SIMD 型超並列計算機 SM-1 における仮想プロセッサ機能の実現

2D－7

貴島 寿郎　湯淺 太一
豊橋技術科学大学

1 はじめに

並 C [1] は，SIMD 型超並列計算機 SM-1 [2] 上でのプログラミング言語および他のプログラミング言語のターゲット言語として開発された SIMD 向き超並列 C 言語である．並 C における基本的な実行環境は，多数のプロセッサ（PE）からなる PE アレイと，フロントエンド（FE）より構成される[1]（図1参照）．並 C は言語

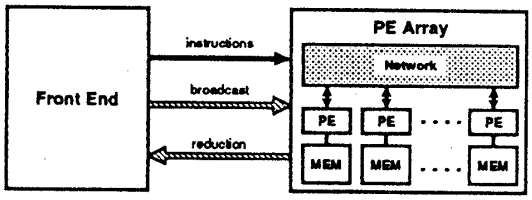

図 1: 並 C の実行環境

仕様上，実行環境に強く依存するようなプログラムを要求するため，実 PE 数を超えるような要素数の並列データ処理の場合，データの配置や並列演算のエミュレーションなどはプログラマが行う必要がある．しかしデータ並列プログラミングにおいては，並列処理に必要十分な個数の PE （仮想 PE ）の存在を仮定することによりアルゴリズムの記述が簡潔となることが多い．

そこで今回，仮想 PE を想定している並 C プログラムを実 PE 上で実行可能な形式に変換するためのトランスレータを実現した．本稿ではこのトランスレータの概要について述べると共に，変換されたプログラムの例をもとに，変換適用時の実行効率に対する影響について考察する．

2 並 C について

並 C のプログラムは，基本的には従来の C 言語で書かれたプログラム中に並列データを処理する部分が埋めこまれた形式となっている．並列データの演算に関する部分は PE アレイ側で実行され，その他の部分は FE 側で実行される．

図 2に並 C のプログラム例を示す[2]．このプログラムは関数 $f(x) = \frac{4}{1+x^2}$ を $[0,1]$ の範囲で数値積分することにより円周率の近似値を求めるものである．基本的には区間 $[0,1]$ を各 PE で等分して担当させ，それぞれの小区間の積分近似値を計算し，最後にそれらの総和をとることにより解を求めている．

```
#pragma PASIZE *= MAG
double calc_pi()
{
    double      width = 1.0 / PASIZE;
    double      sum;
    double $    x = (PINDEX + 0.5) * width, f;

    f = 4.0 / (1.0 + x * x);
    sum = width * (+$ f);
    return sum;
}
```

図 2: 円周率の計算（変換前）

並列データを保持するための変数は並列変数として宣言される．並列変数を宣言する場合はデータ型に $ を後置する．図 2において，x は double 型の並列変数として宣言され，各 PE のローカル・メモリ上に double 型の変数領域が確保される．

並列データをオペランドとする算術演算や並列変数への代入などは，各 PE 上でそれぞれのローカルなデータに対して行われる．並列データが要求される位置に FE 側のデータが現れている場合，その値は全 PE へ配布される．その他，並列変数の内容に何らかの演算を施して1つの値にまとめるためのリダクション演算子が用意されている．図 2では変数 f の値の総和を求める際に +$ というリダクション演算子を用いている．

1行目の #pragma ディレクティブは仮想 PE 数の指定を表している．この例では仮想 PE 数を実 PE 数の MAG 倍と指定している．

Implementation of virtual processors on a SIMD massively parallel computer SM-1
Toshiro Kijima　Taiichi Yuasa
Toyohashi University of Technology
Toyohashi 441, Japan

[1] SM-1 の現行モデルでは，PE アレイは 32 × 32 の2次元メッシュを形成しており，各 PE は 1M バイトのローカルメモリを持っている．

[2] PASIZE は実 PE 数を表す組み込み変数である．

3 変換の基本方針とシステムの構成

本トランスレータで用いている変換手法は文献 [3] で紹介されている MIMD 上でのエミュレーション手法を SIMD に応用したものであり，基本的には並列変数の並列配列化（folding），1つの PE 上に割り当てられたデータに対する並列演算のエミュレーション（emulation loop）などを行う．変換後のプログラムを図 3に示す．

