1L−7

FMMのホスト同期機能を用いたタスクの複数並列実行とその性能予測

定家　健治　岩根　雅彦　中嶋　保幸
(九州工業大学)　(東芝　北九州工場)

1．はじめに

FMM (Flexible Mesh-network Multi-microprocessors) では，ホスト制御での均一処理処理が可能であり，また，PUをいくつかのグループに分け，それぞれのグループを独立に動作させることができる．このとき，計算機上の全ての要素プロセッサを使用して処理を行うのが最適とは限らず問題の粒度や並列性，PU間のデータ転送などのオーバーヘッドを考えると最適なPUの数が決まる．本稿では，このグループ機能を用いて，単一タスク（最適グレインサイズ）の順次実行と，複数タスクグループ（非最適グレインサイズ）の並列実行とのスループットを偏微分方程式を例にとり比較，予測する．

2．ホスト同期SPMD型処理

PUは図1のようにホストのHSYNC割込み機能を使用するためにINTUに対して要求を出す．要求が受け付けられるとPUはINTバスを介して割込み識別と自分のPUNOを送る．INTUは，送られてきたPUNOをもとに要求レジスタのPUパターン部を更新する．そして，この更新された要求レジスタのPUパターン部に包含されるグループの情報の検索を行い，グループ内の全てのPUが同期割込みを行っていれば，INTUはCAMUからグループ情報を引き出すことができ，CAM読みだしレジスタに格納する．そして，引き出されたグループ情報のPUパターン部を要求レジスタのPUパターン部から削除し，INTUはHCに対して割込みをかけグループ内の処理の同期がとれたことをHCに知らせる．

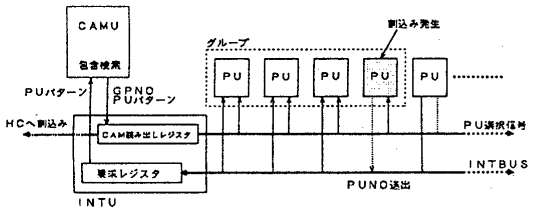

図1　ホスト同期機能

3．ラプラス方程式と並列解法

2次元平面上で次のラプラス方程式を反復法を用いて解く．正方領域0＜x＜1，0＜y＜1において

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad (1)$$

を満足する以下のディリクレ条件下における解とする．

$u(x, 0) = f_1(x)$, $u(x, 1) = f_2(x)$

$u(0, y) = f_3(y)$, $u(1, y) = f_4(y)$

式（1）をx軸，y軸ともhの間隔でN−1等分し，差分化するとN×Nの格子点を持つ．

各格子点を$u_{i,j}$(i, j=1...N)とすると次のようなN×N元の線形方程式になる．

$$u_{i,j} = (u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}) / 4 \qquad (2)$$

この線形方程式をチェビシェフSORによって解く場合にはk回目の繰り返し解$u_{i,j}^{(k)}$を次の漸化式よって計算する．

$$u_{i,j}^{(k+1)} = (u_{i-1,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)})\omega/4 + (1-\omega)u_{i,j}^{(k+1)} \qquad (3)$$

ここにωは緩和因子である．

図2，3にホストとPUでの処理のプログラムを示す．

```
1: GetGroup('pu.exc');
2: TransHostPu();
3: while (1)  {
4:     EventWait(HSync);
5:         if ( HSync == HSync(PUEND) )     TransPuHost();
6:         else      TransNews();
7:     StartGroup();
8: }
```

