メッシュバス計算機上でのデータの移動及び順位決定に要する時間の上下限

5B−8

宮野 英次† 岩間 一雄† 上林 弥彦‡
(九州大学工学部† 京都大学工学部‡)

1. まえがき

メッシュバス計算機（MB，図1）は，メッシュ計算機（MC，図2）の局所通信機能を二次元メッシュ状に配置されたバスによるグローバル通信機能に置き換えた並列モデルである．本モデルはMCに比べて，それ程劣らないハードウェア的実現性を有し，かつ，数多くの問題に対しMCより速いアルゴリズムを提供してくれる．特に，グラフ問題に対しては，連結成分問題等の基本的問題に対しPRAMモデルにも匹敵する対数時間アルゴリズムを実現できることが知られている[1]．しかるに，MBの（MCに比べての）唯一ともいって良い弱点がソーティングであり，$O(n)$時間のアルゴリズムの実現が困難視されていた．しかし，最近，2項分布ソートと呼ばれる新たなソーティングアルゴリズムによってこの弱点も一応解消された[2]．

本論文では，まずこの新アルゴリズムを紹介し，その実用的環境におけるいくつかの改良の可能性を考察する．さらに，本アルゴリズムの評価（どの程度下限に近いか）へ向けての一ステップとして，より基本的な（それ自身としても重要な）ラウティング問題とデータの順位決定（計数ソート）に要する時間の上下限について議論する．

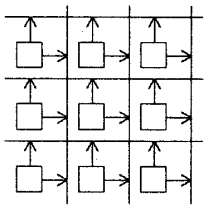

図1

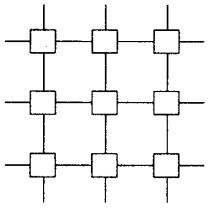

図2

2. 2項分布ソート

局所通信のMCにおいては$n \times n$のデータに対して$O(n)$時間でソートするアルゴリズムが数多く知られているが（例えば[3]），それらは例外なく再帰的な分割統治法を利用している．（典型的なものとして，$n \times n$を$\frac{n}{2} \times \frac{n}{2}$に4分割し，それぞれをさらに4分割するといった手法．）グローバル通信のMBの場合，バスを“切る”ことができないので，このように局所的に他と独立な通信を行なわせることができず，従って上記の様なMC上での$O(n)$アルゴリズムの応用を不可能なものにしていた．

新アルゴリズムである2項分布ソートは，この様な再帰的な分割統治法によらず，行方向のソートと列方向のソートのみを基本演算として全体をソートする．以下に与えるように，$n \times n$の2次元のデータを行優先の蛇順にソートする$O(n)$時間の確率アルゴリズムである．

ステップ1．データを行方向にでたらめに並べ換える（乱化）．
ステップ2．列方向にソートする．
ステップ3．各行を奇数行は左から右，偶数行は右から左へソートする．蛇順ソートが完成していれば終了．
ステップ4．列方向にソートする．
ステップ5．行方向に乱化した後ステップ2へ．

列方向ソートは，1つの列のn個のプロセッサが1本の列方向バスを利用し，いわゆる計数ソートを行なった後並べ換えるというやり方で，$O(n)$の時間で全ての列を並列に実行できる．ステップ1と5は，先ず各プロセッサが乱数を発生させ，その乱数の値によって，同様の行方向ソートを行なえば良い．

従ってアルゴリズム全体の計算時間が（高い確率で）$O(n)$になるためにはステップ2〜5の繰り返しの回数が$O(1)$にならなければならない．このことは直観的にはかなり困難にみえるが，アルゴリズムの名称にも採用された2項分布曲線の対称性を考慮に入れると展望が開けてくる．入力データが0と1のみから成ると仮定すると（0／1原理），ステップ2の後で，各列の1の個数の分布が2項分布になっていることが判る．この分布曲線が，0と1の個数に極端な差がない限り，平均値を中心にして左右対称になるという良く知られた事実を最大限に利用することが本アルゴリズムの基本戦略である．

実際の証明は，上記アルゴリズムのままでは困難で，列方向の部分的乱化や，マトリックスの左右周辺部のデータの移動等の改良が必要になる[2]．しかし，シミュレーションによれば，上記のアルゴリズムはそのままでもかなり良く振舞う．$n \times n$が250^2，500^2，10^6，4×10^6，1.6×10^7の各値において，全ての試行（各値に対し最低50回）に対してステップ2〜5の繰り返し回数が4回で終了しており，実用的には十分である．それ以外にも実用的利点がいくつかあり，その一つは，データの形状（メッシュバス計算機の形状）が正方である必要がないということである．本アルゴリズムは$n \times m$の長方形データに対しても（時間の評価は別として）そのまま動作する．このことは特に，プロセッサ数以上のデータを処理する必要のある場合に有効となる．つまり，1台のプロセッサに何個かのデータを処理させる（つまり何台かのプロセッサの動作をシミュレートさせる）時に，最適なnとmの比を考慮に入れたシミュレーションを行なわせることが可能になる．

