4M−8

並列計算機ADENA用自動並列化コンパイラ:APARC
−概　要−

黒田 茂[1] 材木幸治[2] 若谷彰良[2] 岡本 理[2] 廉田 浩[2]
株式会社 松下ソフトリサーチ[1] 松下電器産業株式会社[2]

1.はじめに

ADENA(Alternating Direction Edition Nexus Array:交互方向編集アレイ)は、科学技術分野における数値シミュレーションを高速に処理することを目的として開発されたMIMD型並列計算機で、そのハードウェアは1台のホスト計算機と2次元的に配置された多数のPE(プロセッサエレメント)からなるADENAキューブ及びADENA用ネットワーク(Hyper Cross Net)から構成される[1]。

並列計算機システムのプログラミング手法には、大きく2つの方法がある。一つは並列記述言語を用い陽に並列動作を記述するものである。もう一つは逐次型言語で記述されたプログラムを自動並列化コンパイラにより並列プログラムに変換するアプローチである[2][3]。現在我々は自動並列化コンパイラAPARC (Adetran PARallelizing Compiler)を開発中であり、これはFORTRANプログラムを、FORTRANに並列動作記述の構文を加えたADETRAN[4]プログラムに変換するものである。APARCは、プログラム中のdoループを検出し、制御構造を解析する。そしてデータ参照関係解析によって並列化を行っている。特に、doループ中に分岐命令やスカラー変数があっても、並列化の対象としている。

本稿では、制御構造解析を中心にその概要について報告する。

2.ADETRAN

ADETRANによる並列処理記述の例を図1に示す。pdoとpendで囲まれた部分を各PEが、インデックス変数j, kに関して並列実行することを示している。

ADENAはPEが2次元アレイ状に並んでおり、各PEに番号(i, j)が付けられている。これを仮にPE(i, j)と表すと、図1に示されたプログラムでは各PE(j,k)が同時にdoループを実行することになる。

```
      pdo j = 1, jmax, k = 1, kmax
       do 10 i = imax - 1, 1, -1
        u(i, /j, k/) = p(i, /j, k/)*u(i + 1, /j, k/) + q(i, /j, k/)
 10    continue
      pend
```

図1. ADETRANプログラム例

3.APARC

FORTRANプログラムはまずpreprocessor部でホスト実行部とADENA実行部に分けられる。次にlexical analyzer/ parser部で並列化のための中間コードを生成し、parallelism ditector/regulator部で並列化を行う。

最後にcode generator部でADETRANプログラムを出力する。その流れを図2に示す。

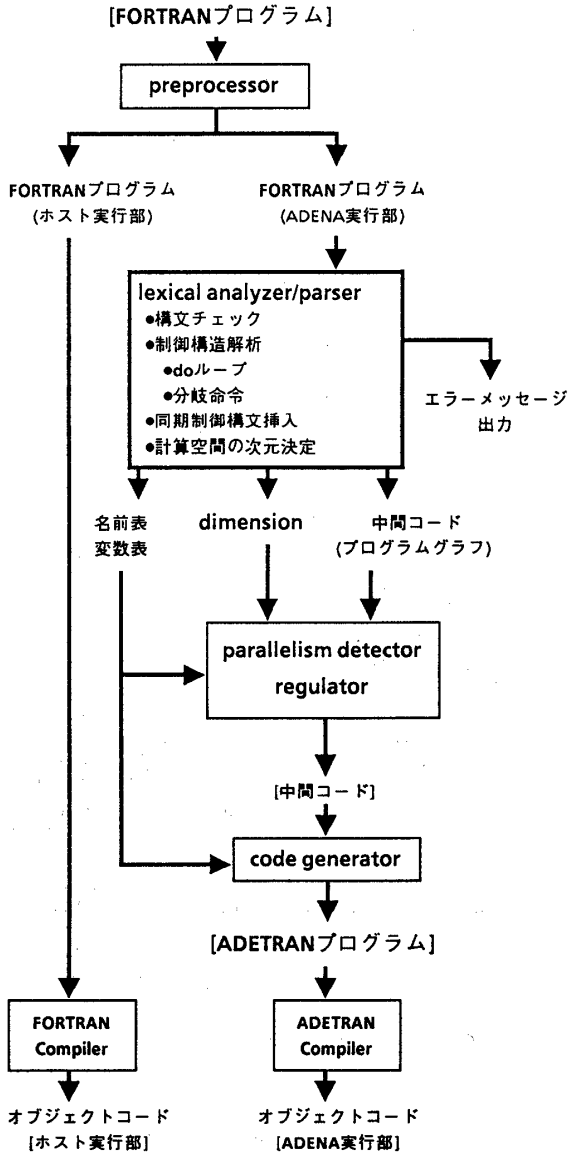


図2.APARCの構成

APARC: ADETRAN Parallelizing Compiler

Shigeru KURODA[1], Koji ZAIKI[2], Akiyoshi WAKATANI[2], Tadashi OKAMOTO[2], Hiroshi KADOTA[2]
Matsushita Soft Research.,Inc.[1] Matsushita Electric Industrial Co.,Ltd.[2]

