Vol. 26 No. 2 情報処理学会論文誌 Mar. 1985

マルチマイクロプロセッサによるソート/サーチ エンジンの試作†

相原玲二†† 阿江 忠††

コンピュータの技術的発達および普及に伴い，さまざまな規模のデータベース・システムが実用化されつつある．データベース・システムの高速化を図るにはその基本処理にあたるソート/サーチ処理時間の短縮が効果的であると予想され，この傾向は扱うデータ数が増加するほど顕著になる．これらの処理を高速に行う一つの方法として，並列処理による専用マシンを用いることが挙げられる．ソート専用マシンには多くの提案があるが，現在のマルチマイクロプロセッサでも容易に実用的なデータ数の処理が実現できるアーキテクチャの一つとして文献1)の方式がある．しかも，この方式は同じマシンをサーチにも使用できるという特長をもつ．本論文では，小規模データベース用のソート/サーチ専用マシンとして約1万件のデータのソート/サーチ用に試作したマシンの製作ならびに実験結果を報告する．

1. まえがき

計算機の応用分野は際限なく広がっているが，実用上のボトルネックの一つとして処理速度の問題がある．たとえば，超高速計算のためのスーパコンピュータなどの研究もその一つに挙げられる．一方，データベース・システムもますますその重要性が認識されつつあるが，処理速度の問題は実行上いろいろな制約を与えている．

われわれは，データベース・システムにおける基本処理にあたるソートとサーチに焦点を当て，ソート/サーチのための専用マシンを実用化のレベルで設計，製作し，実験を行った．

ソートに関しては古くから多くの研究がなされており，並列処理により $O(\log n)$ の速度が得られることは理論的にはよく知られている[3]．しかしながら，実用的にはコストとパフォーマンス両面に問題があり，すぐに実用化レベルのものを製作するわけにはいかない．コストの点では，1（あるいは2）データに一つのセル（比較器，レジスタおよび制御回路をもつ）を割り当てるのは，現在のシングルプロセッサの場合，1データが1メモリに割り当てられているのに比べ，基本単位がかなり大きくなる．定数倍のコスト増とはいえ，データ数が大きくなる実用化レベルでは簡単には許容しがたい．さらに，パフォーマンスの点では，セル間の通信が大きなネックになり，理論上はこの問題の評価はいまだ正当には行われていないが[4]，現実には数千，数万のセルが自由に通信することは不可能に近いと考えられている[5]．

一方，もう一つ別の観点に立てば，「$O(\log n)$ の速度のソートが本当に必要か？」という問題もある．すべてが半導体メモリでしかもバックアップ可能なものになれば，オンメモリですべてを記録することも夢ではないかもしれぬが，当面は，ディスク装置のような二次記憶装置にデータベースの本体が存在するし，将来も，このような形態が続くと考えるのが自然であろう．したがって，データがメモリ上に存在するところからのソートより，ディスクのような装置から直列に入出力するようなシステムでのソートのほうが実用的である（図1）．極論すれば，n 個のデータの入出力に要する時間に比べて極端に速いソートは必要なく，その時間と同程度の速度のソートで十分なことを意味している．$O(n)$ の速度のソートも，実現の方法をよく吟味し係数をある程度以下に抑えられれば十分実用になる．$O(n)$ のソートアルゴリズムは，VLSI に適するもの[6]などが提案されているが，本稿ではコストを現状のマルチマイクロプロセッサで評価してただちに実現可能なものをとり挙げた．さらに，サーチについても同様の観点から実現的なものを採用し，マルチマイクロプロセッサ上での実現を行った．

2. ソート/サーチエンジン

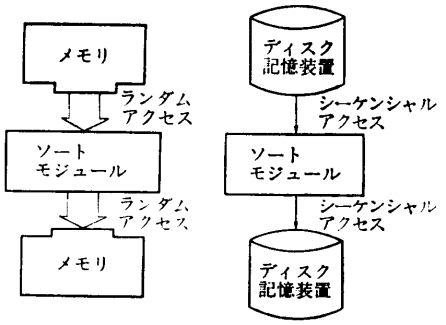
過去の多くの研究のなかから，並列処理を導入したソートアルゴリズムの候補として，

（1） パイプライン木状ソート[1]（図2）

（2） パイプラインマージソート[2]（図3）

† An Implementation of Sort/search Engine on a Multimicroprocessor by REIJI AIBARA and TADASHI AE (Cluster II (Electrical and Industrial Engineering), Faculty of Engineering, Hiroshima University).

