マージ型ベクトル演算機構を用いた非数値処理の高速化方式

鳥居俊一[†] 小島啓二[††] 金田泰[††]
坂田明治[††] 高橋政美[†††]

本論文では，ベクトルアーキテクチャをマージ演算に拡張した内蔵データベースプロセッサ(IDP)を用いた非数値処理高速化の課題と実例を示す．マージ型ベクトル演算は，各オペランドが独立にインデックスを持つ点に特長がある．そのため，主記憶上にオペランドを置く必要があり，従来のベクトル演算以上に次の点で性能上の課題がある．(1)命令の立上りが遅い，(2)キャッシュ等の階層記憶機構への対応，(3)ベクトル化のオーバヘッドの考慮．3個の例題におけるベクトル化の課題と高速化の効果について，具体的な実験結果を報告する．関係データベースでは，複数の検索条件のある問合せ処理に適用し，複数のインデックス利用方式により4倍の高速化を実現した．ソートユーティリティでは，2ウェイのマージ命令を用いてマルチウェイのマージをベクトル化し，4倍高速化の結果を得た．解探索のNクィーン問題においても，2倍の高速化を達成できた．

Acceleration Method for Non-numerical Processing Using a Vector Merge Function

Shiun'ichi Torii,[†] Keiji Kojima,[††] Yasusi Kanada,[††]
Akiharu Sakata[††] and Masami Takahashi[†††]

Methods and problems to utilize a merge vector funtion for non-numerical processing are presented. The Integrated Database Processor (IDP) has highly pipelined merge and search vector functions in addition to conventional vector functions. A vector merge instruction manipulates 3 independently indexed vector operands on main storage, so there are these problems to improve performance. (1) startup time in merge/search vector instructions, (2) accommodation to hierarchical storage system in very long vector operand, (3) vectorization overhead. IDP can be utilized not only for primitive relational operations but also for the following new applications: (1) Relational query using multiple indexes, (2) Multi-way merge/sort utility, (3) Generate and test processing. Vectorization methods and performance improvements in these applications for IDP are also described and analyzed.

1. はじめに

ベクトルプロセッサは，スーパコンピュータとして数値計算分野では大きな成功を収めており，数多くの応用プログラムがその高速性を生かすべく研究されている[1]．しかし，現在のベクトルプロセッサは，数値処理分野には適合しているが，動的データ構造を中心とする非数値処理分野にはうまく適合していない．この様な動的構造プログラムにベクトルプロセッサを適用するには，プログラムを書き直してベクトル型のデータ構造に変換するか，リスト構造にリンクするベクトル型のデータ構造を追加するなどの処理が新たに必要である．この書き直しはベクトル化と呼ばれるが，実行時の効率を考えると一般にオーバヘッドを伴っている．

上記のオーバヘッドに加えて，従来のベクトルプロセッサでは，ベクトル演算が適用できる範囲が限られていた．第一に，従来のベクトルプロセッサでは各ベクトルオペランドのデータタイプと要素長は固定，かつ同一でなければならなかった．これに対して，内蔵データベースプロセッサ(IDP)では，パイプライン方式のマージ／サーチ型のベクトル演算機構を持ち，主記憶上の各ベクトルオペランドごとにデータタイプや要素数の設定を可能にした．第二に，非数値処理でしばしば使用されるタグ付きデータ構造に対しては，

† 日立製作所システム開発研究所
Systems Development Laboratory, Hitachi, Ltd.
†† 日立製作所中央研究所
Central Research Laboratory, Hitachi, Ltd.
††† 日立製作所ソフトウェア開発本部
Software Development Center, Hitachi, Ltd.

デュアルベクトルと名付けた新しいベクトル形式をIDP命令で操作することを可能とした．IDPは本来関係データベース処理を高速化する目的で開発されたものであるが[2),3),23)]，他の多くの種類の非数値処理にも適用可能である点が，他のデータベースプロセッサ[4)~8),25)]と異なる特長である[12),24)]．

本論文では，マージ型演算に拡張されたベクトルプロセッサIDPにおける高速化の課題と具体的な適用例について述べる．高速化における課題としては，ベクトル化オーバヘッドを克服するためのものと，オペランドを主記憶上に置くことに起因するものとに分けられる．第2章では，従来のベクトルプロセッサを非数値処理に適用した場合の問題点と，IDPでの解決策を簡単に示した後，ベクトル化オーバヘッドについて定義する．第3章ではマージ演算に固有の課題をアーキテクチャと適用ソフトウェアの両面から示す．第4章では3個の適用例と性能評価結果について述べる．特にベクトル化のオーバヘッドとIDPによる高速化の課題について実験結果を基に考察する．

2. IDP アーキテクチャ

2.1 従来ベクトルプロセッサアーキテクチャの問題点

ベクトルプロセッサは，今まで数値計算分野の専用プロセッサとして考えられてきたが，非数値計算分野でも高い可能性を持っている．たとえば，スーパコンピュータS810[9)]を推論処理に適用した例[10)]，記号多項式計算のベクトル化[20)]などが知られている．しかしながら，一般には従来のベクトルプロセッサの基本演算機能（算術演算，論理演算，条件付き演算，間接アドレス演算等）だけでは巧妙なベクトル化アルゴリズムによっても，高い適用率が得ることは容易ではない．たとえば整列処理について言えば，多くの試みがなされているものの実用的なベクトル化方法は知られていない[11)]．具体的には，次の2点が従来のベクトルプロセッサアーキテクチャの大きな問題点と考えている．

