1L－8

分散環境をターゲットとした自動並列化コンパイラ及び実行環境

首藤 一幸　菅原 健一　浜中 征志郎　村岡 洋一
compiler@muraoka.info.waseda.ac.jp
早稲田大学　理工学部

1　はじめに

network oriented　本研究の目的は、ネットワークで接続された計算機群を対象とした自動並列化コンパイラの作成である。対象となる計算機群は、ベクトル計算機、マルチプロセッサなど計算資源一般を含む。

task and data parallelism　システムは、基本的に階層的なタスク並列処理を行なう。また並列計算機などネットワークが低遅延であることが期待できる場所では、中粒度の並列処理を行なう。

ただし、現段階ではプログラムの分割は一度行なわれるだけであり、中粒度の並列処理は未サポートである。現在、FORTRAN を入力言語とし、TCP/IP をサポートした UNIX 互換 OS から仮想計算機を構成できるシステムが稼働している。

本稿で構想、設計を述べる。

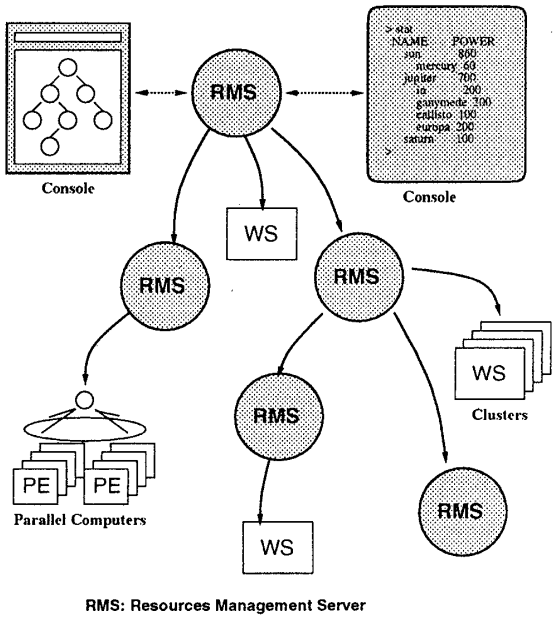

図 1: システムの構成

2　システムの構成

ネットワーク上に階層的に配置された資源管理サーバとプロセッサがツリー (サーバツリー) をなす。ツリーの根に与えられたプログラムは、階層構造により再帰的に分割 (現状では再帰的には分割されない)、分散されていく。このサーバとコンソールと呼ばれるプログラムが仮想計算機を構成する。

2.1　中間表現

プログラムは一貫して１つの中間表現 (図 2) で表される。これは階層的なフロー、タスクグラフであり、データ、制御依存情報など、並列化コンパイラに必要な情報を保持できる。依存解析、プログラムの分割、スケジューリングなどの操作はすべてこの中間表現に対して行なわれる。

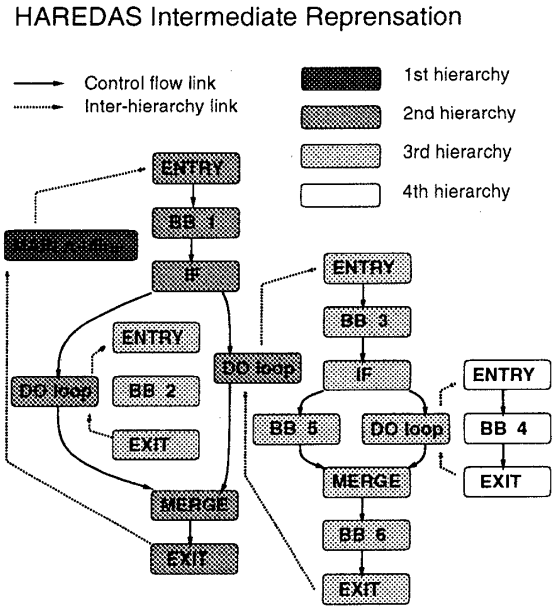

図 2: 中間表現

Parallelizing Compiler for Distributed Environment
Kazuyuki SHUDOH, Kenichi SUGAHARA, Seishiro HAMANAKA, Yoichi MURAOKA
School of Science and Engineering, WASEDA University

