プログラミング 18−13
(1998. 3. 23)

問題のサイズより小さい再構成メッシュ上でのソーティングアルゴリズム

松前 進　　笹田 良治　　都倉 信樹

大阪大学 大学院基礎工学研究科 情報数理系専攻
〒560-8531 大阪府豊中市待兼山町 1 番 3 号

Tel: (06)850-6584

E-mail: {matsumae, sasada, tokura}@ics.es.osaka-u.ac.jp

あらまし

格子状に並べたプロセッサを再構成バスで接続したプロセッサアレイを，再構成メッシュ(reconfigurable mesh) という．再構成バスの形状を変化させることにより，再構成メッシュではプログラムの実行時にネットワークトポロジを動的に変更することができる．

本稿では，大きさ$M \times M$の再構成メッシュを用いて，N 個の値を$O(\frac{N}{M} + \frac{N}{M^2} \log \frac{N}{M^2})$時間でソートするアルゴリズムを示す$(N > M)$.

キーワード　並列アルゴリズム，再構成メッシュ，ソーティング

A Sorting Algorithm on a Small Size Reconfigurable Mesh

Susumu MATSUMAE　　Ryoji SASADA　　Nobuki TOKURA

Graduate School of Engineering Science, Osaka University
1-3, Machikaneyama, Toyonaka, Osaka, 560-8531, Japan

Tel: (06)850-6584

E-mail: {matsumae, sasada, tokura}@ics.es.osaka-u.ac.jp

Abstract

A reconfigurable mesh is a processor array that consists of processors arranged to a 2-dimensional grid with a reconfigurable bus system. The bus system can be used to dynamically obtain various interconnection patterns among the processors during the execution of programs.

In this paper, we present an algorithm for sorting N elements in $O(\frac{N}{M} + \frac{N}{M^2} \log \frac{N}{M^2})$ time on a reconfigurable mesh of size $M \times M (N > M)$.

key words　parallel algorithm, reconfigurable mesh, sorting

1 はじめに

並列計算モデルの一つに，メッシュ結合計算機 (Mesh Connected Computer，以下 MCC) がある．MCC は，複数のプロセッサを格子状に配置し，隣接するプロセッサ間のみを通信リンクで接続したモデルである．隣接するプロセッサ間でしか通信が行えないため，このモデル上で動くアルゴリズムの時間計算量は，使用する MCC の通信直径がボトルネックとなる．

このような問題を解消するため，MCC を拡張したさまざまモデルが研究されている．その一つに再構成メッシュ(Reconfigurable Mesh，以下 RM) というものがあり，幾つか試作も行われている [1]．RM では再構成バスを用いて，プログラムの実行時にそのネットワークトポロジを動的に変更することができる．取りうるネットワークトポロジに制限を加えることにより，RM はさらに幾つかのクラスに分類される．それらのうち，本稿では HV-RM(Horizontal-Vertical RM) と L-RM(Linear RM) を扱う．

RM では，さまざまな問題を非常に高速に解くことが出来る．以下に例を示す．

- N 個の値のソートを大きさ $N \times N$ の L-RM を用いて$O(1)$時間で行う [2].
- 平面上の N 点が与えられたとき，その凸包を $N \times N$L-RM 上で$O(1)$時間で求める [3, 4].

上記のアルゴリズムをはじめとして，これまでに RM で提案されているアルゴリズムでは，入力の大きさ N に依存したプロセッサ数の RM が使用できることを仮定することが多い．しかし，このようなアルゴリズムを実装する際には，必要数のプロセッサが常に利用できるとは限らない．

この問題に対して，大きいサイズの RM の動作を小さいサイズの RM でシミュレートするための，セルフシミュレーションアルゴリズムが提案されている [5]．しかし，このセルフシミュレーションアルゴリズムは任意のプログラムを入力の対象としているため，得られる結果が常に効率の良いものになるとはいえない．本稿で扱うソートの場合，他のアルゴリズムの前処理などに用いられることも多く，その計算量がアルゴリズム全体の計算量に対して大きな割合を占めることも多い．しかし，[2] のソートアルゴリズムを [5] のセルフシミュレーションアルゴリズムを用いて実行させた場合，N 個の要素を$M \times M$L-RM 上でソートするのに $O((\frac{N}{M})^2)$時間かかる $(N > M)$.

そこで本稿では，最初から問題のサイズより小さいサイズの L-RM しか使用できない場合を考えてアルゴリズムを設計することにより，より効率の良い $O(\frac{N}{M} + \frac{N}{M^2} \log \frac{N}{M^2})$時間のソーティングアルゴリズムを示す．

本稿の構成は次の通りである．2 節でモデルの定義を行う．3 節では，以降で用いる置換アルゴリズムを与える．4 節で，本稿のソーティングアルゴリズムのもととなる Column sort[6] を紹介し，5 節でソーティングアルゴリズム SORT1，SORT2 を与える．最後に 6 節で結果をまとめる．

2 再構成メッシュ

RM は 2 次元格子状に配置されたプロセッサで構成される SIMD 型の並列計算モデルである．全てのプロセッサは同期して動作し，同じプログラムを実行する．M 行 N 列のプロセッサから成る RM を $M \times N$RM と書く．3×4RM を図 1 に示す．