（1） ベクトル演算要素への指標付け

従来のベクトル演算では，各ベクトルオペランドは同一の指標を持った要素間でしか演算が許されていない．すなわち，$X(i)*Y(i)\rightarrow Z(i)$ 型のみが可能で，3個の独立した指標を持つ $X(i)*Y(j)\rightarrow Z(k)$ 型は禁止されている．したがって，例えば**図1**に示す様な3個のインデックスの更新（正確には増加）が比較結果により独立に制御されるマージ演算のベクトル化は不可能であった．また，i と j は常に更新されるが k の更新が比較結果に依存するサーチ型の演算は，マスク付き演算機能を利用することによりベクトル化が可能ではあるが，条件の成立率が低いと効率の低下を招きやすい問題があった．

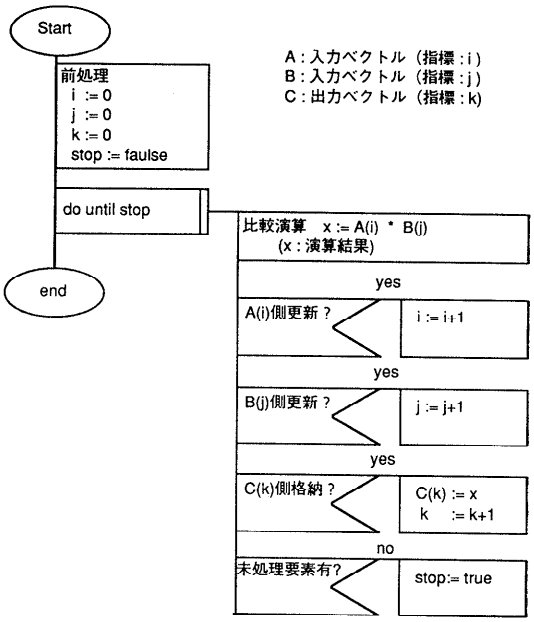


図1 マージ型演算形式
Fig. 1 Architecture of merge operation.

（2） データ構造

従来のベクトル演算が取り扱うことができるデータ構造はベクトルであるだけでなく，各ベクトル要素が単一値でなければならない．したがってタグ付きデータの操作は大きなオーバヘッドを生じる原因となる．

2.2 IDP のアーキテクチャ

上述の問題に対して，IDPでは次の解決策を既に実現した[3),12),23)]．

（1） 各ベクトルオペランドごとの指標

各指標は毎回の演算ごとに独立に更新されるので，マージやサーチの演算が可能となった．

（2） デュアルベクトルの採用

IDPで使用可能なベクトル命令は次の3種に分類できる．

（a） マージ法による集合演算（和，差，積の3命令）

ソート演算命令は和集合演算命令，ジョイン演算命令は積集合演算命令と考えることができる．

（b） 逐次サーチ演算（等号，不等号，大小など6

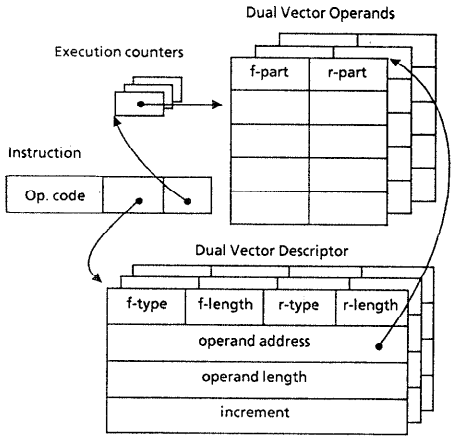


図 2 IDP 命令形式
Fig. 2 IDP instruction format.

命令)

(c) 従来のベクトル演算(論理, 算術, 移動など)

上記(a), (b)の演算が新たに拡張されたもので, 図2の命令形式に示すようにデュアルベクトル形式の3個のオペランドに対して実行される. デュアルベクトルの各要素は, 2つのパートすなわちフロント部とリア部から構成されるので, 任意の二進関係を表現することが可能である. 通常は, フロント部にタグ情報や識別情報を格納し, リア部にソートやサーチでの比較対象となるキー情報を格納する. 関係データベースの基本演算である選択, 結合, 重複排除, 統計演算にIDP命令を適用した例では, 各デュアルベクトルのフロント部にはレコードの内部識別情報 (Row Id), リア部には演算対象カラムの値を格納している. さらに内部処理のアドレス変換にも適用できる[3),23)].

以上の基本アーキテクチャに対し, 実用上の課題を3.2節で具体的に説明する.

2.3 ベクトル化オーバヘッド

一般に, プログラムをベクトル化するために書き直しを行うと, 何らかのオーバヘッドを生ずる. ここでは, ベクトル化オーバヘッドを図3に示したように, オリジナルなプログラムの実行時間と, ベクトル化プログラムのスカラ実行時間の差と定義する. 良く知られているように, 高い性能向上率を得るためには, 加速率とベクトル化率が重要な要素である. しかし, マージやサーチ処理のプログラムをベクトル化する場合には, ベクトル化オーバヘッドについても, 十分な配慮をする必要がある. ベクトル化オーバヘッドの原因としては次のことが考えられる.

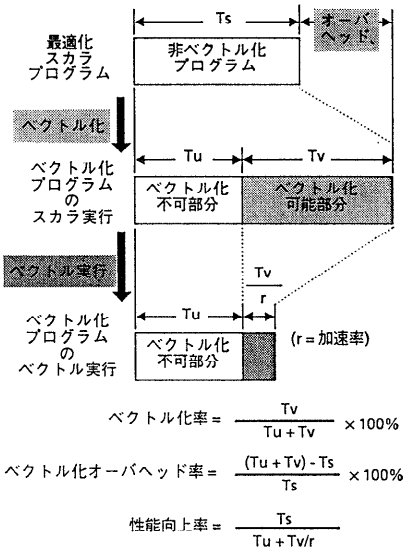


図 3 ベクトルプロセッサによる性能向上
Fig. 3 Performance improvement by vector processor.

