1E－5

手続き間解析を用いた共有メモリ型並列計算機のための自動並列化変換方式

田村智洋　新名博　藤原優史　岩澤京子

東京農工大学工学部電子情報工学科

1．はじめに

並列計算機で動くプログラムを作ることは，各々のプロセッサの同期処理や，データの管理が複雑になるため，逐次型のプログラムを作ることより困難である．筆者の所属する大学の情報処理センターにある共有メモリ型並列計算機では，運搬依存がなく，手続き呼び出しがない for ループのみを自動並列化の対象としている．本研究では，手続き間にまたがるデータフロー解析を行った結果を用い，for ループを並列化対象にして手続き呼び出しがあるループに対しても，並列化変換を行う．

2．システムの機能

本システムは，機能限定したＣ言語で書かれた逐次型プログラムを入力とし，計算機が用意する並列実行命令を埋め込み，並列に実行可能なプログラムを出力する．

入力プログラムには，次のような制限がある．

・引数の無い手続き呼出し

・配列は，一次元配列のみ

・構造体は，扱わない

・ポインタは扱わない

3．並列化変換手順

並列化を行うプログラムの変換手順を述べる．

3．1並列化候補となるループの選択

並列化の対象になる for ループをピックアップする．入れ子になっているループについては，手続き呼び出しを含んでいても外側のループを並列化の対象にする．

3．2　手続きを分類する

逐次に実行されるものと並列に実行されるものとでは変換の方法が違うので，コールグラフや手続き呼出し位置の情報を用い，手続きを３つのグループに分類する．

a．逐次に呼び出されるもの

b．並列に呼び出されるもの

c．逐次にも，並列にも呼び出されるもの

逐次に呼び出される手続きでは，手続き内の for ループについて並列化を行う．対象ループ中の手続き呼び出しは，名称を変更する．

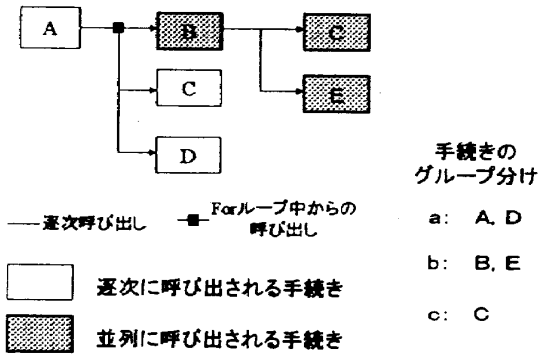

図1　手続きのグループ分けの図

並列に呼び出される手続きは，手続き名を“手続き名＋para”と変え，手続き内の for ループの並列化は行わない．この手続きの中から呼び出す他の手続きも，名称を変えるなど並列実行用に変更する．

ｃのグループは，逐次型の手続きと並列型の手続きが必要なので手続きを複写する．新しく複写された手続きは，並列実行用に変更する．

Automatic Parallelizing Method for Shared Memory Parallel Processors using Interprocedural Analysis

Chihiro Tamura, Hiroshi Niina, Masashi Fujiwara, Kyoko Iwasawa

Tech.,Tokyo Univ. of Agri. and Tech.

3．3並列化変換

ループ毎に，グローバル変数とローカル変数についてループ運搬依存を調べる．

運搬依存がある変数には，使用の直前にウェイト命令を，定義の直後にポスト命令を埋め込む．ただし，定義が，出口を支配していないときは，各繰り返しの最後に埋め込む．また，総和のようなリダクション計算には，特別に通信なしの変換を施す．

運搬依存がない変数は，プライベート変数になり各プロセッサ毎に独立した値を持たせる．ループの後で使用される変数については，定義が出口を支配していない場合にも，繰り返し回数を保持することにより終値を保証する．

4．実験結果

共有メモリ型並列計算機(HP 社 Exemplar S クラス 1 Node 4 CPU)上で，(図２)の逐次プログラムと，このプログラムから自動生成したプログラム(図 3)を，走らせて CPU 時間と経過時間を計測した．

例に挙げた方法で同期処理，終値保証を行った場合と，全く独立な同期処理が必要ない場合の二種のプログラムから，表１の様な結果が得られた．

```
#define MAX 10000
int a;

hantei(){
    int tmep;
    temp = a;
    処理が入る
    a = temp;
}
main(){
    int b,c,i;
    b = 1;
    for(i=0;i<MAX;i++){
        a = i;
        hantei();
        c = a/b;
        if(c>50)
            b = c + a;
    }
    a = c;
}
```

図２ 変換前のリスト例

表１実行結果　同期処理＋終値保証

	CPU 時間(s)	経過時間(s)
逐次実行	０．０６	０．８３
並列実行	０．１０	１．８５

終値保証のみ

	CPU 時間(s)	経過時間(s)
逐次実行	０．０７	１．００
並列実行	０．０８	０．７９

5．終わりに

本システムによって手続き間で運搬依存が無い場合には並列化の恩恵を受けられるようになった．

しかし同期処理を埋め込む場合には，通信の時期や回数，演算とのつり合いを考慮して効率の良い並列化変換の方法を研究していく必要がある．

```
#define MAX 10000
static thread_private int a;
(グローバル変数は，プロセッサごとに独立で持たせる型に変更する)
hanteipara(){
(並列に呼び出される手続き名を変更する)
    int tmep;
・a の値が確定していない場合は，ウェイト命令を入れる
    temp = a;
    処理が入る
    a = temp;
}
main(){
    int b,c,i;
    int *chai,*cita;
(各プロセッサごとにイタレーションと変数データを記録するためのメモリを確保するポインタ)
    b = 1;
・各プロセッサごとに，イタレーションと変数データを記録するメモリを確保する
・ループを並列に実行する命令を入れる(誘導変数 i)
・プライベート変数を決める命令を入れる(c)
    for(i=0;i<MAX;i++){
        a = i;
        hanteipara();
(並列に呼び出されるので手続き呼出しを変更する)
 ・b の値が決まるまでウェイトする命令を入れる
        c = a/b;
        if(c>50)
            b = c + a;
 ・b の値を次のイタレーションにポストする命令を入れる
        cita[myid_] = i;(イタレーションを記録する)
        chai[myid_] = c;(変数データを記録する)
    }
・記録したイタレーション情報を参照して最後に定義した変数データの値をｃに代入する
・記録用に確保したメモリを解放する
    a = c;
}
```

図３　変換をしたリスト例