†† 広島大学工学部第二類（電気系）



(a) メモリ上のデータのソート
Sorting on memory.

(b) ディスク上のデータのソート
Sorting on disk.

図 1 ソートにおけるデータの流れ
Fig. 1 Data flow on two sorting modules.

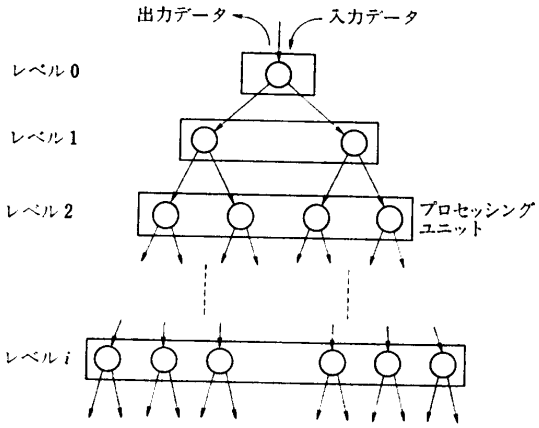


図 2 パイプライン木状ソートモジュール
Fig. 2 A pipeline tree sorting module.

が挙げられる．いずれも，パイプライン処理の導入により $O(n)$ の速度でソートがなされ，データ数 n に対し $\log n$ 個のプロセッサを使用する．

(1)の方法は，逐次処理でのヒープソートに用いられるヒープ木と同様のデータ構造を利用してソートを行うものであり，(2)は，ファイルのソートによく用いられるマージソートを並列処理で行う．いずれの方法もオーダは同じであるが，処理速度においてそれぞれ異なる特徴をもつ．1データあたりの単位処理時間を1とし，n 個のデータをソートする場合，全処理時間（データの入力を始めてからソートされた結果の出力が終了するまで）は，それぞれ

(1) $2n$

(2) $2n+\log_k n-1$

となる．ただし(2)では，k-way マージソートを行う場合を示す．つまり，(1)ではデータの入力後ただちに結果の出力が開始するが，(2)ではデータの入力

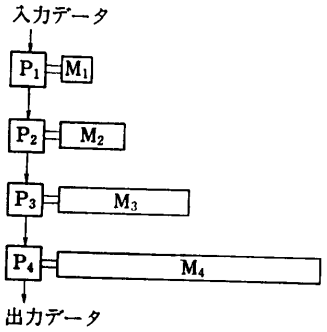


図 3 パイプラインマージソートモジュール
Fig. 3 A pipeline merge sorting module.
P_i: プロセッシングユニット，M_i: メモリユニット

の $\log_k n-1$ 時間後出力を開始する．したがって，1データ群のソート時間については(1)のほうが有利である．一方，多くのデータ群を連続的にソートする場合を考えると，(2)のほうはデータ入力と結果出力とがオーバラップして行えるためやや有利となる．n 個のデータから成るデータ群を連続的にソートする場合，1データ群あたりのソート時間はそれぞれ，

(1) $2n$

(2) $n+\log_k n$

となる．

以上の議論より，ソート時間については，(1)(2)の間に微妙な違いはあるものの，さほどの優劣があるわけではない．しかし，

(i) マルチマイクロプロセッサへの適応性，および

(ii) サーチとの互換性

を考え，本稿では(1)の方式を採用した．その理由は以下のとおりである．

(i)のうち，おもに問題となるのは1プロセッサが管理するデータ数である．提案されている2方式では，いずれもプロセッサ $i(i=0,1,2,\cdots)$ が 2^i((2)の方式で $k=2$ とする）個のデータを格納することになるが，(1)の方式は若干の変更によりプロセッサ $i(i\geq s)$ のデータ数を $2^s(s=0,1,2,\cdots)$ にすることができる[7]（**図 4**）．しかも，この変更によりソート時間が変わることはない．この変更により，プロセッサ $i(i\geq s)$ にすべて同一のものを用いることが可能となり，1プロセッサがさほど大きなメモリをもたないようなマルチマイクロプロセッサには(1)の方式が適している．さらに，(1)の方式は文献1)に示されるパイプライン・サーチと同様の構成をしており，マルチマイクロプロセッサ上へソフトウェアにより実現する