(1) データの構造をベクトル形式に, 静的に書き直したり, 動的に書き写したりする必要がある. IDPでは, デュアルベクトルの作成がこれに相当する.

(2) 効率の良いアルゴリズムから, ベクトル化に適した並列性の高いアルゴリズムに書き直す必要がある. IDPでは, ソートのアルゴリズムがマージ法に限定される.

(3) ループ構造の最内側と外側を入れ換えるような実行順序の変更を行ったため, 前後処理の増加やレジスタを経由した最適化の阻害などのソフトウェア上の性能低下だけでなく, ハードウェア的にもキャッシュなどの階層記憶の効率低下などが発生する場合がある.

しかし, これらは異なる2つのアルゴリズムの性能比較と考えるべきであり, 対象とするプログラムによりその影響は当然まちまちである. 場合によってはベクトル化によりむしろ処理効率が向上する例もあった. 具体的な非数値処理プログラムのベクトル化の実例とベクトル化オーバヘッドと性能向上率については第4章でさらに考察する.

以下, オーバヘッドを含め適用ソフトウェア上の課題については, 3.3節で対策を含めて説明する.

3. マージ演算に固有の高速化の課題

3.1 前提となる要求事項

ここでは IDP が開発された背景とそれに起因する

要求事項を説明する．IDPは，関係データベースの高速化が本来の目的であったため，以下の点を満たす必要があった．

（1） 複数ユーザの時分割使用環境下での稼働

（2） 仮想記憶機構下での稼働

（3） 付加機構としてのコスト／パフォーマンス維持

前半の2項目は実現上，特別な制約とは思われない．（3）からは，スーパコンピュータのような性能を追及した大規模な構成は採用できないことになるが，マージ演算は本来逐次的な性質があり，大規模な並列化は困難である．したがって上記要求事項はIDPに固有なものではなく，マージ演算ベクトル機構を実現した場合には全般的に広く当てはまるものと考える．

3.2 アーキテクチャ上の課題

（a） 命令実行中での割込み／再開の実現

3.1節で述べた（1）（2）の要求事項から，任意時点でのページ不在割込みや入出力／外部割込みを許す必要があった．IDPでは割込み時に，各オペランドのインデックスの進捗状況を汎用レジスタ（3個）に格納させた．

（b） ベクトルレジスタの使用困難

ベクトルレジスタ（VR）は，ベクトル演算における主記憶装置のレスポンスとスループットを改善するキャッシュとして広く採用されている．物理的なVRの要素長は制約があり，長い要素長のベクトル演算では，例えば256個ずつに分割して実行する必要があることが知られている．しかし，マージ演算のように各オペランドのインデックスが独立に増加する演算形態では，256個境界での分割処理がオペランドごとに必要となり，極めて複雑な処理手順となる．さらに，デッドロックを発生する可能性があるため，同一VRからの再読出しもできない．すなわち，読出しの使用目的ごとに256個境界に達する時点が異なるため，同一物理VR上に異なる256個の部分を保持する必要が生じるためである．再読出しだけでなく，VRの本数不足対策としての再割当においても同様な理由からデッドロックの可能性がある．以上の点から，マージ演算でのVR使用は困難であった．したがってIDPでは，前出の図2に示すように3個のオペランドすべてを主記憶上に置く形式にした．また，マージ演算以外の処理においても，間接指標付きの移動命令が良く使われるが，間接指標付けされるベクトルはやはりVR上ではなく主記憶上に置く必要がある．以上の2点から，マージ演算を使用するプログラムの高速化では，従来のベクトル演算以上に主記憶の処理能力が重要な要因と言える．

（c） 命令起動／終了オーバヘッド増加

各オペランドの定義情報だけでなく，前述のように各オペランドも主記憶に置いたため，VRを使用するベクトル命令と比較して，命令起動／終了処理時間がさらに増加する問題がある．これに対してIDPのソート命令では，1命令実行当たりのベクトル要素長を長くするため，**図4**に示すように全ベクトルを分割して2つの入力オペランドとし，各マージのフェーズが全ベクトルに対して1回の命令実行で完了するように工夫した．各マージ命令の実行により，ソート順が保証される部分（ストリングと呼ばれる）の長さが2倍になる．二分割の点は，ストリング境界に合わせる．

また，1命令に複数種類の機能を統合することも，主記憶への負荷減少と相まって有効である．サーチ命令は，スーパコンピュータであれば複数命令に分かれても並列同時実行により高速化が期待できる機能であるが，IDPでは1命令に統合し起動／終了処理時間を減少させるために新設した．

（d） 使用論理規模の削減

IDPの母体であるM 680 Hは，同じくオプション

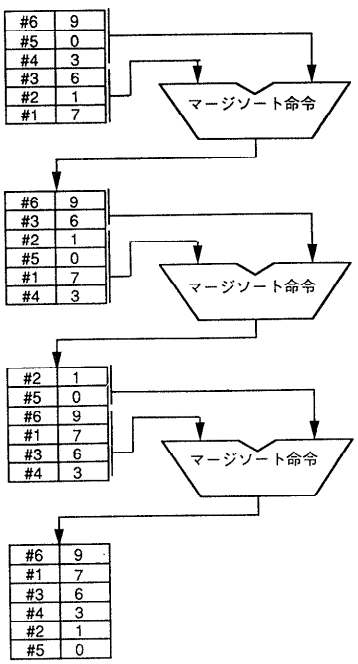


図4 ソート命令のベクトル長確保
Fig. 4 Keeping the long vector length in a merge sort instruction.

