1S−8

# データパラレル超並列計算機における適応型ソートアルゴリズム

岡田 英明　喜連川 優　高木 幹雄
東京大学生産技術研究所

## 1 はじめに

ソーティングネットワークを初めとして、並列ソーティングアルゴリズムは多数提案されており、最近では超並列計算機上への実装についても報告されるようになった。しかし逐次型と違い、並列ソーティングの場合は比較計算量やメモリアクセスの他に、ネットワークを介したデータの移動を考慮しなくてはならなくなるため、プロセッサ数、データ数によって最適なアルゴリズムは異なる。異なるアーキテクチャを持つ計算機ではなおさらのことである。そこで本論文ではデータパラレル超並列計算機におけるデータ数、プロセッサ数に適応したソーティングについて述べるとともにアーキテクチャに応じたソーティングについて考察する。

今回使用したのは、MasPar MP-1 というSIMD型の超並列計算機である。MP-1 は4bit の PE を2次元格子上に配置し、隣接する8方向の PE と通信が可能である X-net と、多段クロスバ結合による汎用通信網ルータを備えている。PE 数は1K である。

ソート対象は32bit 整数のキーのみとし、以下の説明においてPをPE数、Nをデータ数とする。

また、グラフ中の処理時間はPEあたりのデータ数で正規化している。

## 2 実装したアルゴリズムとその特徴

これまでに、既存のアルゴリズムである Bitonic Sort、Radix Sort を実装し[1]、バブルソートの改良を提案、実装した。まず、これらのアルゴリズムの特徴について述べる。

### 2.1 Bitonic Sort

ハイパーキューブ網などに適した固定間通信を用いたマージ型のソートである。MP-1 はメッシュ結合なのでハイパーキューブの埋め込みを工夫して、通信時間を減らして高速化を図っているが、ソート処理時間が $\Theta(N \lg^2 N)$ に比例するので、データ数が多いときには適さない。

### 2.2 Radix Sort

Radix Sort は、キーの下位の $b$ ビットブロックの値からPE内のローカルなランクを求め、それをスキャン加算してグローバルなランクにし、ランクに応じたPEへデータを転送することを次第に上位のビットに対して繰り返すことでソートを行なう。今回の場合は $(32/b)$ 回繰り返すことになる。$b$ を大きくするとスキャンの回数が $2^b$ に比例して大きくなるが、繰り返し回数は $b$ に反比例するのでデータ転送の回数は抑えられる。

スキャン回数がデータ数に依存しないため、PEあたりのデータ数が大きくなるほど効率が上がるが、そのときにはルータによるデータ転送時間が処理時間の大部分を占めてしまうという欠点を持つ。

### 2.3 サンプリングを用いたバブルソート

並列バブルソート（奇偶交換ソート）は、各PEで局所的にデータをソートした後、隣合うPE間でソート列のマージ、スプリットを行なうステップを繰り返してソートを行なうが、最悪の場合、(PE数−1) ステップ必要になるので高、超並列機には適さない。

しかし、ある程度ソートされた状態に近ければ、繰り返しの回数が少なくてすむ。そこで、サンプルソートの考えを用いて、このある程度ソートされた状態を作ることにより、並列バブルソートを高速化することを提案する。そのアルゴリズムは以下のようになっている。

1. PEあたり $s$ 個のキーを取り出して、例えば Bitonic Sort のように比較的小数データのソートに適した並列ソートを行なう。
2. その結果より $d-1$ 個のキーを分割キーとして選ぶ。
3. 分割キーにより分けられるグループに含まれるデータ数を Radix Sort で行なったようにスキャンを用いて求め、全体でのランクを決定する。
4. データをランクに応じたプロセッサに転送する。

この後、並列バブルソートを行なう。先のキー転送により、ある程度ソートされた状態になるので、並列バブルソートのステップ数は少なくてすむ。

サンプル数 $s$ は実験した範囲では $s = 16$ で十分であり、サンプルデータのソートにかかる時間は小さい。スキャンの回数は分割数 $d$ に比例し、バブルソートのステップ数は反比例するが、データ数を変化させた時の処理時間を分割数毎に表すと図1のようになり、分割数が1024のときにスキャンにかかる時間のため少ない分割数の時よりも遅くなっていることがわかる。

PEあたりのデータ数が多くなるほど効率が良くなる特徴は Radix Sort に似ているがルータへの依存度が小さく、逆に計算量は増えているところが異なっている。

---

Adaptive Sorting Algorithm on Massively Data Parallel Computer
H.Okada, M.Kitsuregawa, M.Takagi
Institute of Industrial Science, University of Tokyo