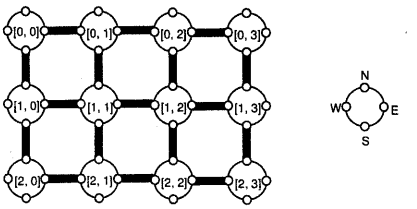

図 1: 3×4RM とプロセッサ

2 次元格子の i 行 j 列に位置するプロセッサを PE$[i, j]$ で表す．各 PE$[i, j]$ はそれぞれ自分の位置 i，j を知っているとし，それぞれ id_x，id_y で参照する．各プロセッサは局所メモリのみを持ち，共有メモリは存在しない．PE$[i, j]$ の局所変数 v は PE$[i, j]$.v で表す．

各プロセッサは N,S,E,W で表される 4 つのポートを持ち，隣接するプロセッサの互いに向かい合うポートは静的な外部バスで接続されている (図 1)．つまり，PE$[i, j]$ のポート S は PE$[i+1, j]$ のポート N と接続され $(0 \leq i < M - 1, 0 \leq j < N)$，同様に，PE$[i, j]$ のポート E は PE$[i, j+1]$ のポート W と接続されている $(0 \leq i < M, 0 \leq j < N - 1)$.

また，内部バスを用いて，各プロセッサは自分の持つ 4 つのポート間を接続することができる．図 2 に全 15 種類の内部バスの形状を示す．内部バスの形状はプログラムの実行時に動的に変更することができる．接続されたバスは全体で一本のバスとみなすことができ，バスの伝送遅延時間はその長さにかかわらず定数時間であるとする．また，同一バスにデータを送信するプロセッサは高々一つであるとする．PE$[i, j]$ の内部バスの形状を図 2 のいずれかに変更することを，PE$[i, j]$:{NW,SE} などと書く．PE$[i, j]$ がポート X にデータ y (y が変数の場合はその値) を送信することを PE$[i, j]$:y $\rightarrow$ X，ポート X からデータを受信して変数 z に代入することを PE$[i, j]$:z $\leftarrow$ X と書く．

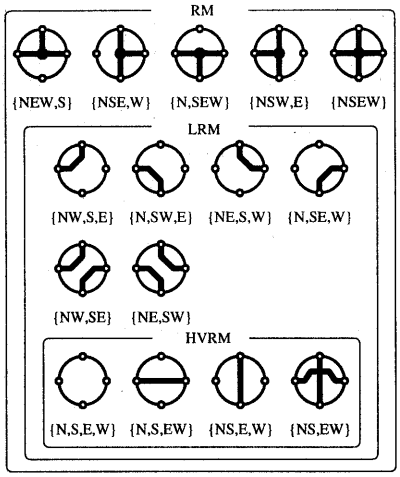

図 2: 内部バスの形状

各プロセッサは単位時間に以下の 4 つの動作を順番に行う.

1. 内部バスの形状を変更
2. 各ポートにデータを送信
3. 各ポートからデータを受信
4. RAM の命令を定数個実行

アルゴリズムの時間計算量は, 上記の 4 つの動作を 1 ステップとしてステップ数で評価する.

本稿では, HV-RM と L-RM のみを扱う. HV-RM とは, 取り得る内部バスの形状を図 2 の {N,S,E,W}, {N,S,EW}, {NS,E,W}, {NS,EW} の 4 つに制限したモデルである. L-RM とは, HV-RM の 4 つに加え, 内部バスの形状を {NW,S,E}, {N,SW,E}, {NE,S,W}, {N,SE,W}, {NW,SE}, {NE,SW} の, 計 10 種類に制限したモデルである. モデルの定義から, HV-RM 上で動くアルゴリズムは L-RM 上でも同じ時間計算量で動くことがいえる.

3 置換アルゴリズム

5 節で述べるソーティングアルゴリズムでは, 行列の各要素を複数のプロセッサに分散して保持させる. また, アルゴリズム中で, その行列の各要素の位置を入れ換える操作を行う. そこで本節では, 各プロセッサに保持される値の置換先が与えられたとき, その置換を実際に行うアルゴリズムを示す. 簡単のため, $N \bmod M = 0$ とする.

補題 3.1 (置換) 以下の問題は, $M \times M$HV-RM 上で $O(KM)$時間で解くことができる. ここで, $\mathcal{A}[i,j,k]$, $\mathcal{B}[i,j,k]$, $\mathcal{C}[i,j,k]$ はそれぞれ PE$[i,j]$.a$[k]$, PE$[i,j]$.b$[k]$, PE$[i,j]$.c$[k]$ を表す $(0 \le i,j < M, 0 \le k < K)$. また, b$[k](0 \le k < K)$ は値として (x,y,z) の 3 つ組をとる.

入力: $\mathcal{A}[i,j,k]$ に保持される要素の置換先として, $(x,y,z) \in \{0,\ldots,M-1\}^2 \times \{0,\ldots,K-1\}$ が $\mathcal{B}[i,j,k]$ に与えられる $(0 \le i,j < M, 0 \le k < K)$. $\mathcal{B}[i,j,k](0 \le i,j < M, 0 \le k < K)$ の値はそれぞれ相異なる.