として内蔵アレイプロセッサ（IAP）が装備可能である[13),14)]．従来のベクトル演算命令が IAP 命令を利用して実現されているだけでなく，拡張された IDP 命令についてもベクトル読出し／書込み制御については IAP との共有化をはかることにより，IDP オプションを M 680 H 本体の 5% 以下の論理規模で実現できた[3),23)]．

（e） マージ演算命令の高速化

IDP も，IAP のパイプライン制御方式を拡張することにより，オペランド読出し，比較演算，結果格納をパイプライン化し，1サイクルピッチで演算結果を得ることが可能である[3),23)]．スカラ処理に比べ，10倍以上の高速化が実現できている．これに対して，通常のベクトル演算は，スカラ命令側の高速化が進んだため付加機構のレベルでは数倍程度が限界であるものが多い．

3.3 適用ソフトウェアの課題

前節の課題の多くは，ソフトウェアの課題にもなっている．

（i） マージ演算／命令セットの制約によるオーバヘッドの克服

2.3 節で述べたベクトル化オーバヘッドを考慮し，デュアルベクトルのフロント／リア構成や生成方法には，転送やフロント／リア入替えを削減する工夫が必要である．

（ii） ベクトル長の確保

IDPでは，命令の起動／終了処理時間が約100要素分の正味実行時間に相当するので，通常のベクトル演算以上にベクトル長の確保が必要である．したがって要素数が十分に長いときの加速率が10倍であっても，要素数が10個以下ではベクトル化の効果がなくなる．

（iii） 階層記憶への対応

マージソート処理では各ベクトル要素が複数回参照されるので，階層記憶機構のキャッシュ（バッファ），2次キャッシュ，割当実記憶の容量を考慮した処理方式が必要となる．例えば，要素数が1万個を越すベクトルのソートでは，キャッシュ容量分に分割して部分ソートを実行したほうがキャッシュミスによる劣化が軽減でき，ベクトル長減少の弊害は無視できることが報告されている[3),23)]．また，各レコード長の長いレコード群のソートでは，レコード本体を動かすより，キーとポインタよりなるデュアルベクトルを動かしたほうが良いことも明らかである．さらに，キー長が長く比較的値の分布に片寄りがなければ，キー部分も上位のみ抽出して粗いソートを掛けたほうが，処理時間を短縮できることも報告されている[22)]．

（iv） ベクトル化率の向上

ベクトル化率が重要な性能要因である点は，通常のベクトル演算と全く同じである．マージ演算では，デュアルベクトルの作成が，オーバヘッドとなっており，この部分もベクトル化して実行時間を短縮することが極めて重要である．第4章の3個の制題は，それぞれマージ命令またはサーチ命令を使用するものであるが，いずれもデュアルベクトルの作成をベクトル化している．

4. 非数値処理プログラムのベクトル化例における高速化の課題

4.1 関係データベースにおける複数インデックス利用

最初の例題は，関係データベースにおける複雑な検索条件が付いた問い合わせ処理への適用である．従来の多くの RDBMS は複雑な検索条件であっても，通常は1個のテーブルに対して1個のインデックスしか利用していない場合が多い[15)]．たとえば，2個のカラムの各々に検索条件が AND で指定された場合，より厳しいと思われる検索条件の付いたカラムのインデックスしか利用していない．残りの条件は，各レコード本体の内容を読み込んで評価する必要がある．この方式の利点は，条件に一致する最初のレコードが早く求まる，作業用に予測不能な大きい領域が不要などの点である．欠点としては，選択したインデックスが適切でないとインデックスがあるにもかかわらず多くのレコード本体を読むことになる点である．

これに対して，IDPでは可能な限り多くのインデックスを利用する方式を採用した[12),21),24)]．**図5**の検索例では，我々がベースとした RDBMS では Title カラムの条件が等号なので，Title カラムのインデックスのみを利用していた．採用した方式では，最初に Title カラムのインデックスを用いて Title が 'Physics' のレコードの Row ID のベクトルを作り，それをソート命令を用いて Row ID 順に整列する．次に，同様に Year カラムのインデックスを用いて発行年が1984年以降のレコードの Row ID のベクトルを作り，整列する．最後に2つのベクトルをマージジョインにより集合積を求め，最終的なレコードの総数と Row ID を得る．

OR 条件についても同様なベクトル化が可能であ

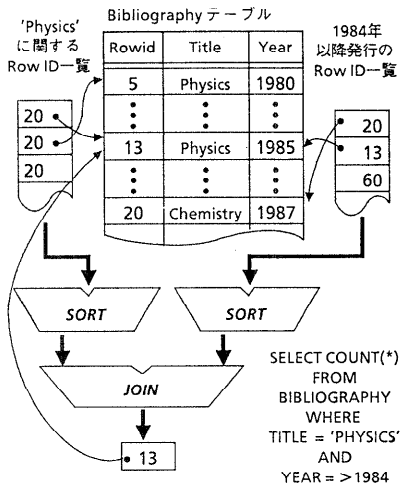


図 5 2インデックス処理
Fig. 5 Query using two indexes.