図 4にトランスレータの構成を示す．構文解析部の開発には yacc （GNU bison）を用いた．また中間コード変換部と並 C コード生成部は Common Lisp で記述されており，KCL 上で実行される．

4 評価

図 2のプログラムにおいて，実 PE 数に対する仮想 PE 数の倍率（VP 比）の増加による実行時間の変化を SM-1 上で測定した．その結果を図 5に示す．縦軸は変換前のプログラムの実行時間に対する実行時間の比である．図中の点線は実行時間比 = VP 比の線を表す．

このプログラムの場合，実行時間比は VP 比に単純に比例するのではなく，VP 比がある程度高くなるまではあまり増加せず，その後 VP 比の約 0.27 倍に漸近する．これは図 2のプログラム 8行目の +$ 演算の部分を変換する際に，図 3の１２から１５行目のように変換後の +$ 演算が 1 回で済むような形に最適化していることにより，1つの仮想 PE 当りの +$ 演算に要するコストが VP 比に反比例して減少していくためであると推察される．SM-1 の場合 +$ は加算の $2.8\log_2 \mathrm{PASIZE}$ 倍程度の演算時間を要するため，リダクション演算を多用するようなプログラムにおいては，このような最適化が効果的であると思われる．

```
double calc_pi()
{
    double      width = 1.0 / _vp_PASIZE;
    double      sum;
    double $    x[MAG], f[MAG];
    double $    _tmp;
    int         _k;

    for (_k = 0; _k < MAG; _k++) {
        x[_k] = (_vp_PINDEX[_k] + 0.5) * width;
        f[_k] = 4.0 / (1.0 + x[_k] * x[_k]);
    }
    _tmp = 0;
    for (_k = 0; _k < MAG; _k++)
        _tmp += f[_k];
    sum = width * (+$ _tmp);
    return sum;
}
```

図 3: 円周率の計算（変換後）

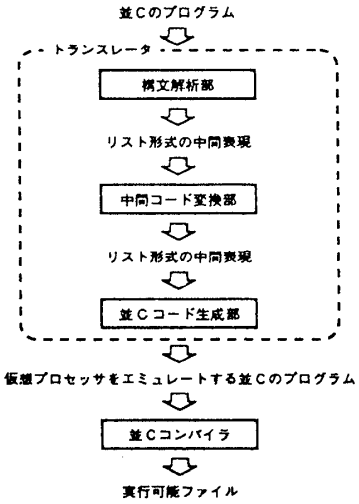

図 4: システムの構成

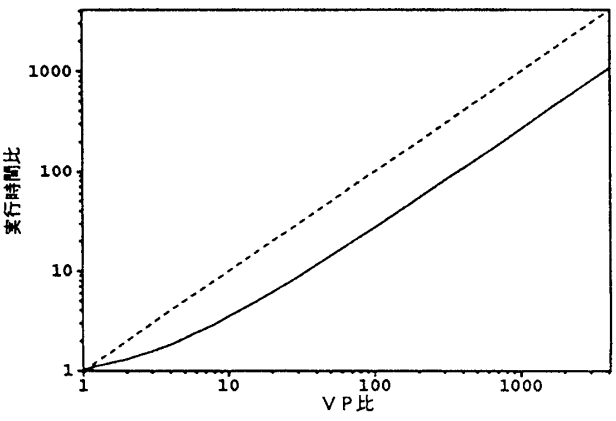

図 5: 実行結果

参考文献

[1] 貴島, 湯浅 : SIMD 型超並列プログラミング言語「並 C 」とそのコンパイラ, プログラミング — 言語・基礎・実践 — 研究会資料 9-9, 1992.

[2] 松田, 湯浅 : SIMD 型超並列計算機 SM-1 （仮称）の概要, 第 5 回 SWoPP 予稿集, 1992.

[3] Philip J.Hatcher, Michael J.Quinn : Data-parallel Programming on MIMD Computers, MIT Press, 1991.