出力: $0 \le i,j,i',j' < M, 0 \le k,k' < K$ に対し,

$$\mathcal{B}[i,j,k] = (i',j',k') \Longrightarrow \mathcal{C}[i',j',k'] = \mathcal{A}[i,j,k]$$

証明 上記の問題を解くアルゴリズム CUBIC_PERM を図 3 に示す. アルゴリズムの表記中, PE$[*,*]$ は全ての PE, つまり PE$[i,j](0 \le i,j < M)$ を表す. 同様に, PE$[x,*]$ は x 行目の全ての PE を表す. また, 変数 v が $(x,y,z) \in \{0,\ldots,M-1\}^2 \times \{0,\ldots,K-1\}$ の値を持つとき, x,y,z をそれぞれ v.x, v.y, v.z で参照するとする. アルゴリズムの詳細については略. ■

Algorithm CUBIC_PERM

{ a$[0,\ldots,K-1]$,b$[0,\ldots,K-1]$,c$[0,\ldots,K-1]$ は問題の入出力に使用される変数. t1,t2,t3, t4$[0,\ldots,M-1][0,\ldots,K-1]$ は作業用変数. t4 は ϕ に初期化されている. }

1: for x:=0 to $M-1$ do
  for y:=0 to $K-1$ do
    PE$[*,*]$:{NS, E, W}
    PE$[x,*]$:a$[y] \rightarrow$ S
    PE$[i,i]$ $(0 \le i < M)$:t1 $\leftarrow$ N
    PE$[x,*]$:b$[y] \rightarrow$ S
    PE$[i,i]$ $(0 \le i < M)$:t2 $\leftarrow$ N
    PE$[*,*]$:{N, S, EW}
    PE$[i,i]$ $(0 \le i < M)$:t2 $\rightarrow$ E
    PE$[*,*]$:t3 $\leftarrow$ W
    PE$[i,i]$ $(0 \le i < M)$:t1 $\rightarrow$ E
    PE$[*,*]$:if t3.y=id$_y$ then t4[t3.x][t3.z] $\leftarrow$ W
2: PE$[*,*]$:{NS, E, W}
3: for x:=0 to $M-1$ do
  for y:=0 to $K-1$ do
    PE$[*,*]$:if t4[x][y]$\neq \phi$ then t4[x][y] $\rightarrow$ N
    PE$[x,*]$:c$[y] \leftarrow$ S

end of CUBIC_PERM

図 3: アルゴリズム CUBIC_PERM

4 Column Sort

本稿で提案するアルゴリズムは，Leighton によって提案された Column sort[6] を L-RM 上で行うことによりソートを行う．よって，まず本節で Column sort について説明する．

Column sort は，$r \times s$ 行列 Q を column major order にソートするアルゴリズムである．ここで，r と s は条件 $r \geq 2(s-1)^2$, $r \bmod s = 0$ を満たす必要がある．Q が 9×3 行列の場合の入出力例を図 4 に示す．

$$\begin{pmatrix} 13 & 10 & 24 \\ 1 & 26 & 6 \\ 5 & 16 & 11 \\ 21 & 8 & 25 \\ 9 & 23 & 4 \\ 22 & 12 & 18 \\ 3 & 17 & 7 \\ 27 & 2 & 19 \\ 14 & 15 & 20 \end{pmatrix} \xrightarrow{\text{Column sort}} \begin{pmatrix} 1 & 10 & 19 \\ 2 & 11 & 20 \\ 3 & 12 & 21 \\ 4 & 13 & 22 \\ 5 & 14 & 23 \\ 6 & 15 & 24 \\ 7 & 16 & 25 \\ 8 & 17 & 26 \\ 9 & 18 & 27 \end{pmatrix}$$

図 4: 9×3 行列を column major order にソート

Column sort は行列 Q に対し，以下の 4 つの操作を順に行う．なお，下記中の Shift, Unshift は [6] での定義と若干異るが，Q が column major order にソートされるという結果に影響を与えることはない．

Phase 1: 各列を下向きにソートした後，Transpose.
Transpose では Q の i 行 j 列の要素を $(jr+i)$ div s 行 $i \bmod s$ 列の位置に置換 (図 5).

Phase 2: 各列を下向きにソートした後，Untranspose.
Untranspose は Transpose の逆変換 (図 5).

Phase 3: 各列を下向きにソートした後，Shift.
Shift は Q の i 行 j 列の要素を $(i + \lfloor \frac{r}{2} \rfloor) \bmod r$ 行 $(j + i \text{ div } \lceil \frac{r}{2} \rceil) \bmod s$ 列の位置に置換 (図 6).

Phase 4: 最左列を除いて各列を下向きにソートした後，Unshift.
Unshift は Shift の逆変換 (図 6).

$$\begin{pmatrix} 1 & 10 & 19 \\ 2 & 11 & 20 \\ 3 & 12 & 21 \\ 4 & 13 & 22 \\ 5 & 14 & 23 \\ 6 & 15 & 24 \\ 7 & 16 & 25 \\ 8 & 17 & 26 \\ 9 & 18 & 27 \end{pmatrix} \begin{matrix} \text{Transpose} \\ \longrightarrow \\ \longleftarrow \\ \text{Untranspose} \end{matrix} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \\ 13 & 14 & 15 \\ 16 & 17 & 18 \\ 19 & 20 & 21 \\ 22 & 23 & 24 \\ 25 & 26 & 27 \end{pmatrix}$$

