データ並列型言語NCXにおける通信コストと仮想プロセッサマッピング

3L－6

栗本貴文　山下義行　中田育男
筑波大学

1　はじめに

データ並列型言語NCX[1]で書かれたプログラムをコンパイルしてMIMD型並列計算機で実行する際に、実行時間をできるだけ短くするための重要な最適化の一つに仮想プロセッサマッピングの問題がある。NCXプログラムでは並列実行の単位として仮想プロセッサ(VP)群を宣言し、データを各VPに分散して並列実行するよう記述されるが、VPをどのように物理プロセッサ(PP)に割り当てるかによってPP間通信の通信発生回数、通信されるデータ量が左右される。そこでこのPP間通信の通信コストができるだけ少なくなるような仮想プロセッサマッピングをコンパイル時に求めることを目的とし、プログラム中に明示的に記述されているVP間通信の式、VPのactivityに関する情報などをプログラムから抽出し、プログラム全体の通信コストを算出して仮想プロセッサマッピングを決定する研究を行なってきた。本稿ではその概要について述べると共に、例題プログラムを用いて求めた通信コストと実際の並列計算機での実行時間との比較、検討も行なう。

2　仮想プロセッサマッピングの方法

本研究で求める仮想プロセッサマッピングは現段階ではNCXのmesh fieldに対してのみ行ない、NCXプログラムをコンパイルする際にVPfieldの各次元に対して(1)Block分割とcyclic分割のどちらが良いか(2)物理プロセッサ総数をどのようにfieldの各次元に分配するか、を求める。その方法はNCXプログラムより抽出した各種情報とユーザより与えられる物理プロセッサ総数をもとに可能性のあるすべてのマッピングの場合に対してプログラム全体の通信コストを求める。この通信コストとは通信発生回数$Cost_S$と通信データ量$Cost_D$の二つの値で表現し、通信準備時間T_sと一データ送信時間T_dを用いることによって一つの通信式における送信時間Tは

$$T = Cost\_S * T_s + Cost\_D * T_d$$

と表現できる。そこでプログラム全体の通信コストを求めた後にこの式を用いて各マッピングの場合の通信時間を比較して、最小となるマッピングを検討する。

2.1　プログラムの解析：情報の抽出

次のような簡単なNCXプログラムの例を用いて必要な情報の抽出について説明する。

```
1  #define size 120
2  field f(size,size) on mesh;
3  int a,b on f;
4  main(){
5    in f(i,j){
6    int k,l;
7    a = i; b = j;
8      for (k = 0;k < size; k++)
9        if (i < 110) a = b*a@(i+1,0);
10     for (l = 0; l < size/2; l++)
11       if (j > 10) b = a*b@(0,j+2);
12   }}
```

計算に必要な情報とは、各VP間通信の式に対して(1)通信相手の記述は(2)その通信式を評価するVP、つまりその通信式の時点でactiveなVPの範囲(3)その通信式のプログラム中での実行回数、という情報と、VPfieldに関する情報である。上記の例の場合VP間通信式は9,11行目にあり、通信式に対する各情報は次のようになる

○通信相手の記述(proc_decl)は各次元毎に抽出する。この場合9行目(i←i+1,j←0)、11行目(i←0,j←j+2),と抽出する。

○activityも各次元毎に抽出する。9行目の通信式の場合、4行目(in文によるactiveとなるfieldの選択)と9行目(if文によるactivityの変更)の記述より、(0<=i<=109,0<=j<=119)とfield indexを用いてactiveの範囲を抽出する。

○通信式の実行回数は、その通信式がloop中に存在するかどうかを現段階では考慮している。9行目の通信式はsize回、11行目の通信式はsize/2回のため、それぞれ実行回数として120回,60回と抽出される。

これらの情報をコンパイル時に抽出するので、コンパイル時に静的に決定できない値が通信式の通信相手の記述、if文によるactivity変更の条件式、loopの制御式に記述される場合はマッピングを決定できない。現段階では各式の記述が次のような場合にマッピングを決定できる。

○proc_declが、定数、整数、field index、静的に解析できるmono変数を用いた式で記述されている場合。

○activityの変更が、field indexに対して定数、整数、静的に解析できるmono変数を用いて行なわれる場合

○loopの制御式が整数、定数、静的に解析できるmono変数を用いて表現されているfor-loop文の場合

VPfieldに関する情報は、次元数、各次元の大きさ、fieldに属する変数名などを1～3行目の宣言部を元に抽出する。

2.2　通信コストの算出

通信コストの算出は次の手順で行なわれる。

(1)PP総数とfieldの次元数よりPPの各次元への配分の考えられるすべての組合せを求める。例えば4台の場合は4×1,2×2,1×4の3通り求まる。

Communication cost and Virtual processor's mapping of NCX program on Distributed-Memory Machine
Takafumi Kurimoto,Yoshiyuki YAMASHITA,Ikuo NAKATA
(Univ. of Tsukuba)

	BB	CB	BC	CC	BB	CB	BC	CC
12 × 1	163.9	164.0	163.9	164.0	696.5	739.6	558.1	692.0
6 × 2	206.3	182.5	134.1	110.3	638.8	724.2	445.8	588.8
4 × 3	205.6	157.9	169.8	122.1	623.5	625.0	611.9	644.5
3 × 4	230.0	158.4	206.1	134.5	664.2	757.2	497.1	582.8
2 × 6	303.6	184.5	291.7	172.6	729.8	740.1	667.6	679.1
1 × 12	292.1	292.1	292.2	292.2	927.3	969.0	754.3	784.3