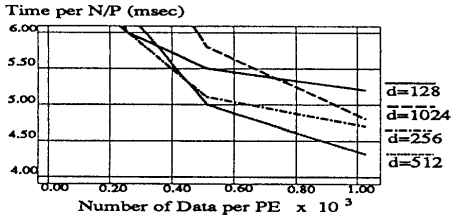


図 1: 改良バブルソートの分割数およびデータ数と処理時間の関係

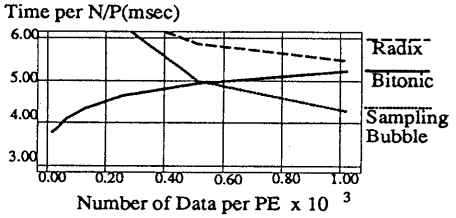


図 2: 3 アルゴリズムのデータ数と処理時間の関係

## 3 データ数、PE 数への対応

以上の 3 つのアルゴリズムについて、PE あたりのデータ数に対するソート処理時間を表すと図 2のようになる。ここで Radix Sort のパラメータは $b = 8$、サンプリングを用いたバブルソートのパラメータは $s = 16, d = 512$ としている。これより、PE あたりのデータ数が 512 以下では Bitonic Sort、それ以上ではサンプリングを用いたバブルソートを用いればよいということになる。

各アルゴリズムの処理時間の内わけおよび MP-1 における およそのコストは表 1のようになっている。表中の H.C はハイパーキューブ軸に沿った通信 (X-net で代用) を表す。

これより、PE 数 ($P$) の変化への対応について考える。例えば MP-1 の場合最大構成時の PE 数は 16K であるが、この時にはバブルソートの分割数を PE 数に比例して増やせば、PE あたりのデータ数で正規化すれば 1K PE のときと大きな違いはないように思える。しかし、それはコストが一定であるとの仮定の下であるのでメッシュ結合で代用しているハイパーキューブ通信やスキャンのコストは PE 数の平方根に比例して大きくなり、ルータ通信コストもそれほどではないが変化することを考慮しなくてはならない。これを考慮した時の予想される 16K PE 時の処理時間は図 3のようになると思われる。すなわち、アルゴリズム選択のしきい値が PE あたりのデータ数 256 程度となる。

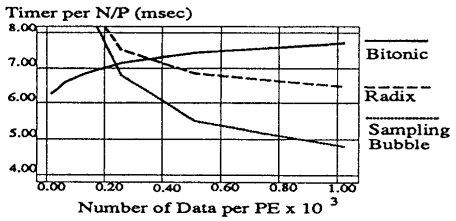


図 3: 16K PE 時のデータ数と処理時間の関係

## 4 異なるアーキテクチャへの対応

ここまでは、MP-1 という計算機を想定していたが、超並列計算機のアーキテクチャはさまざまであり、はじめに述べたように計算機毎に最適なアルゴリズムが異なる可能性が大きい。そこで、異なるアーキテクチャへの対応について考察する。

まず、コネクションマシン CM-2 のように似たようなアーキテクチャを持つものについては表 1で示したコストがわかれば良いものと思われる。

しかし、例えば NCUBE,iPSC などに代表される MIMD 型の計算機についてはアーキテクチャの違いが大きく予想は難しいと思われ、今回あげた 3 つの中に最適なアルゴリズムがあるとは言い切れない。しかし、これらをデータパラレル超並列機として考えると、スキャン命令などをソフトウェア実装してやれば、表 1を用いて概算することが可能となり、ソート時間の上界を与えることができる。ただし、データの転送単位によってその効率が大きくことなることなど考慮すべき点が多々あると思われる。

## 5 おわりに

データパラレル超並列計算機上のソーティングについてアルゴリズムの改良を行なうとともに、いくつかのアルゴリズムを用いて、データ数、PE 数に応じたアルゴリズムを選択して、最適なものに近付けることについて述べた。また、異なるアーキテクチャへの対応について考察した。実機での実験や、より正確な時間概算を導出することで異なるアーキテクチャの計算機への対応を検討していく。

## 参考文献

[1] 岡田、喜連川、高木、“SIMD 型超並列計算機におけるソーティングアルゴリズムの性能比較”, 情報処理学会第 44 回全国大会, 平成 4 年 3 月

表 1: 3 つのソート処理時間の内わけ

| アルゴリズム | H.C | X-net | ルータ | スキャン | 比較計算 |
|---|---|---|---|---|---|
| Bitonic | $(N/P)\lg^2 P$ | | | | $\Theta((N/P)\lg^2 N)$ |
| Radix | | | $(32/b)(N/P)$ | $(32/b)2^b$ | $\Theta((32/b)(N/P)) + \Theta((32/b)2^b)$ |
| Bubble | $s\lg^2 P$ | $4N/d$ | $N/P$ | $d$ | $\Theta(s\lg^2(sP)) + \Theta(N/d) + \Theta((N/P)\lg(N/P))$ |
| コスト | $15\mu s$ | $7\mu s$ | $1000\mu s$ | $320\mu s$ | $10\mu s$ |