図2　ホストプログラム

図2は，ホストプログラムで，<1>行で，グループ内のPUにプログラムをロードする．
<2>は各PUにそれぞれの格子点のデータを転送する．
<4>はイベント待ちのｻｰﾋﾞｽｺｰﾙで割込みがかかるまで待ち続ける．
<5><6>で割込みの種別を判定して，同期割込みならば，外周点の交換のためのNEWS転送を行い，<7>でグループのスタートを開始する．そして，PUENDの割込みならば結果を回収して処理を終了する．

```
1: while (flag_pu == OFF)  {
2: kanwa();
3: falg_even = ON;
4:     for (j = 1; j <= m; j++)  {
5:         for (i = 1; i <= m; i++)  {
6:             if ( ( (i+j) % 2) == EVEN)  {
7:             Tnew=(T[i+1][j]+T[i-1][j]+T[i][j+1]+T[i][j-1])/4;
8:             dif = Tnew - T[i][j];
9:             T[i][j] = w * Tnew + (1-w) * T[i][j];
10:            if (abs(dif) > e )  flag_even = OFF; } } }
11:    WrtGSM Flag[ my_PUno ] = flag_pu
12:    for (i = 0 ; i = all_PUno ; i++ )  {
13:        RedGMS Flag[ i ] = Flag[ i ];
14:        flag_all = flag_all && Flag[ i ]  }
15:    if ( flag_all == ON )  {
16:        HSync( PUEND )
17:        puend; }
18:    else  {
19:            flag_pu = OFF
20:            move_to_buff();
21:            HSync( SYNC );
22:            move_from_buff();  }
23: }
```

図3　PUのプログラム

Parallel Execution of Tasks with Host Syncronize and its Performance Evaluation
Kenji SADAIE[1], Masahiko IWANE[1], Ysuyuki NAKASHIMA[2]
[1]Kyushu Institute of Technology, [2]Toshiba

図3は，PUのプログラムで，<1>はflag_puがOFFの間，処理をする．
<2>は緩和因子の計算の関数である．
<3>〜<9>で格子点の更新の計算を行う．
<10>は収束判定を行い，全ての格子点が収束していれば falg_ev en = ON となる．
<11>はGSMの自分の領域に収束フラグを書き込む．
<12><13>で他のPUの収束フラグを読み込み，<14>〜<16>で全ての収束フラグがONならPUENDのHsync割込みを出し，<17>で処理を終了する．
全てのPUが収束していなければ，<20>で外周点のデータをNEWS転送用のバッファに移す．
<21>はホスト同期のためのHsyncの割込みである．
<22>はホストがNEWS転送を行った後に，NEWS転送用バッファから外周点のデータを格子点のデータ領域に移すための関数である．

4．性能評価

格子点の数は普通，10^3〜10^8のオーダーであり，PUの数には制限があるので問題を分割して1つのPUに複数の格子点を持たせる必要がある．その際，PU間のデータ転送や同期のための割込み処理などのオーバーヘッドを考慮すると処理時間を最小にする最適のグレインサイズが存在する．ここで，格子点の数をN×N，PUの数をP×Pの2次元の正方格子とし，タスクの数をqとすると，1つのタスクを全てのPUを用いて順次処理する場合，2次元のN^2の格子点をP^2台のPUに割り当てるので1つのPUが処理する格子点の数：m_1は$m_1=\lceil N/P \rceil$であり，この処理をq回行う．また，qのタスクを並列に実行する場合には，N×Nの1つのタスクを一辺s（$=\lfloor \sqrt{(P^2/q)} \rfloor$）台の正方格子のPUに割り当て並列に処理するので，1つのPUが処理する格子点の一辺の数：m_2は$m_2=\lceil N/s \rceil$となる．

ラプラス方程式をFMMで並列処理するには，次のような処理時間がかかる．

ホストからPUへのデータ転送時間　：T_{HP}

$T_{HP}(m)=T_{setup}+T_{trans}\times m^2$

PUでの処理時間　：T_{PU}

$T_{PU}(m)=T_{PUP}\times m^2$

Hsyncの処理時間　：T_{Hsync}

$T_{Hsync}(P)=T_{HOST}+T_{INT}\times P^2$

NESW転送時間　：T_{NESW}

$T_{NEWS}(m)=(T_{setup}+T_{trans}\times m)\times 4$

収束判定のためのGMSへのアクセス時間　：T_{GMS}

$T_{GMS}(P)=T_{wrt}\times P^2+T_{read}\times P^2$

PUからホストへのデータ回収時間　：T_{PH}

$T_{PH}(m)=T_{setup}+T_{trans}\times m^2$

T_{setup}：DMAのｾｯﾄｱｯﾌﾟ時間
T_{trans}：1つの格子点の転送時間
T_{PUP}：一つの格子点の処理時間
T_{HOST}：ホストの処理時間
T_{INT}：INTUでの処理時間
T_{wrt}：GMSへの書込時間
T_{read}：GMSからの読み込み時間
m：1つのPUが処理する格子点の一辺の数
となる．

