PVM利用のための並列化コンパイラ*

3E－3

○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コンピュータを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．

このようなネットワークに接続されたコンピュータを，仮想的に分散メモリ型の並列計算機として扱うことを可能にするシステムに，PVM(Parallel Virtural Machine)[1]がある．しかしPVMは，仮想的な並列コンピュータを提供するだけで，そこで動作するプログラムは自分で作成しなければならない．

ここでは，PVM上で実行可能なプログラムの作成支援として，C言語で書かれた逐次的なプログラムをPVM上で並列実行可能なプログラムに変換するコンパイラの構造とその評価について述べる．

2 並列化コンパイラの構成

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，負荷分散部，並列化生成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．

2.1 プログラム解析部

ここでは，与えられたプログラムを基本並列処理単位に分割し，それぞれの関数が他の関数に与える影響を解析する．

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

2.2 負荷分散部

負荷分散部では，生成するプロセスの粒度とプロセス数を決定する．

ここでは，LAN環境においての実現を考えており，ある程度の粒度がないとネットワーク間のオーバヘッド増大により，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ることにより，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により，最適なプロセス数および粒度を決定する．

2.2.1 負荷分散部のアルゴリズム

まず，関数の実行時間を求めるが，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．

つぎに，複数のマシンでの分散処理との実行時間単位の閾値を求める．関数の実行時間が短ければ，それだけ処理時間における通信負荷の増大により逐次処理の方が処理時間が短くなってしまう．関数の実行時間が適度な長さであれば，分散処理させる方が処理時間を短縮させることができる．その境目を決定する．

以上を求めた上で，負荷分散部のアルゴリズムをここで提案する．

1. 関数を親子関係に従ってツリー上に配置
2. 引数や返値を持たない関数は親関数と融合
3. 各並列処理単位の実行時間単位を仮の実行時間単位として決定

4. ツリー上に配置された関数の葉の方からたどり，実行時間単位を見て，閾値より小さければ，その処理単位の親を見る

   (a) もし，親関数の多重ループ内で呼び出されている場合は，その多重ループの最も内側のループと関数呼び出し部を，子関数側に移す変換を両関数で行い，新しい実行時間を求め4. へ移る

   (b) そうでなければ，親関数と融合して新しい並列処理単位とし，実行時間単位を求める

5. 全ての並列処理単位の実行時間単位が，閾値より大きくなるか，並列処理単位が1つに融合されるまで4. に戻る

2.3 並列化生成部

並列化生成部では，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書き換え，PVM上で並列実行可能なプログラム群を出力する．ここでは，出力するPVMのプログラムの形式はマスタ/スレーブモデルとする．

2.4 マスタ/スレーブモデル

マスタ/スレーブモデルは，マスタープログラムが複数のスレーブプログラムを生成・制御し，各スレーブプログラムが各並列処理単位を実行するモデルである．

マスタはpvm_mytid() を呼び出し，PVMシステムの利用とプロセス間通信を実行可能にする．そしてpvm_spawn() を呼び出し，PVMを構成する各マシンでスレーブプログラムを実行する．各スレーブプログラムはpvm_mytid() を呼び出し，プロセス間通信を実行可能にする．続いて，pvm_send() あるいはpvm_recv() を呼び出し，プロセス間通信でメッセージを交換する．スレーブプログラムのpvm_send() とpvm_recv() の間に置かれた並列処理単位の処理を行う．

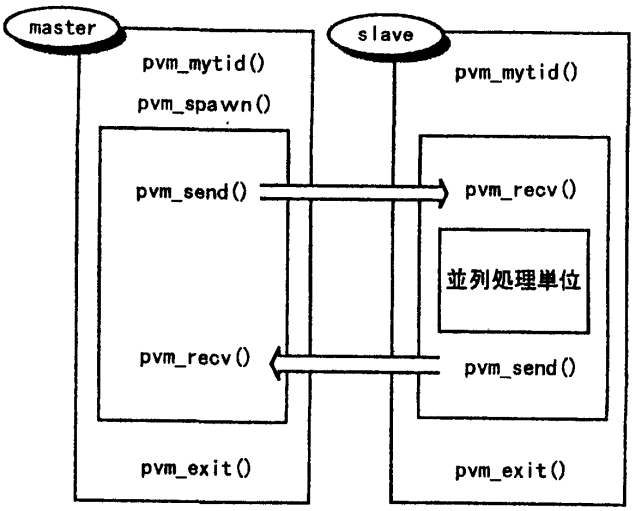

図 1: マスタ/スレーブモデル

3 おわりに

今回PVM上で動作するプログラムの作成支援としてコンパイラの構造についてのべた．主にループ処理から並列性を導いているため，すべてのプログラムで並列性を導けるとは限らない．今後，そのようなプログラムに対してどのように並列性を導出するかについて検討を進める．

参考文献

[1] Ai Geist, Adam Beguelin, Jack Dongarra, Robert Manchek, Vaidy Sunderam: "PVM3 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,(1995.3)