図 5: Transpose / Untranspose

$$\begin{pmatrix} 1 & 10 & 19 \\ 2 & 11 & 20 \\ 3 & 12 & 21 \\ 4 & 13 & 22 \\ 5 & 14 & 23 \\ 6 & 15 & 24 \\ 7 & 16 & 25 \\ 8 & 17 & 26 \\ 9 & 18 & 27 \end{pmatrix} \begin{matrix} \text{Shift} \\ \longrightarrow \\ \longleftarrow \\ \text{Unshift} \end{matrix} \begin{pmatrix} 24 & 6 & 15 \\ 25 & 7 & 16 \\ 26 & 8 & 17 \\ 27 & 9 & 18 \\ 1 & 10 & 19 \\ 2 & 11 & 20 \\ 3 & 12 & 21 \\ 4 & 13 & 22 \\ 5 & 14 & 23 \end{pmatrix}$$

図 6: Shift / Unshift

5 ソーティングアルゴリズム

まず，$M \times M$L-RM 上でのソーティング問題を次のように定義する．簡単のため，$N \bmod M = 0$ とする．

定義 5.1 (ソーティング問題) $M \times M$L-RM 上で N 個の値をソートする問題を次のように定義する．ここで，$N > M$ とし，$\alpha[i]$ は PE$[0, i \bmod M]$.a$[i \text{ div } M]$ を表すものとする $(0 \leq i < N)$.

入力: N 個の値が $\alpha[0, \ldots, N-1]$ に与えられる
出力: $\alpha[0, \ldots, N-1]$ はソートされている

■

本稿で提案するソーティングアルゴリズムでは，入力される N 個の値を $r \times s$ 行列 Q に対応づけ，Column sort を用いて Q を column major order にソートする．Q は図 7 で与える $(rs = N)$.

$$Q = \begin{pmatrix} \alpha[0] & \alpha[r] & \cdots & \alpha[(s-1)r] \\ \alpha[1] & \alpha[r+1] & \cdots & \alpha[(s-1)r+1] \\ \vdots & \vdots & \ddots & \vdots \\ \alpha[r-1] & \alpha[2r-1] & \cdots & \alpha[rs-1] \end{pmatrix}$$

図 7: $r \times s$ 行列 Q

Column sort を行うには，r と s が条件 $r \geq 2(s-1)^2$ を満たす必要がある．以降では，$s = M^2$ が可能な場合 $(N \geq 2(M^2-1)^2M^2)$ と，$s = M^2$ が可能でない場合 $(N < 2(M^2-1)^2M^2)$ の，2 つのケースに分けてアルゴリズムを示す．

5.1 $N \geq 2(M^2-1)^2M^2$ の場合

このとき，$r = \frac{N}{M^2}$, $s = M^2$ とおくと，

$$r = \frac{N}{M^2} \geq 2(M^2-1)^2 = 2(s-1)^2$$

となり，条件 $r \geq 2(s-1)^2$ を満たす．簡単のため $N \bmod M^4 = 0$ とする．r, s はともに正整数，$r \bmod s =$

0となる．よって，QにColumn sortを適用できる．以下，L-RM上でどのようにColumn sortを行うかを述べる．

$Q_{i,j}$はPE[j mod M, j div M].q[i]で保持する．つまり，Qのj列目の要素はPE[j mod M, j div M]の局所変数q[$0,\ldots,r-1$]によって保持される．Qの要素をこのように各プロセッサに割り当てることにより，Qの各列のソートは各PEが独立して行うことができる．Qに対するTranspose, Untranspose, Shift, Unshiftの各操作は，3節のCUBIC_PERMを用いて行う．以上を行うアルゴリズムSORT1を図8に示す．

Algorithm SORT1

{ $\alpha[i]$はPE[0, i mod M].a[i div M]$(0 \le i < N)$, $Q[i,j]$はPE[j mod M, j div M].q[i]を表す$(0 \le i < r, 0 \le j < s)$ }

Phase 0: $\alpha[i]$の値を$Q[i \bmod r, i \operatorname{div} r]$へ格納．
Phase 1: 各PEはq[$0,\ldots,r-1$]を逐次マージソート．QをTranspose.
Phase 2: 各PEはq[$0,\ldots,r-1$]を逐次マージソート．QをUntranspose.
Phase 3: 各PEはq[$0,\ldots,r-1$]を逐次マージソート，QをShift.
Phase 4: 各PEはq[$0,\ldots,r-1$]を逐次マージソート(PE[0,0]は除く)．QをUnshift.
Phase 5: $Q[i,j]$の値を$\alpha[jr+i]$へ格納．

end of SORT1

図 8: アルゴリズム SORT1

補題 5.1 *アルゴリズムSORT1における，Phase 0, Phase 5は，いずれも*$O(\frac{N}{M})$*時間で行うことができる．*

証明 Phase 0の，$\alpha[i]$の値を$Q[i \bmod r, i \operatorname{div} r]$へ格納するアルゴリズムを図9に与える．

図9の1では，PE[0,j]の局所変数a[$0,\ldots,\frac{N}{M}$]に，$\alpha[j\frac{N}{M}],\ldots,\alpha[(j+1)\frac{N}{M}-1]$に保持されていた値が格納される．つまり，PE[0,j]には，QのjM列から$(j+1)M-1$列まで，M列分の要素が集められる．QのjM列から$(j+1)M-1$列の要素は，j列目に位置するPE[$*$,j]がそれぞれ保持するので，後は列バスを用いてこれらの値を下方に送るだけでよい．これは3で行う．

