためである。一方、各クラスタは、CP から指定されたマクロタスクを実行し、終了後、CP に実行終了信号を返す処理を行なう。つまり、CP がマクロタスクの実行を PC に行なわせ、PC からそのマクロタスクに分岐がある場合は分岐信号と実行終了信号、ない場合は実行終了信号を受けとると実行可能なマクロタスクを検出し、レディキュー上のマクロタスクを PC に割り当て、実行させる。本方式ではループなどを特別扱いせず全てのマクロタスクの実行を CP が管理する。

例として図 3の then 部を実行する場合のマクロタスクグラフを図 5(a) に、PC2 台で実行した時の実行イメージを図 5(b) に示す。図 5(b) 中の各 PC1、PC2 に書かれているものが実行されたマクロタスクである。図 3の then 部の内、while、repeat2 ループの 2 イタレーションのみを示している。また、repeat、eow はループの同期を取るためのダミーマクロタスクで、CP が管理を行なうだけで PC には割り当てられない。図 5(a) で、while(MT3)、基本ブロック 2(MT7) がまず、PC1、PC2 に割り当てられる。マクロタスクの実行中に repeat2 が CP 内で処理され基本ブロック 5(MT8)、while 内の基本ブロック 3 (MT4) が実行可能となり、until2(MT9) が終了すると基本ブロック 5(MT8)、同様にして eow が処理され、while(MT3) が実行可能となる。このように、while(MT3) とそのループボディにあたる基本ブロック 3(MT4) はループという構造を超えて異なる PC で実行される。

4 まとめ

本稿では各種アプリケーションの自動的な並列処理を目的とした専用目的コンパイラにおける制御構造、ネストによるループを超越した超マクロデータフロー処理手法を提案した。今後、マクロタスク要素を自動的に融合することにより、オーバーヘッドの低減を目指す予定である。

参考文献

[1] 笠原, "並列処理技術", コロナ社,1991-06.
[2] H.Kasahara, H.Honda, S.Narita,"A Multi-Grain Compilation Scheme for OSCAR",Proc.4th Workshop on Languages and Compilers for Parallel Computing, Aug. 1991.
[3] 笠原等, "OSCAR のアーキテクチャ", 信学論 J71-D,8(1988-08)
[4] 前川等, "近細粒度タスクを用いた電子回路シミュレーションの並列処理", 並列処理シンポジウム JSPP'92,pp453-460
[5] 中野等, "マルチプロセッサシステム上での非線形方程式求解の並列処理", 電気学会論文誌,vol133-c,pp947-954,No11,1993
[6] 山本等, "連続・離散時間制御システムシミュレーションの並列処理", 電気学会論文誌,vol133-c,pp939-946,No11,1993