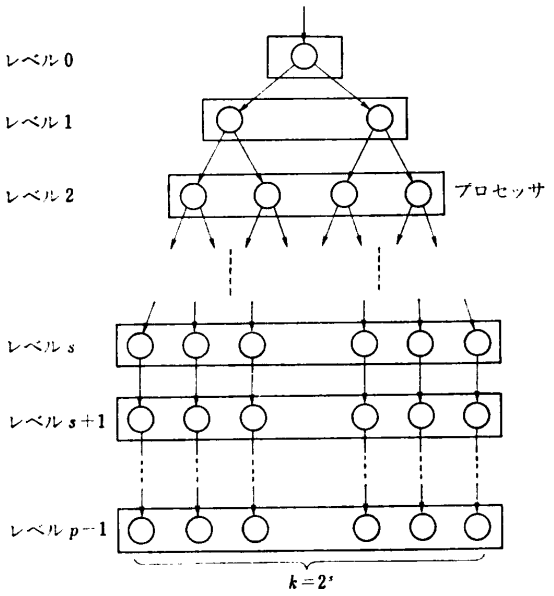


図 4 ソート/サーチ エンジンのデータ構造
Fig. 4 Data structure of the sort/search engine.

場合同一ハードウェアで済む．本稿ではこの(1)の方式を採用し，以下，この方式を提唱している文献1)の呼称のソート/サーチエンジンを用いる．

図4の変更により，処理されるデータの数を n (ソートにおいてはソートデータ数，サーチにおいては格納されサーチされるデータ数)，1プロセッサに格納されるデータ数の最大値を $k(=2^s)$ とすると，必要となるプロセッサ数 k は，

$$k=\begin{cases}\lceil(n+1)/k\rceil-1+\log_2 k & (n\geq 2k)\\ \lceil\log_2(n+1)\rceil\end{cases} \qquad (1)$$

により求まる．

ソート/サーチエンジンの動作はそれぞれ二つのフェーズから成る．ソートにおいては，ホストコンピュータから送られてくる被ソートデータを各プロセッサで比較を行いながら次々と図4の構造に格納してゆく入力フェーズ，入力が終了した後，各プロセッサでデータ比較を行いながら次々とデータを出力してゆく出力フェーズである．サーチにおいては，ホストコンピュータから送られてくるソートされた被サーチデータをあらかじめ決められた順序で図4の構造に格納する格納フェーズ，格納が終了した後，ホストコンピュータから送られてくるサーチデータを各プロセッサで比較して一致するデータを捜し，一致すればそのデータの識別子を得る検索フェーズである．いずれのフェーズもパイプライン動作を行い，ソートの入力フェーズ，サーチの格納フェーズ，検索フェーズはいずれも**図5**のタイムチャートに従う．また，ソートの出力フェーズはデータが双方向に転送されやや複雑になり**図6**のように動作する．ただし，図5，図6とも，ソート/サーチエンジンをマルチマイクロプロセッサ上で動作させ，かつ，各プロセッサで入出力処理と内部の処理は並行に行われない場合を示している．なお，本稿ではソート/サーチエンジンの各プロセッサの動作の詳細は省略する．

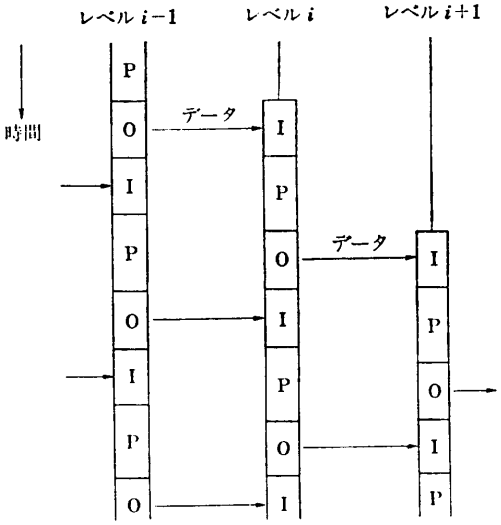


図 5 パイプライン処理のタイムチャート(1)
Fig. 5 Time chart of pipeline processing (1).
I：データ入力，P：内部処理，O：データ出力