1は[7]の置換アルゴリズムを用いて$O(\frac{N}{M})$時間で実行できる．2, 3は，定義から$O(\frac{N}{M})$ステップで実行できる．以上より，全体として$O(\frac{N}{M})$時間でPhase 0が実行できる．

Phase 5については略． ■

1: 文献[7]の置換アルゴリズムを用いて，
　$\alpha[i]$を$\alpha[(i \bmod \frac{N}{M})M + (i \operatorname{div} \frac{N}{M})]$へ置換
　$(0 \le i < N)$.
2: PE[$*,*$]: {NS, E, W}
3: for x:=0 to $M-1$ do
　　for y:=0 to $r-1$ do
　　PE[0,$*$]: a[xr+y] → S
　　PE[x,$*$]: q[y] ← N

図 9: SORT1 の Phase 0 を行うアルゴリズム

補題 5.2 *アルゴリズムSORT1における，Phase 1から Phase 4までは，いずれも*$O(\frac{N}{M} + \frac{N}{M^2}\log\frac{N}{M^2})$*時間で行うことができる．*

証明 QのTranspose, Untranspose, Shift, Unshiftに関しては，アルゴリズムCUBIC_PERMを用いてQを置換すればよい．例えばTransposeでは，$Q[i,j]$に保持されている値を$Q[(jr+i) \operatorname{div} s, i \bmod s]$へと移動させる．各PEは$Q$の要素を$r$個づつ保持しているので，各PEがそれらの置換先を計算するのに$O(r)$時間かかる．その後，CUBIC_PERMを用いて実際に置換を行う．補題3.1より，このCUBIC_PERMの実行に必要な時間は$O(rM)$である．よって，Transposeは$O(r+rM)$=$O(rM)$=$O(\frac{N}{M})$時間で実行できることがいえる．Untranspose, Shift, Unshiftに関しても同様に，$O(\frac{N}{M})$時間で実行できることを示せる．

また，各PE[i,j]がq[$0,\ldots,r-1$]を逐次マージソートするのに必要な時間は$O(r\log r)$=$O(\frac{N}{M^2}\log\frac{N}{M^2})$である．

以上より，Phase 1から Phase 4は，いずれも$O(\frac{N}{M} + \frac{N}{M^2}\log\frac{N}{M^2})$時間で実行できることがいえる． ■

定理 5.1 $N \ge 2(M^2-1)^2M^2$ *とする．このとき，アルゴリズムSORT1を用いて*$O(\frac{N}{M} + \frac{N}{M^2}\log\frac{N}{M^2})$*時間でソーティング問題を解くことができる．*

証明 SORT1のPhase 0で，αに保持されている値は$Q[i,j]$=$Q_{i,j}$となるようにQへ格納される．以降，Phase 1から Phase 4まででQをColumn sortする．Phase 4の終了時点でQに保持されている値はcolumn major orderにソートされている．最後にPhase 5で，$Q[i,j]$の値を$\alpha[jr+i]$に格納する．以上より，最初にαに与えられた値はソートされることがいえる．

補題5.1より Phase 0, Phase 5にかかる時間はいずれも$O(\frac{N}{M})$である．また，補題5.2より Phase 1から Phase 4にかかる時間はいずれも$O(\frac{N}{M} + \frac{N}{M^2}\log\frac{N}{M^2})$である．したがってSORT1全体の時間計算量は$O(\frac{N}{M} + \frac{N}{M^2}\log\frac{N}{M^2})$であることがいえる． ■

5.2 $N < 2(M^2-1)^2M^2$ の場合

この場合，$s = M^2$，$r = \frac{N}{M^2}$ とおくと，

$$2(s-1)^2 = 2(M^2-1)^2 > \frac{N}{M^2} = r$$

となり，Column sort を実行するために必要な条件を満たすことができない．実際，$s < M^2$ でなければならないことが分かる．

そこで，Column sort を行うのに必要な条件を満たすように，$s = \frac{1}{2}N^{\frac{1}{3}}$，$r = 2N^{\frac{2}{3}}$ とおくことにする．簡単のため $\frac{1}{2}N^{\frac{1}{3}}$ は正整数になるとする．しかし，$s < M^2$ であるため，SORT1 の場合と同様の$\mathcal{Q}$のマッピングを行った場合，$\mathcal{Q}$の各列のソートの際に $M^2 - s$ 個のプロセッサが idle 状態となり，効率が悪い．

例えば $N = 8M^3$ の場合を考えてみる．このとき $r = 8M^2$，$s = M$ であるから，Q は $8M^2 \times M$ 行列となる．SORT1 では Q の各列は s 個のプロセッサにそれぞれ 1 列分づつ割り当てられるため，Phase 1 から Phase 4 のソートの際には M 個のプロセッサしかソートに使用されない．また，この ($\mathcal{Q}$の一列分の) ソートに要する時間が$O(r\log r)$=$O(8M^2\log(8M^2))$=$O(\frac{N}{M}\log M)$時間となってしまい，効率が悪い．

