4G−11 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

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

ソーティングは基本的なアルゴリズムの1つであり、大容量データを扱う超並列計算機にも不可欠な処理である。そこで、本稿ではBitonic Sort、Radix Sortの異なる2種類のソートアルゴリズムの実装を行ない、その性能評価について述べる。

対象とした超並列計算機は、SIMD型のMasPar MP-1である。MasParは、4ビットの要素プロセッサを2次元格子状に配置しており、8方向のPEに対する通信網(X-net)と多段クロスバを用いた汎用通信網を備えている。今回、使用したのはプロセッサ数1Kのものである。

2 Bitonic Sort

バイトニックソートは、2つのバイトニック列のマージを繰り返すことによりソートを行なう。長さ2^{d-1}の2つのバイトニック列をマージするには、2^{d-1}個の比較器を用いてd回の並列比較を行なう必要がある。したがって、N個のキーをソートするには$\lg N$回のマージが必要となるので、全体では$O(\lg^2 N)$回の並列比較を必要とする。マージの際のキーの比較交換が規則的であるので、SIMD型の超並列計算機に適したアルゴリズムであり、プロセッサの利用の点から見れば最適だが、逐次ソートの最適比較回数を$O(N\lg N)$と考えると比較回数そのものは最適ではない。

2.1 仮想化

プロセッサ数よりも多くのキーをソートするには1つのPEに複数のキーを割り当てなければならないので、何らかの仮想化手法が必要となるが、その方法として、PE内に仮想的なプロセッサを持たせる方法と、キーを単独のものとしてでなくソートされたキーの列とみなし、キーの比較交換ではなくソート列のマージスプリットをする方法がある。ここでは、前者をピュアバイトニック、後者をブロックバイトニックと呼ぶこととする。

キーの数をN、プロセッサ数をPとすると、ピュアバイトニックでは比較の回数は$O((N/P)\lg^2 N)$となる。ブロックバイトニックでは、はじめに例えば$O((N/P)\lg(N/P))$の逐次ソートを行ない、それから長さ(N/P)のソート列のマージスプリットを
$O(\lg^2 P)$回行なうので、$O((N/P)(\lg(N/P)+\lg^2 P))$回の比較が必要になる。したがって、$P$に対して$N$が十分大きいと、ブロックバイトニックの方がピュアバイトニックよりも比較回数は少なくて済む。例えばP=1Kでは、N/P=500程度で比較回数は同じになることになる。

Performance Comparison of Sorting Algorithms for SIMD type Massive Parallel Computer
H.Okada, M.Kitsuregawa, M.Takagi
Institute of Industrial Science, University of Tokyo

2.2 ハイパーキューブ結合網の実現

バイトニックソートはハイパーキューブ結合網に適しているが、MasParは2次元格子結合されているので、PEのインデックスをハイパーキューブ通信に適したものにすることが必要となる。その代表的なものとして、行主体インデックスとシャッフル行主体インデックスがある。

行主体ではi次元のハイパーキューブに沿った通信は、

- $i < \lg N\ /2$のとき 2^i East
- $i \geq \lg N\ /2$のとき $2^{i-lgN/2}$ South

となり、シャフル行主体では、

- iが偶数のとき $2^{i/2}$ East
- iが奇数のとき $2^{i/2}$ South

となる。バイトニックソートでは、下位次元のハイパーキューブにそった通信が多く用いられるのでシャッフル行主体の方が通信コストが小さくなる。

MasParのように隣接8PEと結合している場合は、

- iが0のとき 1 NE
- iが1のとき 1 SE
- iが2のとき 1 E
- iが奇数のとき $2^{i/2}$ NE
- iが偶数のとき $2^{(i-1)/2}$ SE

のように、隣接プロセッサ間で1次元よけいにとれるので通信コストをさらに減らすことができる。

通信コストを減らす方法としてさらに有効なのがハイパーキューブのマッピングを可変にすることである。比較するキーが各PEで1対1の関係にあるので、キーの半分を交換して、各PE内で比較交換を行うようにするのである。こうすると通信コストをおよそ半分にすることができる。

2.3 実装結果

以上のようにバイトニックソートをMasParに実装し、32ビット整数のキーを1M個ソートした時の行主体のブロックバイトニック、ピュアバイトニックおよびピュアバイトニックでシャッフル行主体、8方向リンクを用いたもの、ハイパーキューブ軸を可変にしたものについて、全体の処理時間、X-netによる通信時間、比較時間、その他の時間を表1に、キーの数を変化させた時の行主体のブロックバイトニック、ピュアバイトニック、およびピュアバイトニックでハイパーキューブ軸を可変にしたものの処理時間のようすを図1に示す。

通信コストについては予想通りの結果が現れているが、ブロックバイトニックとピュアバイトニックの比較ではブロックバイトニックが予想以上に遅いことがわかる。これは、比較の時間および、その他のところに分類されている時間が非常に大き

いものとなっているからである。

表 1: バイトニックソートの各処理時間 time/(N/P)(msec)