表 1: PP 総数 12 台の時の通信コスト (左側) と並列実行の結果 (右側)[msec]。通信コストの方は計算した結果に対して pilot1 の T_s, T_d を元に算出した時間である。また、[BB] という表記は一次元目： Block, 二次元目： Block の分割を意味する

(2) ある PP の組合せに対してプログラム全体の通信コストを次のようにして求める

(2.1) ある一つの通信式に対して、proc_decl,activity を元に各次元の分割のすべての組合せの場合の通信コストを求める。例えば先の例題は二次元なので、(一次元目、二次元目) =(Block,Block),(Block,Cyclic),(Cyclic,Block),(Cyclic,Cyclic) の 4 通りの場合の通信コストをそれぞれ求める。求めた結果に通信実行回数を掛けてその通信のプログラム中での通信コストとする。

(2.2) (2.1) を通信式すべてに対して行ない、それぞれの通信式の通信コストで同じ分割の組合せのものを足し合わせてプログラム全体の通信コストとする。

このようにして、PP の組合せ×分割の組合せの数だけプログラム全体の通信コストが求められるので、これらを比較して最小の通信コストとなる PP の配分、各次元の分割を決定する。手順 (2.1) における通信コストの計算法は次の二種類を考えた。

(I) 全調査法：すべての active な VP に対して、proc_decl を元に通信相手の VP を求め送信者と受信者がどの PP に割り当てられるかを調べる方法である。調べた結果をもとに各 PP 間での通信コストを求め、それらの最大値を持ってその通信の通信コストとする。すべての VP に対して計算を行なうので、field が大きくなると計算時間が大きくなる。

(II) 通信 index 法: 文献 [2] の通信関数の最適化方法を応用して計算する方法で、データを実際に送受信する VP だけに着目して送信 index, 受信 index という番号を割り振り、それを元に各 PP の通信コストを計算する方法。送信 index 番号と、受信 index 番号が同じ VP 間同士で通信が起きるよう両 index を割り振る必要があるため、VP 間通信が一対一で発生し、index を割り振られる VP の番号が当間隔である場合、つまり送信 index, 受信 index を割り振る VP の番号が $C_1 * field\_index + C_2$ (C_1, C_2 は整数値) と表現できる場合にしかこの方法を用いて計算することができない。(ただし、Broadcast 通信,reduction 通信の一対 N 通信はこの計算法をさらに応用することによって通信コストを求めることができる) しかしこの方法は各 PP の所持する index 範囲を計算して通信コストを求めるため実行時間は PP 台数にしか左右されずに済む。

このように通信 index 法の方が通信コストの計算時間が短くて済むので、プログラムを解析した結果通信 index 法を用いて計算できない場合にかぎり全調査法で通信コストを求める。例のプログラムはすべて index を割り振れる条件を満たしているので通信 index 法で求めることができる。

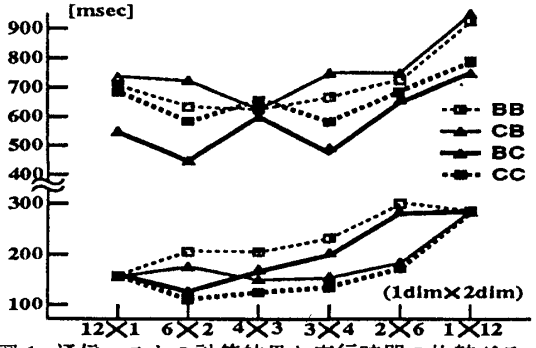

図 1: 通信コストの計算結果と実行時間の比較グラフ

3 実験、および考察

先に用いた二次元の field の例題を用いて、PP 総数 12 台の場合の通信コストの計算結果と実際に並列実行を行なってみての実行時間との比較を行なう。通信コストの計算はプログラム解析を手動で行ないそこから得た情報を元に計算させた結果で、表 1 の左側ににまとめた。並列実行は並列計算機 pilot-1 上で並列実行環境 EXPRESS[3] を用いて実行した結果で表 1 の右側にまとめてあり、両方をグラフにまとめたのが図 1 である。この結果から、大筋のマッピングの方向は決められる (物理プロセッサを 6 × 2、4 × 3、3 × 4 の範囲にして、CC または BC で分割する) が確実に現段階の計算だけでマッピングを決定できないことがわかる。これは通信コストの計算に通信の受信待ちが現段階では考慮されていないこと、また Cyclic 分割の場合の実行時の通信関数が Block 分割時に比べて多少計算量が多いことがその理由ではないかと考えている。

4 まとめ

本稿では NCX プログラムをコンパイルする時に、プログラムを解析してプログラム全体の通信コストを求めることで仮想プロセッサマッピングを求める方法について述べた。今後の展開としては、受信待ちなど並列計算機の traffic の考慮、各 VP の計算量を考慮してのマッピング、その他通信コストを計算しての再マッピングの自動指定の研究などがあげられる。

参考文献

[1] 超並列 C 言語 NCX 仕様書 version3: Septenber,1993

[2] S.K.S.Gupta,S.D.Kaushik,C.-H.Huang,and P. Sadayappan (Ohio State University): Compiling Array Expressions for Efficient Execution on Distributed-Memory Machines: Technical Report OSU-CISRC-4/94-19

[3] Nippon Steel Corp.,Nichimen Data System Corp.: 'ParalellWare User's Guide&Reference Manual': 1992