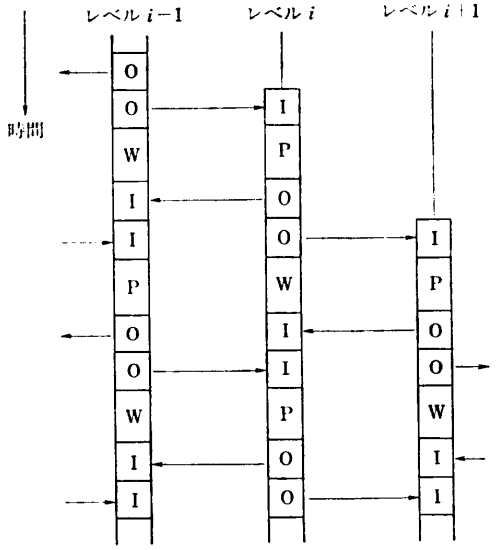


図 6 パイプライン処理のタイムチャート(2)
Fig. 6 Time chart of pipeline processing (2).
I：データ入力，P：内部処理，O：データ出力，W：待ち

3. ハードウェア

ソート/サーチエンジンをマルチマイクロプロセッサにより実現する場合のハードウェア構成，性能の予測，さらに問題点等について考察する．

3.1 利点と欠点

ハードウェア構成は非常に簡潔である．**図7**のようにプロセッサを一次元状に並べ，隣り合うプロセッサ間で通信が可能となるように何らかの結合があればよい．汎用のプロセッサや入出力素子を用いることの利点は

（1） 各プロセッサの構成が簡潔，

（2） 一次元状の結合のため実装が容易，

（3） 汎用品利用のため低価格で実現が可能，

などである．一方問題点は，内部処理に関しては，

（4） 使用するプロセッサによりデータの処理単位が限られる（8～32ビット程度），

（5） 専用でないため余分な操作(データ転送など)が必要となる，

（6） データ入力，データ処理，格納アドレス計算などを同時に（並列に）処理できない，

（7） 不要な制御回路，演算回路，インストラクションなどをもっている．

また，プロセッサ間通信に関しては，

（8） 通信速度が使用プロセッサ，使用素子により限られる，

（9） 通信プロトコルのために時間を必要とする，

などが挙げられる．

3.2 性能予測

ソート/サーチエンジンにおける各プロセッサの処理はデータ比較，データ交換，プロセッサ間のデータ移動がおもなものであり，それらは図5，図6のタイムチャートに従うことから，実現した場合の性能予測を行う．データの比較，交換などプロセッサ内部の処理に必要なインストラクション数を1データあたりm（インストラクション/データ），プロセッサの処理速度をS（インストラクション/秒），プロセッサ間を移動するデータの長さを1データあたりd（バイト/データ），プロセッサ間の通信速度をB（バイト/秒）とすると，1データあたりの内部処理時間はm/S(秒/データ)，1データあたりの通信時間は$2d/B$(秒/データ）となる．n個のデータを処理する場合，ソート/サーチ各フェーズの処理時間t(秒)は，

$$t=(m/S+2d/B)n \qquad (2)$$

で表すことができる．

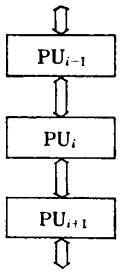


図7 ソート/サーチ エンジンのためのプロセッサ間結合
Fig. 7 Connection between processors for sort/search engine.
PU: プロセッサおよびメモリ

表1 マイクロプロセッサの実行速度比較
Table 1 Comparing execution time among microprocessors.

処　理	Z-80 (4 MHz) (μs)	Z 8000 (6 MHz) (μs)*	MC 68000 (8 MHz) (μs)*
32ビットデータ比較	40.5	2.33	1.75
32ビットデータ交換	57.5	4.50	3.75

* 比較，交換ともレジスタ・メモリ間の処理とする．

例として，1件6バイト（うち4バイトがキーデータ，2バイトが識別子）のデータ10,000件のソートおよび10,000件のデータに対する10,000件のデータサーチをZilog Z-80 (4 MHz クロック）で実現する場合について求める．ソート入力フェーズの内部処理は1データあたり1回の比較と1/2の確率で交換が行われるとして，**表1**より

$$m/S=1\times 40.5+1/2\times 28.7=69(\mu\text{ 秒})$$

となる．また，プロセッサ間通信はZ-80の周辺素子のパラレルポートを用いるとすると，そのときのデータ転送速度は$B=25$（kバイト/秒）（実測値）より，

$$2d/B=2\times 6/(25\times 10^3)=480(\mu\text{ 秒})$$

となる．これらより，

$$t=(69+480)\times 10,000=5.49(\text{秒})$$

と求まる．同様にして他のフェーズについても求める．

ソート出力フェーズ：$t=7.09$(秒)