3. メッシュバス計算機上でのラウティング

2項分布ソートは（高い確率で）cn時間でソートを完了するが，その系数cの値は必ずしも小さいとはいえない．その良さの評価を行なう最良の方法はメッシュバス上でのソーティング時間の（自明でない）下限を与えることであるが，それは困難な問題であるので，とりあえずソーティングの部分問題でありそれ自身重要なラウティング問題の計算複雑さを考察してみよう．ラウティングとは，$n \times n$のデータとその行き先の対（2つ以上のデータが同じ行き先を持つことはない）が与えられた時，全てのデータをその行き先へ移動させることである．なお，局所計算のMC上でのソーティングについてはその上下限が既にほぼ一致しているが，その下限はほとんどラウティングの下限そのものであるといっても良い．

本論文では以下の仮定のもとに議論を進める．（1）各プロセッサの行なう“1ステップ”の動作は，(i) 行及び列のバスのデータを読み込み，(ii) 定数回の内部計算を行ない，(iii) 必要なら行と列（あるいは一方）のバスへデータを書き込む（その値が次のステップで読み込まれる）ことから成る．（2）1本のバスへ2台以上のプロセッサが同時にデータを書き込むことを許さない．（3）バスに書き込める値は入力として与えられたデータと行き先の1つの組（必ずしも最初に自分に与えられたものでなくても良い）に限る．

初歩的なアルゴリズムとしては以下のものが考えられる．

ステップ1．各データを，左から順に，行バスを利用して行き先の列の位置まで移動させる．
ステップ2．今度は，各データをその行き先の行の順に，列方向に移動させる．

ステップ1の後では，1台のプロセッサに2個以上のデータが駐留する可能性があるので，ステップ2の移動の順番に注意を払わねばならない（単純に上のプロセッサからという訳にはいかない）．

Time Bounds for Routing and Counting Sort Problems on the Mesh of Buses
Eiji Miyano, Kazuo Iwama and Yahiko Kambayashi
Faculty of Engineering. Kyushu University and Faculty of Engineering. Kyoto University

定理１．ラウティングは$2n$ステップで実行できる．

逆に，$\frac{n}{2}-1$ステップでラウティングができたと仮定してみる．すると，あるプロセッサPで，Pは$\frac{n}{2}-1$ステップのどのステップでも，行方向バスにも列方向バスにもデータを書き込まず，かつ，それらのバスは全てのステップにおいて他のいずれかのプロセッサが書き込みを行なっているようなものを見つけることができる．Pへの入力の値を変化させても依然としてPはバスへの書き込みができない（データの衝突を招く）という事実を導くことにより以下の下限が得られる．

定理２．ラウティングは$\frac{n}{2}-1$ステップではできない．

定理１と２の上下限の開きはかなり大きい．現在のところ以下の様な自明でない改良に成功している．

定理３．ラウティングは$1.5n$ステップでできる．

証明．定理１では各データをステップ１で行き先の列の位置まで移動させ，ステップ２で行の位置に移動させたが，ステップ１では列方向のバスが，ステップ２では行方向のバスが未使用であるために，$2n$ステップを必要とした．そこで，なるべく多くのバスを並列に利用することを考える．

ステップ１　左上の$\frac{n}{2}\times\frac{n}{2}$個の各データは左から順に，右下の$\frac{n}{2}\times\frac{n}{2}$個については右から順に行バスを利用して行き先の列の位置まで移動させる．左下の$\frac{n}{2}\times\frac{n}{2}$個の各データは下から順に，右上の$\frac{n}{2}\times\frac{n}{2}$個については上から順に列バスを利用して行き先の行の位置まで移動させる（図３）．こうして，$\frac{n}{2}$ステップで各データは少なくとも行か列のいずれか一方の行き先までは，正しく移動しているはずである．

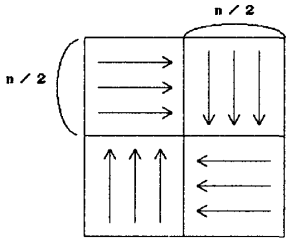

図３

ステップ２　今度は，各データをその行き先の行の順に列方向に移動させ，同時に行き先の列の順に行方向に移動させる．例えば，ステップ１の実行後あるプロセッサが，行き先が１行と３行のデータ（列は正しい）と３列のデータ（行は正しい）を持っているとすると，１ステップ目に行き先が１行であるデータを列バスに書き込み，２ステップ目は何も行なわない（行き先が２行，２列のデータを持つ他のプロセッサがそれぞれバスへ書き込みを行なう）．３ステップ目に３行のデータを列バスに，３列のデータを行バスに書き込む（このとき同時書き込みはない）．こうして，nステップで各データは正しい位置に移動することができる．□