以上のことから，各列のソートをそれぞれ一つの PE で行うのではなく，M^2/s 個の PE から構成されるサブメッシュを用いて行うことにする．

簡単のため $M/\sqrt{s} = K$ となる正整数 K が存在するとする．サブメッシュへの分割は，$M \times M$L-RM を $K \times K$ の大きさのサブメッシュS[i]($0 \le i < s$) に分割することにより行う．S[i] は PE[x, y]($(i \bmod \sqrt{s})K \le x < (i \bmod \sqrt{s}+1)K$, $(i \operatorname{div} \sqrt{s})K \le y < (i \operatorname{div} \sqrt{s}+1)K$) で構成される．また，S[$i$] の x 行 y 列に位置するプロセッサを $\mathrm{PE}_i[x, y]$ で表す．つまり，$\mathrm{PE}_i[x, y] = \mathrm{PE}[(i \bmod \sqrt{s})K + x, (i \operatorname{div} \sqrt{s})K + y]$ である．以下，L-RM 上でどのように Column sort を行うかを述べる．

$Q_{i,j}$ は $\mathrm{PE}_j[0, i \bmod K]$.q[$i \operatorname{div} K$] で保持する．つまり，$\mathcal{Q}$ の j 列目の要素はサブメッシュS[j] の最上行のプロセッサによって保持される．$\mathcal{Q}$ の各列のソートはその列を保持している各サブメッシュが独立して行う．Transpose，Untranspose，Shift，Unshift の各操作は，3 節の CUBIC_PERM に少し変更を加えたもので行う．以上の処理を行うアルゴリズム SORT2 を図 10 に示す．

補題 5.3 アルゴリズム *SORT2* における，*Phase 0*, *Phase 5* は，いずれも$O(\frac{N}{M})$時間で行うことができる．

証明 Phase 0 については，補題 5.1 の証明で示したアルゴリズムに少し変更を加えるだけで行える．詳細については略．Phase 5 についても同様． ■

補題 5.4 アルゴリズム *SORT2* における，$\mathcal{Q}$の *Transpose, Untranspose, Shift, Unshift* は，いずれも$O(\frac{N}{M})$時間で行うことができる．

Algorithm SORT2

{ $\alpha[i]$ は PE[0, $i \bmod M$].a[$i \operatorname{div} M$]($0 \le i < N$), $\mathcal{Q}[i, j]$ は $\mathrm{PE}_j[0, i \bmod K]$.q[$i \operatorname{div} K$] を表す ($0 \le i < r, 0 \le j < s$) }

Phase 0: $\alpha[i]$ の値を$\mathcal{Q}[i \bmod r, i \operatorname{div} r]$ へ格納．

Phase 1: 各サブメッシュS[i] は$\mathcal{Q}[*, i]$ をソート．$\mathcal{Q}$ を Transpose．

Phase 2: 各サブメッシュS[i] は$\mathcal{Q}[*, i]$ をソート．$\mathcal{Q}$ を Untranspose．

Phase 3: 各サブメッシュS[i] は$\mathcal{Q}[*, i]$ をソート．$\mathcal{Q}$ を Shift．

Phase 4: 各サブメッシュS[i] は$\mathcal{Q}[*, i]$ をソート (S[0] は除く)．$\mathcal{Q}$を Unshift．

Phase 5: $\mathcal{Q}[i, j]$ の値を $\alpha[jr + i]$ へ格納．

end of SORT2

図 10: アルゴリズム SORT2

証明 3 節の CUBIC_PERM を少し変更したもので行える．詳細については略． ■

次に，サブメッシュS[i] で$\mathcal{Q}[*, i]$ をソートする際に用いるアルゴリズム MERGESORT を示す．MERGESORT は，$M \times M$L-RM 上での N 個の値のソーティング問題を$O(\frac{N}{M}\log\frac{N}{M})$時間で解く．MERGESORT を与える前に MERGESORT で用いる幾つかのアルゴリズムを先に述べる．

補題 5.5 (前半部分の取り出し) 以下で定義される問題は，$M \times M$L-RM 上で定数時間で解くことができる．ここで，$\alpha[i]$, $\beta[i]$, $\gamma[i]$ はそれぞれ PE[0, i].a, PE[0, i].b, PE[0, i].c を表す ($0 \le i < M$)．

入力: 長さ M の 2 本のソート済みリストが，$\alpha[0, \ldots, M-1]$, $\beta[0, \ldots, M-1]$ にそれぞれ与えられる．

出力: $\gamma[0, \ldots, M-1]$ に，$\alpha[0, \ldots, M-1]$ と $\beta[0, \ldots, M-1]$ をマージしてソートすることにより得られるリストの前半部分が格納される．つまり，$\alpha[0, \ldots, M-1]$ と $\beta[0, \ldots, M-1]$ をマージしてソートした結果のリストを $L[0, \ldots, 2M-1]$ とすると，$\gamma[i] = L[i]$ が成り立つ ($0 \le i < M$)．

また，全 PE は γ 中に含まれる α の要素の数 l_a，β の要素の数 l_b を知る．

証明 上記の問題を解くアルゴリズムを図 11 に示す．

GET_FRONT_HALF では，1 から 3 で α, β の要素のうち小さい方から M 個の要素を γ へ取り出す．次に

4から13でl_aと$l_b(=M-l_a)$を求め，全プロセッサにブロードキャストする．最後に14でγをソートすることにより，問題の出力におけるγとしている．

1の置換操作は定数時間で行うことができる([7]参照)．また，14のソートも[2]のソーティングアルゴリズムを用いることで定数時間で行える．以上より，GET_FRONT_HALF全体は定数時間で実行できることがいえる． ■