サーチ格納フェーズ：$t=4.80$(秒)

サーチ検索フェーズ：$t=5.49$(秒)

3.3 性能の向上

Z-80の予測例では，各フェーズとも全体の処理時間に占める通信時間の割合が大きく，約87%以上であり，処理速度の向上には通信速度の向上が効果的で

あることが予測される．たとえば，2ポートの共有メモリを用いる結合[10]とすれば，前例と同じく Z-80 を用いても，B=130（kバイト/秒）となる．ソート入力フェーズの場合，$2d/B$=92（μ 秒），t=1.6(秒) となって前例の 4.4 倍となる．

さらに，同様のアーキテクチャでプロセッサを Motorola MC 68000 にした場合，ソート入力フェーズは，表1などから

$m/S=1\times1.75+1/2\times3.75$
$=3.6(\mu$ 秒)
$2d/B=12(\mu$ 秒)
$t=0.16$(秒)

となる．ただし，この場合も共有メモリ結合とする．

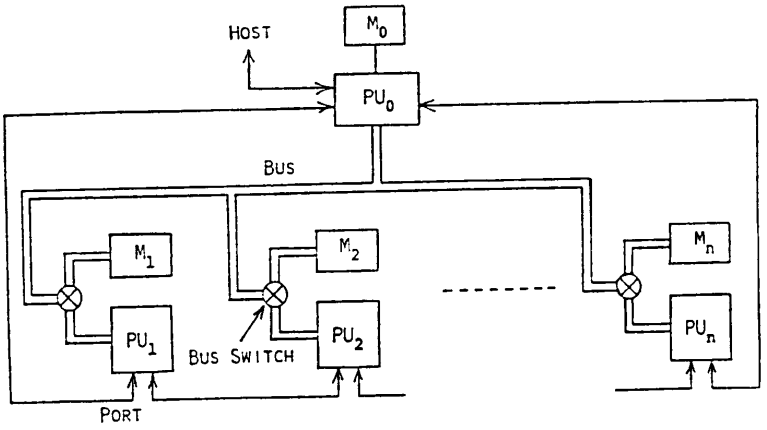


図 8 マルチマイクロプロセッサ UNIP の構成
Fig. 8 Configuration of multimicroprocessor UNIP.
PU_i：プロセッサ，M_i：メモリ（i=0 マスタ，$1\le i\le n$：スレーブ）

4. ソート/サーチ実験のためのマルチマイクロプロセッサ

今回ソート/サーチの実験に用いたマルチマイクロプロセッサ UNIP[8] の構成を**図 8** に示す．UNIP はマスタ1個，スレーブ 31 個のプロセッサからなるマスタスレーブ方式のマルチマイクロプロセッサで，すべて Zilog Z-80 A (4 MHz クロック) を使用している．また，ホストコンピュータに接続されて動作する付加プロセッサであり，ホストとマスタがパラレルポートにより接続される．UNIP は可能な限り簡潔な構成をとることを設計の第1条件としており，プロセッサ間は単一バス結合とパラレルポート結合を併用している．その特徴を列挙すると，

（1） 簡潔な構成，実装が容易かつ小型
(431 mm×222 mm×254 mm：電源別)，
（2） プロセッサ数の拡張が容易，
（3） バスとポートの併用により融通性がある[9]，

などの長所があるが，反面，

（4） プロセッサ間通信がバスに集中しがちとなる，
（5） マスタへの負担が大きくなる場合がある，

などの問題がある．

ソート/サーチエンジンの実現においては，ソート/サーチいずれも実行中のプロセッサ間通信はポートを用いているため(4)の問題は起きない．また，今回の実験において，マスタはホストとスレーブ間の通信のみを行っているため(5)の負担の問題もない．

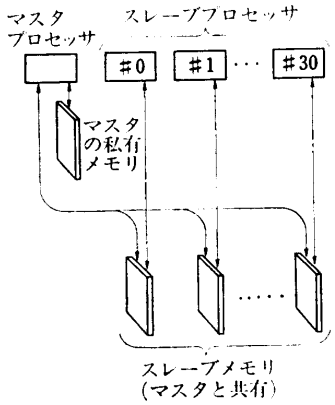


図 9 UNIP におけるメモリアクセス
Fig. 9 Memory access scheme on UNIP.