る．最後のマージジョイン命令をマージソートの集合和命令と重複の排除のサーチ命令で置換する．否定についても，集合差の演算に IDP を利用することができるので，原理的には条件のすべての論理的組合せに対して IDP が適用可能である．本方式は情報検索システムにおいて広く使用されているものであるが，それぞれの CPU 処理に IDP 命令が適用可能であり，マージ演算向きの方式と言える．

この方式の欠点は従来方式の利点の裏返しであり，最初の該当レコードが確定するまでに最後の該当レコードとほとんど同じ処理時間を要する，ソート処理に時間がかかる，ソート用に作業領域が必要などが考えられる．最初の2つは IDP を利用することにより改善できた．最後の問題は，半導体高集積化技術の進歩による大容量主記憶が解決できる．

表1は，前述の検索例での CPU 時間の実測値である．ここでベクトル化のオーバヘッドは，「スカラ最適化プログラムの実行時間」と「ベクトル化プログラムのスカラ命令のみによる実行時間」の差と定義され，この適用例では200%近い大きな値となる．この原因は，ソートアルゴリズムの相違である．スカラ最適化プログラムでは，高速なクィックソートのアルゴリズムを利用しているのに対し，ベクトル化プログラムはマージソートのアルゴリズムに書き替える必要があった．**図6**は，上記検索処理のうち Row ID 情報の抽出とソート処理に相当する部分のCPU処理時間の内訳を推定したものであり，マージソートがクィックソートより効率が悪いが，ベクトル化により高速化が実現されていることを示している．ソート処理部分の実行時間については，各種ソフトウェアアルゴリズムと IDP 命令を別途比較した実験からも，クィックソートはマージソートよりも2倍以上早いが，IDP命令はさらにクィックソートより最大10倍近く高速であることが報告されている[3),23)]，検索処理全体としても，ベクトル化率が98%と高率であることと相まってベクトル化オーバヘッドを克服し，4倍の高速化が実現できた．

表 1 2インデックス処理実行結果
Table 1 Measurement result of Query using two indexes.

測定条件	CPU 時間
最適化スカラプログラム	72 msec
ベクトル化プログラムのスカラ実行	207 msec
ベクトル化プログラムのベクトル実行	18 msec

注）各インデックスより 8000 エントリ抽出

この例題では，デュアルベクトル生成に相当する処理が Row ID 抽出であるが，特にオーバヘッドにはなっていない．むしろ，図6では処理ルーチンのリコーディングにより最適化が図られ，さらに IAP 命令で高速化されている．残された性能上の課題は，ディスクからインデックス情報を読みだすための VSAM アクセス法等の OS 処理の高速化である．これには，一括読出し等が必要である．この RDB の例は，積／和等の集合演算をソートとマージの手法で処理しようとする汎用的なアルゴリズムのベクトル化例

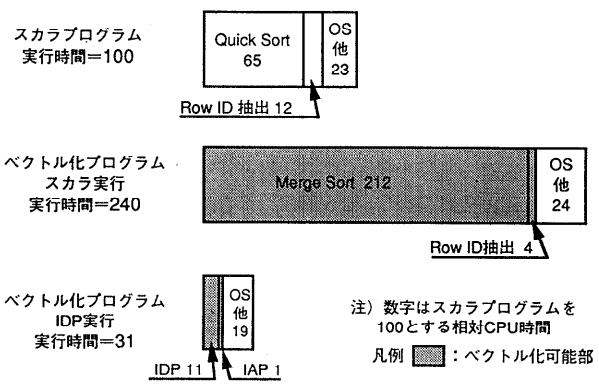


図 6 Row ID リスト作成処理内訳
Fig. 6 Breakdown of the construction of Row ID list.

であり，RDBMS 以外の多くの非数値処理においても IDP が効率的手法として利用できる可能性を示している．

4.2 ソートユーティリティ

ソートユーティリティは事務計算分野において，もっとも普及しているプログラムの1つである．特にマルチウェイマージは，ワークファイルへの I/O を削減するために広く利用されている．一方，IDP ソート命令は2ウェイを基本とした命令であるので，2ウェイマージ器を使った効率の良いマルチウェイマージ実現法を開発した[12]．

図7は，新マルチウェイマージ法でのデュアルベクトル関連のデータの流れを示したものである．2進木構造の各ノードは，この図では説明の都合上3個であるが，ベクトルプロセッサの起動オーバヘッドを削減するため，実際は数百個以上の要素からなるデュアルベクトルである．各デュアルベクトルの要素のフロント部は，主記憶上に読み込まれたレコードの主記憶アドレスを格納し，リア部はレコードのキー値を格納する．デュアルベクトルはリーフからルートに向かってマージされ，外部ファイルには完全にソートされたレコードが順次書き出される．この図では，最初の3個レコードが書き出され，ルートノードが空になった状態を示している．この図には示していないが，入力がランダムな場合には前処理として，主記憶に格納できる範囲で入力を分割／ソートし，それぞれをファイルに書き出しておくソートフェーズ処理が必要である．しかし，この処理も，デュアルベクトルを作成すれば，IDP のソート命令が直接適用できる．

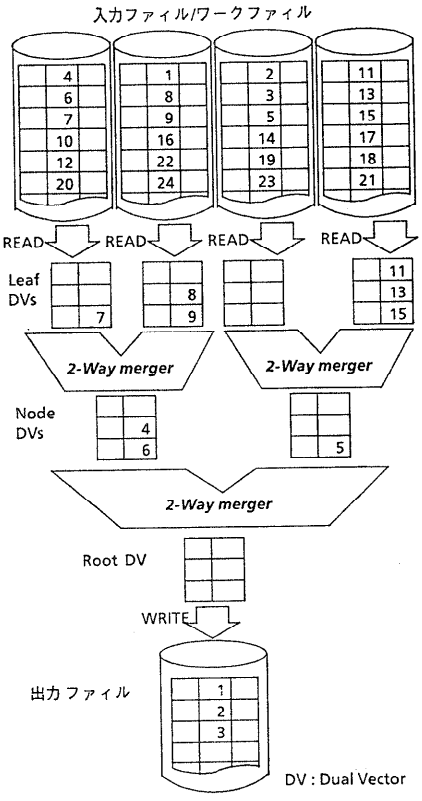


図 7 新多ウェイマージ法のデータの流れ
Fig. 7 Data flow in multi-way merge using 2-way vector merger.