Algorithm GET_FRONT_HALF

{ $\alpha[i]$, $\beta[i]$, $\gamma[i]$ はそれぞれ PE[0,i].a, PE[0,i].b, PE[0,i].c を表す ($0 \le i < M$)．α,βはソート済み．la, lb は出力の値 l_a, l_b をそれぞれ保持する．t1, t2 は作業用変数 }

1: $\beta[i]$ を$\gamma[M-i-1]$ へ置換 ($0 \le i < M$).
2: PE[0,$*$]: if a≥c then t1:=1 else t1:=0
3: PE[0,$*$]: if t1=0 then c:=a
4: PE[0,$*$]: if t1=0 then {N, S, EW}
else {N, S, E, W}
5: PE[0,0]: "1" → E
6: PE[0,$*$]: t2 ← W
7: PE[0,$*$]: {N, S, EW}
8: PE[0,$*$]: if t2=1 and t1=1 then id_y → E
9: PE[0,$*$]: la ← W
10: PE[$*$,$*$]: {NS, E, W}
11: PE[0,$*$]: la → S
12: PE[$*$,$*$]: la ← N
13: PE[$*$,$*$]: lb:=M−la
14: [2] のアルゴリズムで，γを定数時間でソート．

end of GET_FRONT_HALF

図 11: アルゴリズム GET_FRONT_HALF

補題 5.6 (2本のリストの merge) 以下で定義される問題は$M \times M$L-RM 上で$O(\frac{N}{M})$時間で解くことができる．ここで，$\alpha[i]$,$\beta[i]$,$\gamma[i]$ はそれぞれ
PE[0,i mod M].a[i div M],PE[0,i mod M].b[i div M], PE[0,i mod M].c[i div M] を表す．

入力: 長さNの2本のソート済みリストが，$\alpha[0,\ldots,N-1]$, $\beta[0,\ldots,N-1]$ にそれぞれ与えられる．
出力: $\alpha[0,\ldots,N-1]$ と $\beta[0,\ldots,N-1]$ をマージしてソートすることにより得られるリストが $\gamma[0,\ldots,2N-1]$ に格納される．

証明 上記の問題を解くアルゴリズム MERGE を図12に示す．簡単のため，$N \bmod M = 0$ とする．

MERGE では2の for ループの中身を1回繰り返すたびにγの要素を先頭からM個づつ生成していく．補題5.5により GET_FRONT_HALF は定数時間で行えることがいえる．また，詳細は省略するがその他の操作も定数時間で行うことができる．よって，MERGE に要する時間は for ループの繰り返しの回数のオーダー，すなわち$O(\frac{2N}{M})$=$O(\frac{N}{M})$であることがいえる． ■

Algorithm MERGE

{$\alpha[i]$,$\beta[i]$,$\gamma[j]$ はそれぞれ
PE[0,i mod M].a[i div M],
PE[0,i mod M].b[i div M],
PE[0,j mod M].c[j div M] を表す ($0 \le i \le N, 0 \le j < 2N$)．$\alpha,\beta$ はソート済み．また，PE[$*$,$*$].a[$\frac{N}{M}$], PE[$*$,$*$].b[$\frac{N}{M}$], PE[$*$,$*$].c[$\frac{N}{M}$] には番兵として $+\infty$ が入っているものとする．i, ia, ib, a′, b′ は作業用変数．$\alpha'[i]$, $\beta'[i]$ はそれぞれ PE[0,i].a′,PE[0,i].b′ を表す．}

1: ia=0; ib=0;
2: for i=0 to $\frac{2N}{M}-1$ do
case ia< N and ib< N:
GET_FRONT_HALF の入力条件を満たすように $\alpha[i_a,\ldots,i_a+M-1]$ と $\beta[i_b,\ldots,i_b+M-1]$ をそれぞれ置換したものを，$\alpha'[0,\ldots,M-1]$ と $\beta'[0,\ldots,M-1]$ へ格納する．次に α',β' に対して GET_FRONT_HALF を適用し，得られるリストを $\gamma[\mathrm{i}M,\ldots,(\mathrm{i}+1)M-1]$ へ格納する．
ia:=ia+l_a; ib:=ib+l_b;
case ia≥ N:
$\beta[\mathrm{ib},\ldots,\mathrm{ib}+M-1]$ を $\gamma[\mathrm{i}M,\ldots,(\mathrm{i}+1)M-1]$ へ格納する．ib:=ib+M;
case ib≥ N:
$\alpha[\mathrm{ia},\ldots,\mathrm{ia}+M-1]$ を $\gamma[\mathrm{i}M,\ldots,(\mathrm{i}+1)M-1]$ へ格納する．ia:=ia+M;

end of MERGE

図 12: アルゴリズム MERGE

補題 5.7 アルゴリズム *SORT2* において，Qの各列のソートは$O(\frac{N}{M})$ 時間で行うことができる．

証明 図13に SORT2 においてQの各列のソートの際に用いるアルゴリズム MERGESORT を与える．補題5.6が成り立つことから，簡単な計算を行うことによって，$M \times M$L-RM 上でのN個の値のソーティング問題を，MERGESORT を用いて$O(\frac{N}{M}\log\frac{N}{M})$時間で解けることがいえる．