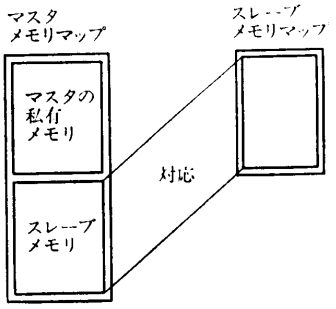


図 10 UNIP のメモリマップ
Fig. 10 Memory mapping on UNIP.

なお，UNIP でのメモリアクセスは，マスタからスレーブ方向のみ許され（**図 9**），

（1） ブロードキャスト書込み，
（2） 個別アクセス

の2種が用意されている．(1)はすべてのスレーブメ

モリ同一内容を書き込むもので，(2)は各スレーブメモリをバンク切換えにより一つだけ選択して読み書きを行う．マスタから見たメモリマップを**図10**に示す．この，バスを使う通信は，ソート/サーチ実験においては，起動時の各スレーブへのプログラムロードおよび初期化の際用いられる．

5. ソート/サーチ実験

実験は，ホストに8ビットマイクロコンピュータ(Z-80 A, 4 MHz クロック）を用い，増設したパラレルポートへ UNIP を接続して行った．実現したソート/サーチエンジンの仕様を以下に示す．

- 1データ：キーデータ　32ビット（4バイト）
  識　別　子　16ビット（2バイト）

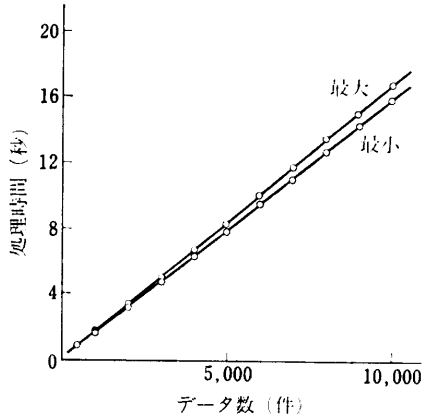


図11 ソートエンジンによるソート時間
Fig. 11 Sorting time by sort engine.
キーデータ：4バイト，識別子：2バイト

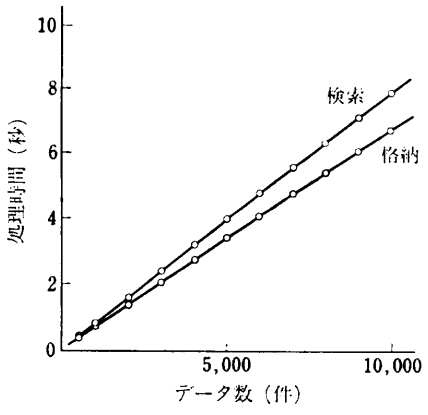


図12 サーチエンジンによるサーチ時間
Fig. 12 Searching time by search engine.
キーデータ：4バイト
識　別　子：2バイト　} 格納
キーデータ：4バイト(入力)
識　別　子：2バイト(出力)　} 検索

- 扱いうるデータ数：11,775件
- プロセッサ当りの最大データ数：512件
- 使用プロセッサ数：31（マスタを除く）
- プログラム記述言語：アセンブリ言語

ソート/サーチそれぞれにつき1万件までの処理を行ったときの処理時間を**図11, 12**に示す．このとき，ソート/サーチエンジン自体の性能を求めるため，ホスト上のプログラムは可能な限り速く動作するよう工夫してある．図中最大，最小とあるのは，処理プログラムの関係上データの値（シーケンス）により処理時間が異なるため，その最大となるであろうデータ，最小となるであろうデータをそれぞれ入力して測定した結果である．つまり，上限および下限である．さらに，**表2**には各フェーズにおけるホストコンピュータとの間の平均データ転送速度を示す．最後に，試作システムにおける処理時間の内訳を**表3**に示す．この表の値はレベルおよび入力されるデータ値により異なるが，平均値または代表値で示す．

これらの結果と3.2節での予測を比較するとかなり近いことがわかる．予測と結果の差のうち，ソート出力フェーズは双方向通信のための待時間の影響，サーチ格納フェーズは予測の際内部処理時間を0としたが実際にはいくらかは処理があることの影響，サーチ検索フェーズはマスタプロセッサがホストからのデータを受けてスレーブに送りながら，同時にスレーブからの結果をホストへ送っていることの影響などが考えられる．

表2 ソート/サーチエンジンの平均データ転送速度
Table 2 Average data transfer rate of sort/search engine.