qのタスクを順次処理する場合，N^2の格子点をP^2台のPUを用いて処理し，その処理をq回繰り返すので，総実行時間　T_Sは

$$T_S=\{T_{HP}(m_1)\times P^2+(T_{PU}(m_1)+T_{NEWS}(m_1)+T_{Hsync}(P)+T_{GSM}(P))\times r+T_{PH}(m_1)\times P^2\}\times q \quad (4)$$

r：反復回数
となる．

また，並列処理に処理する場合，1つのタスクに割り当てられるPUの数はs^2となり，グループごとにNEWS転送，HSyncの割込みをかけるので，総実行時間　T_Pは

$$T_P=T_{HP}(m_2)\times s^2+(T_{PU}(m_2)+T_{NEWS}(m_2)\times q+T_{Hsync}(s)\times q+T_{GMS}(s))\times r+T_{PH}(m_2)\times s^2 \quad (5)$$

となる．
ここで，反復回数rが大きいとデータ転送時間T_{PH}とT_{HP}は無視することができる．

図2は，FMMにおいて，$T_{setup}=1000$，$T_{trans}=32$，$T_{PUP}=1500$，$T_{HOST}=1000$，$T_{wrt}=48$，$T_{read}=32$，格子点の数N×Nを64×64，タスクの数qを4，反復回数rを2000とし，PUの数を変化させた時の順次処理と並列処理の実行時間のグラフであり，横軸はPUの一辺の台数，縦軸はマシンサイクルを表す．

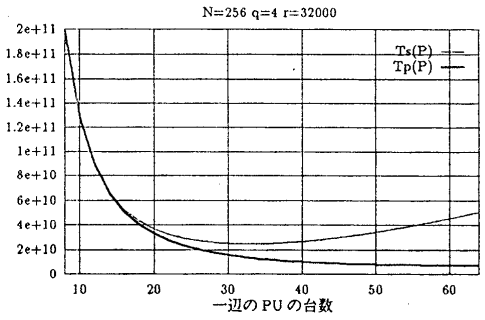

図4　順次処理と並列処理の比較

この結果，FMMでは順次処理の場合，256×256個の格子点を処理するとき最適のPUの数は32×32台であり，最適グレインサイズは8×8となる．そして，並列処理で同じPUの数の場合は最適グレインサイズではないが約200%処理速度が向上する．また，さらにPUの台数を増やして63×63台の最適グレインサイズで並列処理を行なうと順次処理の最適値と比べて約300%処理速度が向上する．これは，NEWS転送や，同期処理などのオーバーヘッドのために1つのPUが処理する格子点の数が2倍に増えても処理時間は2倍にはならないためである．

5．結び

偏微分方程式の解法やスムージングなどの均一の処理をホスト制御SPMD型処理で行う場合，単一のタスクを処理する場合，最適のグレインサイズで処理するのが最も効率がよいが，複数のタスクを処理する場合は，FMMのグループ処理機能を用いて複数のタスクを同時に並列処理すると同じPUの台数ではグレインサイズは非最適になるが全体として処理速度は向上する．さらに，PUの数を増やしてグレインサイズを最適にすると最も処理速度が速くなる．今後の課題としては，この予測結果をもとに実際のシステムでの実測値との比較，検討を行う必要がある．

参考文献

[1] H. J. Siegel, J. B. Armstrong, D. W. Watson, :Mapping Computer-Vision-Related Tasks onto Reconfigurable Parallel-Processing System, COMPUTER, 1992-2, pp. 54-63

[2] C. Siva Ram Muirthy, V. Rajaraman, :A MULTI MICRO-PROCESSOR ARCHITECTURE FOR SOLVING PARTIAL DIFFERENTIAL EQUATIONS, Microprocessing and Microprogramming, 20, (1987), pp113-118

[3] 塚原他：グループ共有メモリを持った柔軟な格子結合型並列計　算機，SWoPP日向灘' 92, 1992