	全体	X-net	比較	他
ブロック・行主体	11.6	4.51	2.82	4.27
ピュア・行主体	7.61	4.12	1.86	1.63
ピュア・シャッフル	7.19	3.61	2.32	1.26
ピュア・8方向	6.91	3.32	2.28	1.31
ピュア・軸可変	6.06	2.42	1.97	1.67

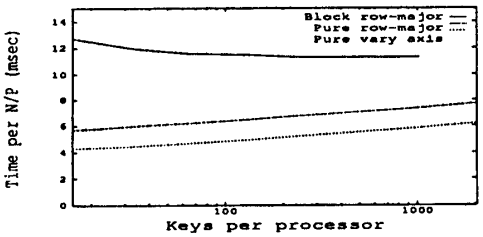

図 1: キーの数と処理時間の関係 time/(N/P)(msec)

3 Radix Sort

ここで採用する基数ソートは計数型のソートであり、バイトニックソートのように比較に基づくものではなく、安定であるという特徴を持つ。

基数を 2^b とすると 1 つの基数ソートのステップは次のようになる。

1. 各 PE で下位 b ビットのブロックのランクを求める。
2. スキャン加算することにより、全体でのランクを求める。
3. キーをランクに応じた PE へ転送する。

これを b ビット毎に上位のビットに対して繰り返す。したがってキーが 32 ビット整数の場合は $32/b$ ステップでソートが完了する。

3.1 基数の選択

このアルゴリズムの実行時間を見積ると、ルータ、スキャンにかかる時間を R,S、これらと比例係数を同じくするその他の時間を α,β とすれば、$(32/b)((N/P)(R+\alpha)+2^b(S+\beta))$ となる。

b が小さいとステップ数 $(32/b)$ が多くなるのでキー転送の回数も多くなることになるが、スキャンの回数は少なくなる。一方、b が大きいとステップ数が減り、キー転送の回数も少なくなるが、スキャンを行なう回数 (2^b) が多くなる。ただし、スキャンにかかる時間はキーの数には依存しないので、PE あたりのキーの数 (N/P) が大きい時にはスキャンを多く行なっても大きな影響を及ぼさないと思われる。

3.2 実装結果

基数を変えて 512K 個の 32 ビット整数キーのソートを行なった時の実行時間とルータ、スキャンに費やされた時間、およびその他の時間を表 2 に示す。また、b が 4,8 のときにキーの数を変えた時のソート時間を図 2 に示す。

表 2: 基数と処理時間の関係 time/(N/P)(msec)

基数	全体	ルータ	スキャン	その他
2^2	26.3	22.5	-	3.8
2^4	10.6	8.41	0.07	2.12
2^8	5.84	4.00	0.66	1.18

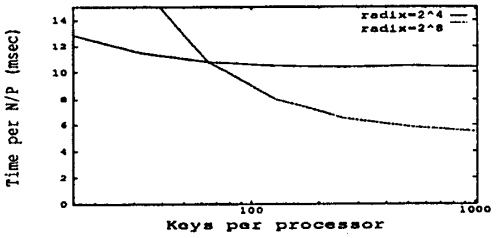

図 2: キーの数と処理時間の関係 time/(N/P)(msec)

表 2 より、スキャンよりもルータの通信時間のほうが支配的であることがわかる。この結果より 1 度のスキャン、ルータにかかる時間を求めるとルータの方がスキャンの 3 倍遅いことがわかる。したがって、キーの数がある程度以上になると基数を大きくした方が全体の処理時間は短くなる。

4 まとめ

SIMD 型超並列計算機である MarPar に Bitonic Sort,Radix Sort の 2 種のソートアルゴリズムを実装し、その性能を評価した。

バイトニックソートでは、間接アドレッシングを用いないことやハイパーキューブ結合を変更することによって処理時間を短縮できるが、キーの数が多い時には不適であるといえる。

基数ソートでは、ルータによる通信時間が大きいため、基数を大きくとることが有効であり、キーの数が増えるにしたがって漸近的に最適に近づき、バイトニックソートよりもやや高速にすることができた。

今後は、他の超並列計算機への実装やアーキテクチャにふさわしいアルゴリズムについて検討したい。

参考文献

[1] J. F. Prins, J. A. Smith, "Parallel Sorting of Large Arrays on the MasPar MP-1", The third symposium on the FRONTIERS OF MASSIVELY PARALLEL COMPUTATION, 1990.

[2] G. E. Blelloch, C. G. Plaxton, C. E. Leiserson, S. J. Smith, B. M. Maggs, M. Zagha, "A Comparison of Sorting Algorithms for the Connection Machine CM-2", Thinking Machines Corporation, 1991.

[3] Hussein M. Alnuweiri, V. K. Prasanna Kumar, "Optimal VLSI Sorting with Reduced Number of Processors", *IEEE Trans. Comput.*, vol.40, pp.105-110, Jan. 1991.

[4] Isaac D. Scherson, Sandeep Sen, "Parallel Sorting in Two-Dimensional VLSI Models of Computation", *IEEE Trans. Comput.*, vol.38, pp.238-249, Feb. 1989.