3-1.lexical analyzer/parser

lexical analyzer及びparserでは、それぞれ字句解析から構文解析を行う。並列性抽出のために、parserでは中間コードを生成する。この中間コードには、制御構造解析、同期制御構文挿入、及び計算空間の次元決定に関する情報が登録されている。

a)制御構造解析

ここでは多重doループ構造が並列化可能かどうか以下の2点についてチェックする。

(1)goto文の分岐先

doループ内にgoto文がある場合、そのgoto文がどのdoループの外側に飛び越したかによって、並列化できるdoループを決定する。これはgoto文の飛び越し先が、goto文を含むdoループの外側では、そのdoループは初期値から終値までを各PEに割当て、独立に実行することができない。

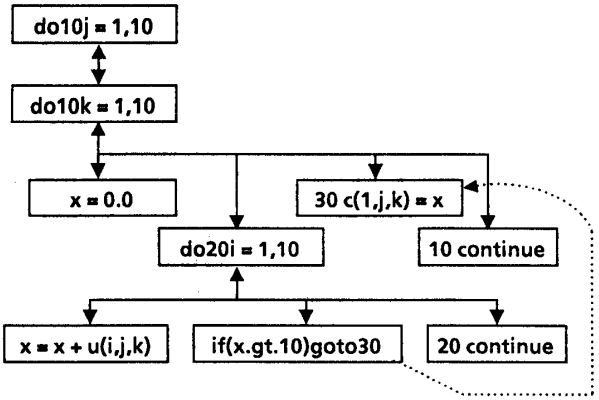
(2)多重ループ間に実行文を含まない

また多重ループ間に実行文が含まれる場合、そのループに関して並列実行できない。

以上の様子を並列化可能な例で、中間コードを交えて図3に示す。

```
      do 10 j = 1,10
      do 10 k = 1,10
      x = 0.0
      do 20 i = 1,10
         x = x + u(i,j,k)
         if(x.gt.10)goto 30
20    continue
30    c(1,j,k) = x
10    continue
```

並列化可能な場合のプログラム



doループの様子を表した中間コード

図3.doループの並列化判定

この例では、doループのインデックス変数j,kのdoループ内において並列化可能であるといえる。

以下にdoループの構造解析の手順を示す。

1)doループを親とし、そのループ内に存在する制御文や実行文などを、その子とする。

2)その関係を上下の矢印で示したように上下両方にポインタでリンクする。

3)doループ内にgoto文が存在するとき、その飛び越し先の実行文へのリンクを作成する。

このように構造を解析し中間コードを生成する。

4)3)で飛び越し先の実行文から、その親であるdo文までリンクをたどる。

5)その親であるdo文を含めて外側のdoループで並列化可能と判断する。

b)同期制御構文挿入

ADETRAN言語では、全PEの同期を行うために、同期制御構文を用意している。

c)計算空間の次元決定

ADENAシステムでは、ユーザーが計算空間を必ず2次元問題と3次元問題のどちらかに統一してプログラム開発を行う必要がある。つまりAPARCはADENA実行部のプログラムを解析するとき、計算空間を2次元の場合か3次元の場合かを判断し、どちらかに決定しなければならない。ここではADENA実行部に出てくる配列の次元を調べ、その次元を中間コードに登録する。2次元配列と3次元配列が混在されている場合はエラーメッセージを出力する。

3-2.parallelism detector/regulator

parserで求めた中間コードから、並列性抽出を行う。doループを検出し、その構造が並列化可能かをチェックする。もし不可能ならば、エラー処理を行う。可能ならば、並列処理用ブロックを中間コードに持たせ、プロセッサ間通信の構文情報を中間コードに登録する。

図3において、doループの構造に着目すると並列化可能であると判断できる。また、図3においてデータ参照関係に着目すると、インデックス変数j,kのdoループについては、並列化可能である。次に内側のdoループについては、スカラー変数xが回帰的に出現している。しかしこのスカラー変数xはこの実行文の前では左辺にある。つまりインデックス変数j,kについてはデータ参照関係を生じない。従って、インデックス変数j,kのdoループは初期値から終値までを各PEに割当て独立に実行できる。

3-3.code generator

parser,parallelism datector/regulatorで生成された中間コードをもとにADETRANプログラムを出力する。このとき、並列化できなかった場合、その旨を出力プログラムにメッセージを追加して出力する。

4.おわりに

以上のように、ADENA用自動並列化コンパイラAPARCについて制御構造解析を中心に述べた。

現在APARCは、幾らかのアプリケーションを変換できている。今後数多くのアプリケーションを変換して評価を行っていく。

5.参考文献

[1] H.Kadota, et al,"VLSI Parallel Computer with Data Transfer Network:ADENA",Proc.ICPP,pp319-332(1989).
[2] 村岡,"超並列処理コンパイラ",コロナ社,(1990).
[3] David A. Pauda,"Advanced compiler optimizations for supercomputers",Comm.ACM,1986.
[4] T.Nogi,"Parallel Programming Laguage ADETRAN", Memoirs of the Fac. Eng., Kyoto Univ., Vol. L1,part 4, Oct.,1989.