定理４．各プロセッサに与えられる入力データの値の取り得る範囲をCとする．このとき，もし定数$\alpha(<1)$が$C>(\alpha n)^{\frac{\alpha}{1-\alpha}}$を満たすならば，ラウティングは$\alpha n$ステップではできない．

証明．各初期入力データが同一の行に移動するような入力列を考える．仮に，αn回で移動ができたとすると，各行について少なくとも$n-\alpha n$個のプロセッサは行方向バスへの書き込みをおこなうことはない．そのプロセッサPへの入力値を変化させてみる．Pが列バスにも書き込みを行なわないとすると，上と同様の考察により矛盾が生じる．つまりPは列バスへの書き込みを必ず行なうことになる．Pの値による影響が何らかの形で他の行バスに生じなければ，バスに書き込まれる値に変化はなく，入力データの変化を知ることはできない．行バスに値が載ると新たに行バスに書き込めないプロセッサが生じるというように，単に”値”をその値自身によって伝えるという方法では不可能に見える．しかし，値そのもののみではなく値がバスに載った順序など様々な情報が利用可能である．以下，１つの行バスにどの程度の情報量を載せることができるかを計算してみる．

各プロセッサがバスに書き込む値は，初期入力もしくは以前にバスから読み取った値であり，行方向バスに一度載った値の再送をしない場合について先に考察する．各行１ステップ目に書き込みを行なうプロセッサは一意に決まり，C通りの値を行方向バスに載せることができる．２ステップ目以降はバスの値により次に書き込みを行なうことのできるものが一意に決定し（そうでなければデータが衝突する），このプロセッサは最大で$2C$通りの値を書く（１ステップ目で列方向バスを利用して１個のデータが読み込まれている）．同様に，３ステップ目では$3C$通りとなり，αnステップまで考えることにより，

$$C\cdot 2C\cdot 3C\cdots(\alpha n)C=C^{\alpha n}\cdot(\alpha n)!$$

通りの異なった場合の数が考えられる．αnステップのうち例えばiとjステップで再送を行なったとすると，例えばiステップ目では，それまでバスを流れたi通りのデータの中から１つを選ぶことができる．よって，情報量としては，

$$C\cdot 2C\cdots(i-1)C\cdot i\cdot(i+1)C\cdots(j-1)C\cdot j\cdot(j+1)C\cdots(\alpha n)C$$

となる．iとjの選び方は${}_nC_2$通りあるので，２回再送を行なった場合は，

$$C^{\alpha n-2}{}_nC_2(\alpha n)!$$

通りの異なった場合の数となる．一般にKステップで再送を行なったとすると，

$$C^{\alpha n-k}{}_nC_k(\alpha n)!$$

となり，全体の情報量としては，次のようになる．

$$\sum_{k=1}^{\alpha n}C^{\alpha n-k}{}_nC_k(\alpha n)!$$

再送しない場合の数$C^{\alpha n}(\alpha n)!$と，１行の入力列の場合の数C^nが

$$C^n>C^{\alpha n}(\alpha n)!$$

となれば，入力の場合の数全てをバスに載せるデータの違いによって区別することができなくなる．

$$\begin{aligned}C^n &> C^{\alpha n}(\alpha n)^{\alpha n}\quad(>C^{\alpha n}(\alpha n)!)\\ logC &> \alpha\log C+\alpha\log\alpha n\\ C &> (\alpha n)^{\frac{\alpha}{1-\alpha}}\end{aligned}$$

Cがこのような値のときは，上記再送をする場合の情報量はしない場合に比べて無視できる程度に小さい．よって，Cの幅が上式を満たす時バス上を流れる情報量よりも入力列の情報量の方が大きくなってしまう．行バスはn本あるので送れる情報量は増えるが，入力の情報量も同様に増えるので，以上の解析で十分である．□

上下限の一致は以上よりnと$1.5n$の間で起こる訳であるが，これ以上の改良については手がかりに乏しく，現在のところ根拠のある予想も得られていない．また上記の（２）と（３）の仮定を緩めることについても考察を進めており，特に（２）についてはその除去はそれほど難しくないと思われる．

なお，ソーティングには，必ずしもデータの移動を必要とせず，各データの順位が計算できれば良いとの立場もある．このような立場でのソーティング（計数ソートと呼ばれることが多い）の下限については，上記定理２と４の証明がほとんどそのまま応用でき，同様の下限を得ることができる．

参考文献

[1] K.Iwama and Y.Kambayashi, "An $O(\log n)$ parallel connectivity algorithm on the mesh of buses", Proc. 11th IFIP World Computer Congress, 89, pp.305-310.

[2] 岩間, 宮野, 上林," メッシュバス機械上での並列ソーティングアルゴリズムについて", 情報処理学会アルゴリズム研究会資料, 18-2, 1990.

[3] C. Schnorr and A. Shamir, "An optimal sorting algorithm for mesh connected computers", 18th ACM Symposium on Theory of Computing, 86, pp.255-263.