本アルゴリズムを PAD[16]で記述したものを**図8**に示す．本アルゴリズムは，リプレイスメントセレクション法[17]と，データ構造が2進木となっている点で似ているが，以下の点に大きな相違がある．

(1) 各ノードは1個のレコードと対応しておらず，数百個のレコードと対応する数百要素のデュアルベクトルである．

(2) 2つのノード間の比較は，ベクトル間の比較として IDP マージソート命令で実行される．

(3) 木構造はルートからリーフに向かって各ノードの空き状態に応じてスキャンされる．

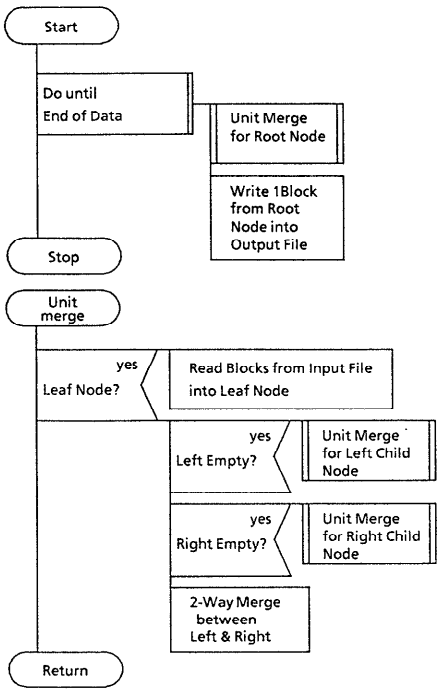


図 8 新多ウェイマージ法の処理手順
Fig. 8 New multi-way merge using 2-way merger.

表 2 マージソートユーティリティ実行結果
Table 2 Measurement result of merge/sort utility.

測定条件	CPU 時間
最適化スカラプログラム (Replacement Selection)	5.24 sec
ベクトル化プログラムのスカラ実行 (Vector Merge Sort)	4.59 sec
ベクトル化プログラムのベクトル実行 (IDP Merge Sort)	1.38 sec

注) 30B×500K レコードのソート (4 byte キー)

表 2 は，入力がランダムな場合の実行結果を示したものである．この応用ではベクトル化のオーバヘッドは存在していない．これは各ノードでの比較では，1 レコードごとより，多数レコードごとに一括実行する方式が効率の面で優れているためと考えられる．すなわち，ループ構造を入れ換えることにより，最内側の比較処理のループが，ノードすなわちレコードごとからベクトル要素ごとに最適化できるためと思われる．一方，ベクトル化率は 80% に留まっているため，全体の性能向上率は 3 倍程度である．ベクトル化できていない部分の主な処理は，主記憶上でのレコード本体の移動と，ディスク入出力の OS 処理である．前者は，性能向上が IDP ではあまり期待できないのでベクトル化を実施していないが，スーパコンピュータのような超高速なデータ転送機能があれば，この分もベクトル化して処理時間が短縮できることが確かめられている．後者については，大容量拡張記憶等を利用できればベクトル処理とは独立に削減が可能である．

この性能面の問題は，レコード長が長くなると上記ベクトル化できない部分の比率が増加し，一層顕著になる．**図 9** は，レコード長を変えた場合の IDP での CPU 時間内訳 (推定) を比較したものである．30 バイトのレコードの場合には，全処理の約 1/3 がベクトル処理であるのに対して，400 バイトのレコードの場合には 1/20 以下となる．これは**図 10** に示したように，レコード本体の主記憶上での移動が，二進木上を沿った移動をなくしてもレコード当たり 4 回も存在するためである．

ベクトル長については，入力レコード数やキー値の分布にもよるが十分な長さが確保できた．30 バイトのレコード 60 万個のソート処理を 8 個のワークファイルのマージで実施した場合では，リーフ以外のノードに約 800 要素のデュアルベクトルを用意すれば，1 回のマージ命令当たり平均 500 個以上の要素が出力さ

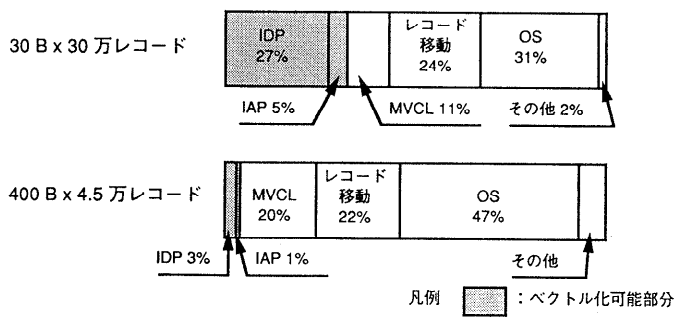


図 9 マルチウェイマージソートの処理時間内訳
Fig. 9 Breakdown of CPU time in multi-way merge sort.

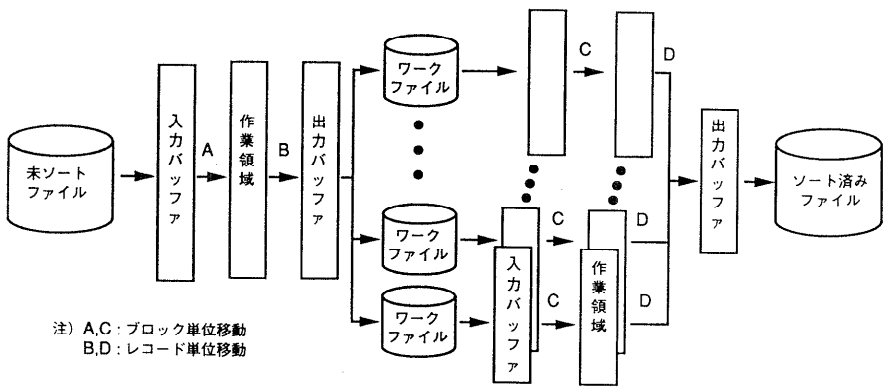


図 10 ソート全体でのレコード情報の移動経路
Fig. 10 Data flow of records in the merge sort process.