Qの各列のソートは，各サブメッシュが自分のサブメッシュ内に保持しているQの要素をソートすることにより行う．Qのj列目の要素$Q[i,j](0 \le i < r)$は$\mathrm{PE}_j[0, i \bmod K].\mathrm{q}[i \operatorname{div} K]$に保持されているので，これらの値に対してMERGESORTを実行することにより，Qのj列目をソートすることができる．Qの一列の要素数はr，サブメッシュの大きさは$K \times K$であるから，このサブメッシュ上でMERGESORTを実行するのに必要な時間計算量は(c_1は定数)

$$c_1 \frac{r}{K} \log \frac{r}{K} = c_1 \frac{r\sqrt{s}}{M} \log \frac{r\sqrt{s}}{M}$$
$$= c_1 \frac{rs}{M\sqrt{s}} \log \frac{rs}{M\sqrt{s}} = c_1 \frac{N}{M\sqrt{s}} \log \frac{N}{M\sqrt{s}}$$
$$< c_1 \frac{N}{M} \frac{\log N}{\sqrt{s}} = c_1 \frac{\sqrt{2}N}{M} \frac{\log N}{N^{\frac{1}{6}}}$$

ここで，任意の正整数Nに対して$\frac{\log N}{N^{\frac{1}{6}}} < c_2$となる定数$c_2$が存在する．よって，

$$c_1 \frac{\sqrt{2}N}{M} \frac{\log N}{N^{\frac{1}{6}}} < \sqrt{2} c_1 c_2 \frac{N}{M} = O(\tfrac{N}{M})$$

がいえる．よって補題が成り立つ． ■

Algorithm MERGESORT

{ $\alpha[i]$は$\mathrm{PE}[0, i \bmod M].\mathrm{a}[i \operatorname{div} M](0 \le i < N)$を表す }

if $N \le M$ then

[2]のアルゴリズムを用いて$\alpha[0, \ldots, N-1]$を定数時間でソート．

else

$\alpha[0, \ldots, \lfloor \frac{N}{2} \rfloor - 1]$にMERGESORTを適用．

$\alpha[\lfloor \frac{N}{2} \rfloor, \ldots, N-1]$にMERGESORTを適用．

$\alpha[0, \ldots, \lfloor \frac{N}{2} \rfloor - 1]$と$\alpha[\lfloor \frac{N}{2} \rfloor, \ldots, N-1]$に対して，MERGEを適用．

end of MERGESORT

図 13: アルゴリズム MERGESORT

定理 5.2 $N < 2(M^2-1)^2M^2$とする．このとき，アルゴリズムSORT2を用いてソーティング問題を解くことができる．また，この時に必要な時間は$O(\frac{N}{M})$である．

証明 SORT2の Phase 0で，αに保持されている値は$Q[i,j]$=$Q_{i,j}$となるようにQへ格納される．以降，Phase 1から Phase 4まででQを Column sort する．Phase 4の終了時点でQに保持されている値は column major order にソートされている．最後に Phase 5で，$Q[i,j]$の値を$\alpha[jr+i]$に格納する．以上より，最初にαに与えられた値はソートされることがいえる．

補題 5.3より Phase 0，Phase 5にかかる時間はいずれも$O(\frac{N}{M})$である．また，補題 5.4，補題 5.7より Phase 1から Phase 4にかかる時間もいずれも$O(\frac{N}{M})$であることがいえる．したがって SORT2 全体の時間計算量は$O(\frac{N}{M})$であることがいえる． ■

6 まとめ

N個の値を$M \times M$L-RMでソートするアルゴリズムを示した．その時間計算量は，$N \ge 2(M^2-1)^2M^2$のときは$O(\frac{N}{M} + \frac{N}{M^2} \log \frac{N}{M^2})$，$M < N < 2(M^2-1)^2M^2$のときは$O(\frac{N}{M})$である．また，$N \ge 2(M^2-1)^2M^2$のときに用いるアルゴリズム SORT1では L-RM の能力を必要としないため，SORT1は HV-RM や各行各列にブロードキャストバスのついたメッシュ結合計算機上でも実行できる．

参考文献

[1] M. Maresca. Connection Autonomy in SIMD Computers: A VLSI Implimentation. *J. of Parallel Distributed Computing*, 7(2):302–320, October 1989.

[2] M. Nigam and S. Sahni. Sorting n Numbers on $n \times n$ Reconfigurable Meshes with Buses. *J. of Parallel Distributed Computing*, 23(1):37–48, October 1994.

[3] J. Jang, M. Nigam, V. K. Prasanna, and S. Sahni. Constant Time Algorithmes for Computational Geometry on the Reconfigurable Mesh. *IEEE Trans. Parallel and Distributed Systems*, 8(1):1–2, January 1997.

[4] M. Nigam, V. K. Prasanna, and S. Sahni. Computational Geometry on a Reconfigurable Mesh. In *Proc. 8th Int'l Parallel Processing Symposium*, pages 86–93, April 1994.

[5] Y. Ben-Asher, D. Gordon, and A. Schuster. Efficient Self-Simulation Algorithms for Reconfigurable Arrays. *J. of Parallel Distributed Computing*, 30(1):1–22, October 1995.

[6] T. Leighton. Tight Bounds on the Complexity of Parallel Sorting. *IEEE Trans. Comput.*, C-34(4):344–354, April 1985.

[7] 笹田，松前，都倉. 固定サイズの再構成メッシュ上で任意個の値をソートするアルゴリズム. 電子情報通信学会技術研究報告, COMP97-76:1–12, December 1997.