処理フェーズ		平均データ転送速度（キロバイト/秒）
ソート	入力フェーズ	8.9
	出力フェーズ	6.3
サーチ	格納フェーズ	9.0
	検索フェーズ	7.7

表3 ソート/サーチ時間の内訳
Table 3 Sortirg/searching time of each part.

処理フェーズ		本質的に必要な処理(%)	データ転送(%)	アドレス計算その他(%)
ソート	入力フェーズ	13.9	62.5	23.6
	出力フェーズ	10.5	84.4	5.1
サーチ	検索フェーズ	12.5	86.2	1.3

6. むすび

本論文では，ソート/サーチエンジンを，汎用マイクロプロセッサからなるマルチプロセッサ上に実現することを考察し，試作機での実験結果を報告した．試作機ではプロセッサ間結合方式やメモリ容量等に制約があったため，プロセッサ台数の割に処理速度や処理可能なデータ数などがそれほどよくない．しかし，基本処理の処理時間から予測した全処理時間は実行結果に近く，マイクロプロセッサ数十台でのパイプライン処理がうまく動作することが実験的に確認できた．

結果を8ビットシングルプロセッサ上でのソートと比較すると，4バイトデータ10,000件で約14.2秒という実験結果を得ている．ただし，これはデータの入出力時間は含まれておらず，また識別子もないので一概に比較することはできないが，試作機のほうが2～3倍程度の速度である．シングルプロセッサはクイックソートを用いており，データ件数がさらに増えればその比率は大きくなってゆくと予想できる．

試作機での平均データ処理量は10kバイト/秒程度であるが，プロセッサ間通信の影響は大きく，結合方式の変更により4倍程度の高速化が望める．また，16ビットマイクロプロセッサを用いることにより，さらに1桁程度の高速化が期待される．プロセッサ1個にかかる費用は8ビットに比べ2～5倍程度であるので16ビットのほうがパフォーマンス/コストは2～5倍といえる．

謝辞　本試作システムの開発にあたって多大なご協力をいただいた広島大学工学部計算機工学研究室の飯田　優（現　富士通(株)），岡田高幸（現　日本アイ・ビー・エム(株)），森福　茂（現　沖電気(株)）各氏に深く感謝する．

参考文献

1) Tanaka, Y. et al.: *Pipeline Searching and Sorting Modules as Components of a Data Flow Database Computer*, in S. H. Lavington (ed.): *Information Processing* 80, pp. 427-432, North-Holland Pub. Co. (1980).
2) 喜連川優他：パイプラインマージソータの構成，信学技報，EC 82-32 (1982).
3) Batcher, K. E.: Sorting Networks and Their Applications, Spring Joint Computer Conference, AFIPS, Vol. 32, pp. 307-314 (1968).
4) Hirschberg, D. S.: Fast Parallel Sorting Algorithms, *Comm. ACM*, Vol. 21, No. 8, pp. 657-661 (1978).
5) Jones, A. K. and Schwarz, P.: Experience Using Multiprocessor Systems—A Status Report, *ACM Comput. Surv.*, Vol. 12, No. 2, pp. 121-165 (1980).
6) 安浦寛人，高木直史：並列計数法による高速ソーティング回路，電子通信学会論文誌，J 65-D, pp. 179-186 (1982).
7) Ae, T. and Aibara, R.: Experimentation and Analysis of Multiprocessor Systems, IEEE Proc. Real-Time Systems Symp., pp. 69-80 (1982).
8) 相原玲二，阿江　忠：並列パイプラインプロセッサ UNIP の応用，信学技報，EC 82-31 (1982).
9) 相原玲二他：多数個プロセッサ・アレイーバス・アレイ，信学技報，EC 82-64 (1982).
10) 阿江，高橋，松本：共有メモリ結合によるマルチマイクロプロセッサの並列動作について，信学論(D)，J 65-D, pp. 322-329 (1982).

(昭和59年4月2日受付)
(昭和59年10月18日採録)