ウエーブフロント型並列処理におけるタイルサイズの決定方式

5G－5

太田 寛　斎藤 靖彦　海永 正博　小野 裕幸
（株）日立製作所システム開発研究所

1．緒言

科学技術分野の計算の大半は大規模配列データを扱うループ計算で占められ、このようなループを超並列計算機で実行するには、データパラレル方式にもとづいたループ並列処理が一般的である。つまり、配列データやループの各イタレーションをプロセッサエレメント（以下PEと呼ぶ）に分散し、各PEが互いに通信しながら並列処理を行う。

ここで問題となるのは、PE間通信が処理全体のボトルネックになることである。特にデータ転送の起動オーバーヘッドが非常に大きいので、ループ運搬依存をもつ多重ループをウエーブフロント型並列処理する場合、イタレーション一回ごとにデータ転送を行うと性能がでない。この問題を解決するため、数回～数百回のイタレーションに一回の割合でデータ転送をまとめて行う研究がいくつか発表されている[1][2][3]。そこでの基本的な手法は、ループのイタレーション空間を、「タイル」と呼ばれるイタレーションの集まりで分割し（これをタイリングと呼ぶ）、各タイルをPEに割り付け、データ転送はタイルを単位として行うものである。

タイリング方式には、タイルを大きくとれば、通信回数は少なくてすむが、他のPEの開始が遅れて並列性が十分に生かせないというトレードオフがある。このため、タイルの大きさが全実行時間にどのような影響を及ぼすかを解析して最適なタイルサイズを決定する必要がある。

Ramanujamら[3]は、タイルの各辺の長さが等しいなどの特殊な条件下での最適なタイルサイズを示した。Hiranandaniら[2]もいくつかの特別な場合について最適タイルサイズを示している。しかし、一般的な条件下での最適タイルサイズの決定方式は、これまで提案されていなかった。本報告では、依存ベクタの方向が任意の場合に二次元タイリングを適用し、必ずしもタイルの各辺の長さが等しくなくてもよいという条件下で、最適タイルサイズを決定する方式を提案する。

Optimizing Tile Size for Parallelizing Wavefront Loops
Hiroshi OHTA, Yasuhiko SAITO, Masahiro KAINAGA and Hiroyuki ONO
Systems Development Laboratory, Hitachi Ltd.

2．タイルサイズの決定方式

(1) ウエーブフロント型ループの定義

n重完全ネスト(n>1)で、最外側ループとその一つ内側のループにループ運搬依存をもつループをウエーブフロント型ループと呼ぶ。外側二重ループのイタレーションを二次元イタレーション空間で表し、外側ループのインデックスを第1次元、内側ループのインデックスを第2次元で表す。

例えば、次の二重ループはウエーブフロント型ループである（ここで e,g>0, f>=0とする）。

```
for (i=1; i<=N; i++)
    for (k=1; k<=M; k++)
        A[i][k] = ( A[i-e][k+f], A[i][k-g]を含む計算)
```

このループにはフロー依存があり、依存ベクタの集合{(e,-f), (0,g)}で表せる（図1）。

(2) 二次元タイリング

タイリングは外側二重ループに対して行う。タイル形状は依存関係を守るような平行四辺形タイルとする。上の例の場合、イタレーション空間を図1のように平行四辺形タイルで分割する。

ここでタイルサイズbは、配列データをPEに一次元非巡回ブロック分割するときのブロックサイズと一致させる。また、依存関係を守るために、第2次元方向で隣合う二つのタイル間で右のタイル内のイタレーションから左のタイル内へ依存ベクタが入らないようにaを決定する。タイルサイズSの決め方は後述。　第1次元方向に関してタイルをPEに分散割り付けし、各タイルはイタレーション実行前に他のPEからのデータ転送を受けて自PEのデータを更新し、タイル内処理を行い、自分が更新した値を他のPEのタイルに送信する。これがタイリングによるウエーブフロント型並列処理である。