れている．各ノードの容量を増せばさらに，ベクトル長の増加が期待できるが，以下の理由で比較的少容量に抑えている．前述の図10に示したようにマージフェーズにおいて，レコード本体の移動回数を減らすためレコード本体はリーフからルートに直接転送される．各リーフノードではルートから出力作業領域に転送される時点までレコードの内容保持を保証しつつ，かつ速やかに空き作業領域には後続レコードを取り込まなくてはいけない，この各リーフノードのレコード管理を高速化するために，ルート以外の木構造の上位ノードすべてが該当リーフノードからのレコードで満たされた場合を想定した予測制御を行っている．すなわち，各リーフノードのレコード用作業領域には上位ノード段数分のレコード領域を別に確保する必要がある．したがって，リーフ以外のノードの容量を少なくし，実効的なリーフノードの容量を多くしないと，マージのCPU処理とワークファイル読出しのI/O処理との並行化が阻害される．

4.3 *N*クィーン問題

解探索問題への適用例として，有名な N クィーン問題[18]をここでは取り上げる．使用するアルゴリズムは，並列バックトラック法[19]に基づいている．ここでは，第1カラムから第 $n-1$ カラムまでクィーンが安全に配置できた解の各々に対して，第 n カラムにクィーンを配置し，先に置いた $n-1$ 個のクィーンとの競合をチェックする処理が中心となる[12]．IDPでは，チェック処理にサーチ命令 (Sequential Search Not Equal : SQSNE) を使用し，チェック操作とチェックで残った解の収集操作を同時に行える．これに対して従来のベクトル演算では，チェック操作では比較命令でベクトルマスクだけを生成し，残りの解の収集操作は別ベクトル命令が必要となる．

図11は，盤面の大きさを，N=5から11までについて実行した結果である．ここで“スカラ最適化版は”，通常のバックトラック法のアルゴリズムを用いたFORTRANプログラムである．“並列バックトラック版IAP単独”は，従来のベクトル命令（IAP命令）を使用したプログラムの実行結果を示し，“並列バックトラック版IDPとIAP使用”は，IAPとIDPの両ベクトル命令を使用した実行結果を示す．ここでは，150%程度のベクトル化オーバヘッドが存在するが，IDPとIAPの組合せでは，スカラ最適化版と比較して2倍近い高速化を実現している．IAP単独では，残った解の収集に適当な命令がないため，マスク付き演算機構を使用している．この機能により解の収集処理の命令はなくせるが，マスクの値が“0”の場合が多いため，やはり性能が出ない問題が残る．

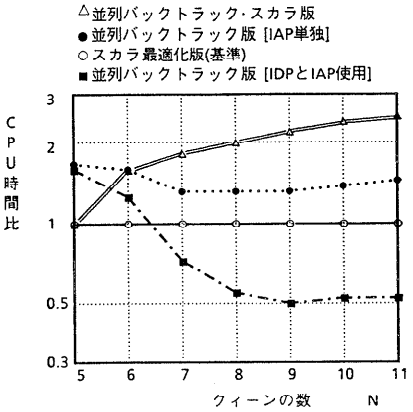


図11 Nクィーン実行結果
Fig. 11 Measurement result of N queen problem.

IDP高速化の課題は，サーチ命令の入力となるデュアルベクトルの作成処理の高速化である．この部分は，IAP系の命令でベクトル化されているが，実行時間ではサーチ命令以上の実行回数と実行時間を要している．

5. むすび

マージ演算のベクトル機構（内蔵データベースプロセッサ：IDP）を用いた高速化を3個の非数値処理の例題で実施し，以下の点をマージ演算の適用上の課題として示した．

（1） 主記憶上にオペランドを置かざるをえないことに起因するベクトル長の確保，階層記憶への対応

（2） 主にマージ演算／命令セットの制約に起因するベクトル化オーバヘッド

今後の課題としては，主記憶上の移動を中心とする通常ベクトル演算の強化，複数機能を統合化した専用命令の新設が挙げられる．

謝辞 本研究の機会と数多くの助言を頂いた，堀越彌氏，千葉 常世氏，高橋 重捷氏，小高 俊彦氏および吉住誠一氏に深謝します．

参考文献

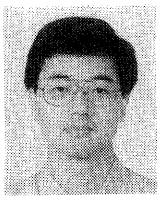
1) Hwang, K.: *Tutorial on Supercomputers: Design and Applications*, IEEE Computer Society Press (1984).

