LAN環境におけるプログラムの並列化についての研究*

1L－9

○宮嶋 義博　松山 実　横井 利彰
武蔵工業大学

1 はじめに

大学や研究所では複数の計算機をLANで接続したシステムが広く利用されている．このシステムを活用して並列処理を行うことが可能になれば，プログラムの実行速度の向上が期待できる．ただ，このようなシステムを人間の手で作成することは容易でない．また，異なる機種が接続されている場合，ネットワーク透過性を実現する上で，アーキテクチャによって異なる部分を吸収する手間は大きい．

ここでは，このような環境で並列処理を容易にするために，C言語で書かれた逐次的なプログラムを，PVM(Parallel Virtual Machine)[1]上で並列実行可能なプログラムに変換するコンパイラを提案する．

2 PVM

PVMは，ネットワークに接続された異機種UNIXコンピュータ群を，単一の並列コンピュータとして利用することを可能にするソフトウェアシステムである．これにより，多数のコンピュータの持つ計算パワーを，一つの大規模計算機問題に集結して処理を行うことができる．

PVMは，アプリケーション，マシン及びネットワークレベルでの異機種間利用をサポートする．PVMは，異なるコンピュータ間の整数あるいは浮動小数点数の表現の違いを吸収するためのデータ変換を扱うことができる．そして，PVMは多様なネットワークで接続されたバーチャルマシンを実現する．

*Parallelizing Compiler in LAN Environment
Yoshihiro Miyajima, Minoru Matsuyama and Toshiaki Yokoi, Musashi Institute of Technology

3 並列化コンパイラの構成

本報告における並列化コンパイラは，入力としてC言語プログラムを受け取り，それをプログラム解析部，負荷分散処理部，並列化処理部の三つの処理部で処理し，PVM上で並列実行可能なプログラム群を出力する．

プログラム解析部は，通常のコンパイラの字句解析部，構文解析部に相当する部分で，ここで入力されたプログラムを解析し，並列化のための情報を得る．負荷分散処理部は，意味解析部に相当し，プログラムの並列化を行う．並列化処理部は，コード生成部に相当し，PVMプログラムを出力する．

3.1 プログラム解析部

ここでは，与えられたプログラムを並列処理単位に分割し，それぞれの並列処理単位が他の並列処理単位に与える影響を解析する．

プログラムの並列処理単位への分割は，計算機システムの構成に依存する．例えば，多くのマルチプロセッサからなる計算機システムの場合，各プロセッサ間の通信手段として，共有メモリや通信バスなど，特別の通信機構を備えており，通信にそれほど時間がかからない．そのため，ブロック単位や，ループ単位，あるいはより細かな文単位での並列処理による処理時間短縮化が可能である．しかし，

LAN環境などの分散処理システムにおいては，通信オーバヘッドが増大し，このような単位をプロセスとした並列処理では，速度向上は望めない．つまり，並列処理単位は，計算機のシステムに合致したものを採用しなければならない．そこで，並列処理単位を関数単位で分割する．

次に，各並列処理単位を実行することによって，他の並列処理単位にどのような影響を与えるかを解析する．並列処理単位を，入力されたデータを処理し結果を出力するものと考え，それぞれの並列処理単位に対して，入力変数，出力変数を定義することで，他の並列処理単位への影響の有無を表す．ここで，入力変数とは，並列処理単位が実行時に値を参照する変数を指し，出力変数とは，並列処理単位の実行によって，値が変更される変数を指す．

3.2 負荷分散部

ここでは，生成するプロセスの粒度と数を決定する．プロセスの粒度・数によって，実行時の効率が大きく変化するので，この決定は重要である．プロセスの粒度を小さく，数を多くすれば，並列実行による実行効率の向上を期待できるが，その反面，各プロセス間での通信オーバヘッドが大きくなる．プロセスの粒度を大きく，数を少なくすれば，その逆となる．

ここでは，並列処理単位の粒度と数について検討する．PVMでは，メッセージ交換方式による疎結合型のマルチプロセッサシステムモデルを採用している．そのため，ある程度の粒度がないと通信オーバヘッドの増大により，全体の処理向上は望めない．

そこで，関数を最小単位として，その実行時間により関数を融合させ並列処理単位を決定する方法[2]がある．たとえば，ある関数が他の関数から頻繁に呼び出されている場合は，その２つをまとめて１つの並列処理単位とすれば，オーバヘッドが削減できる．よって，関数の実行時間と各関数の呼び出し回数の関係より，最適なプロセス数および粒度を決定する．ただ，実際に関数の実行時間をあらかじめ知ることは不可能である．そこで，各関数の実行時間はその関数が扱うデータ数とループ回数から推定する．

3.3 並列化生成部

ここでは，負荷分散部で得たプロセスへの融合の情報をもとに，実際にソースプログラムを変換し，PVM上で並列実行可能なプログラム群を出力する．

UNIX上でネットワーク透過性を実現するのに鍵となる技術は，ソケットライブラリによるネットワーク通信と，selectシステムコールによる入出力の多重化である[3]．しかし，これらを利用したプログラムの作成は容易ではなく，とりわけ，アーキテクチャの異なる部分を吸収する手間は大きかった．

そこで，ここではPVMライブラリを用いることでこれらの手間を吸収し，異なるアーキテクチャのホストへも簡単に並列システムを拡張できる．

4 おわりに

今後，PVMの特徴を踏まえた上で，並列化コンパイラを実装し，実アプリケーションを用いてこのコンパイラの評価をする予定である．

参考文献

[1] AL Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, Vaidy SunDeram: "PVM 3 USER'S GUIDE AND REFERENCE MANUAL", 1993.

[2] 水上 正，石川 知雄: "LAN環境における関数を基準とした並列処理単位の検討"，電子情報通信学会総合大会講演論文集 D-79，1995年3月．

[3] 東田 学: "なぜPVM over ATMか"，UNIX MAGAZINE 1995.2，pp.24-32.