(3) 最適タイルサイズの決定

次に実行時間をタイルサイズSの関数T(S)として見積もる。イタレーション空間の第1次元のサイズをN、第2次元のサイズをMとする。

t=（1イタレーションの処理時間）、c=（1タイルの通信時間/t）とおく。ここでcは転送メッセージサイズに拠らない定数とみなす。タイル1個の処理時間はt(bS+c)と近似でき、最後のPEが起動するまでに実行されるタイル数は(aN/(bS)+N/b)、最後のPEが実行するタイル数は(M/S)であるから、実行時間T(S)は

$$t(bS+c)\{ aN/(bS)+N/b+(M/S) \} \quad (式1)$$

と近似できる。実行時間T(S)を最少にするSは、方程式 T'(S) = 0 の解であり

$$S = \mathrm{floor}\{ c(aN+bM)/(bN) \}^{1/2} \quad (式2)$$

を得る。ここで、解Sを整数値で近似している。

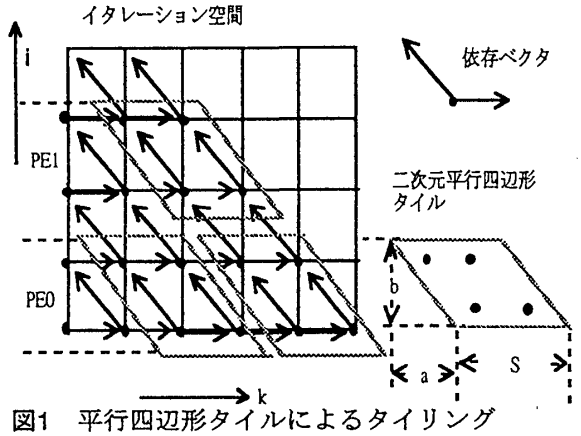

図1　平行四辺形タイルによるタイリング

3．iPSC/860での評価

本方式によってウエーブフロント型ループを並列化したSPMD型コードを8台構成のiPSC/860 上で実行して性能を計測した。

図2に、リバモア23番を8台で並列実行した場合の実行時間を示す。イタレーション空間のサイズはN=1000, M=300で、依存ベクタは{(1,0), (0,1)}であり、タイル形状は矩形である。計算モデルで見積った実行時間(式1)も示す。実測値とモデル値のグラフ形状は全体的によく似ており、最適タイルサイズ(S=3)も実測と理論値が一致している。実測値とモデル値が縦軸方向でずれる原因としては、1イタレーション当りの処理時間の見積もりが小さかったせいと思われる。また、タイルサイズ2と3の間が大きく開いているのは、キャッシュミスなどのメモリ遅れによるものと思われる。

図3に並列化したプログラムの性能向上比を示す。イタレーション空間のサイズがN=200,M=1000で、依存ベクタが{(1,-1), (0,1)}となる二重ループ（タイル形状は非矩形）とリバモア23番を示す。

4．結論

タイリングによるウエーブフロント型並列処理において、タイルサイズを最適に設定する方式を提案した。この方式は、ループネストの依存ベクタが一般の場合でも適用可能であり、並列化されたプログラムの効率は54%～80%である。今後の課題としては、多次元ループへの拡張、キャッシュを考慮したタイリング方式の開発があげられる。

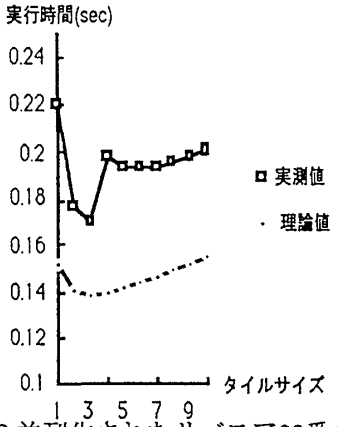

図2 並列化されたリバモア23番の実測値と理論値

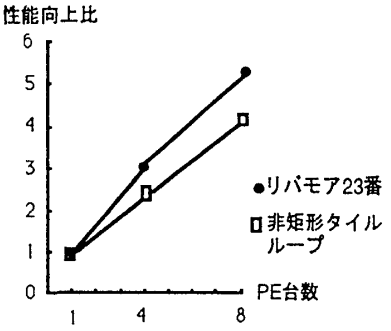

図3　性能向上比

5．参考文献

[1]M.E.Wolf and M.S.Lam: A Loop Transformation Theory and an Algorithm to Maximize Parallelism: IEEE TRANS.on Parallel and distributed Systems, pp.452-471, Vol.2, No.4, Oct. 1991

[2] Hiranandani、他: Evaluation of Compiler Optimizations for Fortran D on MIMD Distributed-Memory Machines: Proc. of 1992 Int'l Conf. on Supercomputing, pp.1-14, (1992-7)

[3] Ramanujam,他: Tiling Multidimensional Iteration Spaces for Multicomputers: J. of Parallel and Distributed Computing ,16,pp. 108-120(1992)