2.2 構成要素

資源管理サーバ コンパイル時には、プログラムの分割、分割されたタスクのスケジューリング、そして、自分の子となっているプロセッサのための実行形式の生成を行なう。

プログラムの実行時には、まずタスクの起動、タスクから要求される通信先アドレスの解決を行ない、実行中はタスクごとの実行条件を管理し、タスクの制御を行なう。

プリプロセッサ 入力言語で記述されたプログラムを中間表現に変換し、さらに中間表現のテキスト表現に変換してファイルに格納する。ユーザはこのテキスト表現をサーバに渡す。

コンソール ユーザとサーバのインタフェース。仮想計算機のコンソール、シェル。ユーザはこれでサーバに接続し、仮想計算機の情報を得たり、コンパイル、プログラムの実行などを指示する。

3 コンパイル

親サーバから渡されたプログラムのタスクへの分割、スケジューリングが、ツリーの根から末端にかけて再帰的に行なわれる (現在の実装では、タスクの階層的な分割は行なっていない)。その後、各サーバは中間表現からC言語のコードを生成し、そのコンパイルを行なう。

このコンパイルは外部のCコンパイラに頼る。命令レベルの最適化で既存のコンパイラより優れたものを作るには労力がかかり過ぎること、が理由である。

3.1 プログラムのタスクへの分割

現在は、最外側ループ、連続した代入文、制御の分岐から合流まで、をひとつのタスクとして、プログラムを分割している。現時点では静的スケジューリングを採用しているため実行時でないと分岐先決定できない条件文は一つのタスクにまとめている。

タスクの統合を行なって粒度を調整する、といったことは行なっていない。

3.2 スケジューリング

スケジューリングの前処理として、各タスクに対して実行開始条件の算出、処理コストの見積り、タスクのレベル算出を行なう。その結果に基づき、CP(クリティカルパス) 法を用いてタスクを子に割り当てる。

このスケジューリングはコンパイル時に行なっている。仮想計算機を構成する計算機が多様ゆえに、実行時スケジューリングは時間もしくは空間的なオーバーヘッドが大きすぎるためである。

静的スケジューリングでは効率に限界があるので、機種非依存でかつ充分に低レベルな中間言語を、動的スケジューリング後に実行形式に変換することも考えている。が、変換、通信先解決のオーバーヘッドと、効率の上昇のどちらが大きいかは未評価。

4 プログラムの実行

実行制御 依存解析の結果から算出される最早実行条件に基づいたタスクの実行制御が、サーバツリーにより階層的に行なわれる。

タスクは基本的には、データ依存解析の結果に基づきデータフロー方式で駆動される。必要なデータが揃いしだい実行が開始され、実行決定の通知を親サーバから届いていれば、他タスクにデータを送信し、親に実行終了を通知する。

通信先アドレスの解決 階層的にスケジューリングを行なっているため、各タスクは通信相手の場所をコンパイル時には知らない。実行時に解決される。プログラムの実行開始時に一斉起動されるタスクは、親サーバに問い合わせる。サーバは、自身で解決できない場合、他のサーバに問い合わせる。

5 実用化へ向けて

現在動いているシステムは、構想のサブセットである。実用に耐えるものとするために以下を考えている。

FORTRAN77 の完全サポート

一部、サポートされていない FORTRAN77 の構文のサポート。

並列計算機のサポート
中粒度並列処理のサポート

ツリーの末端、低遅延ネットワークを持つ部分での中粒度の並列処理のサポート。並列計算機、高性能クラスタなどのシステムへの組み込みだけでなく、データの分散方法なども検討が必要。

再帰的なプログラム分割、スケジューリング

その効果を検討する。