2) Torii, S., Kojima, K., Yoshizumi, S., Sakata, M., Takamoto, Y., Kawabe, S., Takahashi, M. and Ishizuka, T.: A Database System Architecture Based on a Vector Processing Method, *Proceedings of Third International Conference of Data Engineering*, pp. 182-189 (1987).
3) Kojima, K., Torii, S. and Yoshizumi, S.: IDP —A Main Storage Based Vector Database Processor—, *Proceedings of 5th International Workshop on Database Machines*, pp. 60-73 (1987).
4) Hsiao, D. K.: Data Base Computers, *Advances in Computers*, Vol. 19, pp. 1-64 (1980).
5) Tanaka, Y.: A Data Stream Database Machine with Large Capacity, *Proceedings of International Workshop on Database Machines* (1982).
6) Kitsuregawa, M., Tanaka, H. and Moto-Oka, T.: Architecture and Performance of Relational Algebra Machine Grace, *Proceedings of International Conference on Parallel Processing*, pp. 241-250 (1984).
7) Shibayama, S., Kakuta, T., Miyazaki, N., Yokota, H. and Murakami, K.: Query Processing Flow on RDBMS Delta's Functionally-Distributed Architecture, *Proceedings of International Conference of Fifth Generation Computer Systems* (1984).
8) Neches, P. M. and Shemer, J.: The Genesis of a Database Computer, *IEEE Comput.*, Vol. 17, No. 11, pp. 42-56 (1984).
9) Nagashima, S., Inagami, Y., Odaka, T. and Kawabe, S.: Design Consideration for a High-speed Vector Processor: S-810, *Proceedings of IEEE International Conference on Computer Design '84*, pp. 238-242 (1984).
10) Kanada, Y. and Sugaya, M.: Vectorization Techniques for Prolog, *Proceedings of 1988 International Conference on Supercomputing*, pp. 539-549 (1988).
11) 石浦菜岐佐，高木直史，矢嶋脩三：ベクトル計算機上でのソーティング，情報処理学会論文誌，Vol. 29, No. 4, pp. 378-385 (1988).
12) Torii, S., Kojima, K., Kanada, Y., Sakata, M., Yoshizumi, S. and Takahashi, M.: Accelerating Non-numerical Processing by an Extended Vector Processor, *Proceedings of Fourth International Conference of Data Engineering*, pp. 194-201 (1988).
13) Wada, K., Nagashima, S. and Odaka, T.: Design for a High Performance Large-Scale General Purpose Computer, The HITACHI M-680 H Processor, *Proceedings of IEEE International Conference on Computer Design : VLSI in Computers* (1985).
14) Takanuki, R. and Umetani, Y.: Optimizing FORTRAN 77, *Hitachi Review*, Vol. 30, No. 5, pp. 241-246 (1981).
15) Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A. and Price, T. G.: Access Path Selection in a Relational Database Management System, *Proceedings of 1979 ACM SIGMOD Conference*, pp. 23-34 (1979).
16) Futamura, Y. and Kawai, K.: Problem Analysis Diagram (PAD), *JARECT*, Vol. 12, オーム社 (1984).
17) Knuth, D. E.: *Searching and Sorting*, *The Art of Computer Programming*, Vol. 3, pp. 251-262, Addison-Wesley (1973).
18) Colomb, S. W. and Baumert, I. D.: Backtrack Programming, *JACM*, Vol. 12, pp. 516-524 (1965).
19) 金田 泰，小島啓二，菅谷正弘：ベクトル計算機のための探索問題の計算法「並列バックトラック計算法」，情報処理学会論文誌，Vol. 29, No. 10, pp. 985-994 (1988).
20) 村尾裕一：ベクトル計算機による記号多項式計算，情報処理学会記号処理研究会資料，62-3 (1991).
21) 久代康雄，阿部 淳：関係データベース RDB1 —集合演算に対するベクトルプロセッサの適用—，第 34 回情報処理学会全国大会論文集，pp. 497-498 (1987).
22) 坂田明治，三科雄介，小島啓二，鳥居俊一：内蔵データベースプロセッサ IDP—拡張ソート処理—，第 35 回情報処理学会全国大会論文集，pp. 413-414 (1987).
23) 小島啓二，鳥居俊一，吉住誠一：ベクトル型データベースプロセッサ IDP，情報処理学会論文誌，Vol. 31, No. 1, pp. 163-173 (1990).
24) 鳥居俊一，小島啓二，金田 泰，坂田明治，吉住誠一，高橋政美：拡張ベクトル演算による非数値処理高速化，信学会技報，Vol. 88, No. 56, DE 88-8, pp. 57-64 (1988).
25) Inoue, U., Satoh, T., Hayami, H., Nakamura, T. and Fukuoka, H.: Rinda: A Relational Database Processor with Hardware Specialized for Searching and Sorting, *IEEE Micro*, Vol. 11, No. 6 (1991).

(平成4年2月6日受付)
(平成4年10月8日採録)

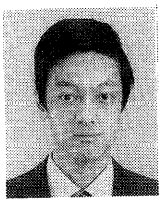
鳥居　俊一（正会員）

1949 年生．1971 年東京大学工学部計数工学科卒業．1973 年同大学院修士課程修了．同年(株)日立製作所中央研究所入社．以来，大型計算機，スーパコンピュータおよびデータベース高速化の研究に従事．現在同社システム開発研究所第 6 部主任研究員．



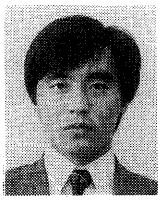
小島　啓二（正会員）

1956 年生．1980 年京都大学理学部卒業．1982 年同大学院修士課程修了．同年(株)日立製作所中央研究所入所．以来データベースマシン，ユーザインタフェースの研究に従事．ソフトウェア科学会会員．



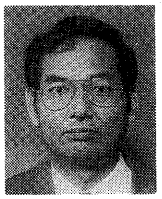
金田　泰（正会員）

1956 年生．1979 年東京大学工学部計数工学科卒業．1981 年同大学院情報工学専門課程修了．同年 4 月に(株)日立製作所中央研究所入所後，Fortran コンパイラ開発，スーパコンピュータの論理型言語処理系の研究に従事．現在は同研究所から新情報処理開発機構つくば研究センタに出向し，自己組織的計算のモデルおよびその大量文字情報処理への応用などの研究に従事．1985 年山内奨励賞受賞，1991 年元岡賞受賞．計算言語とその処理系，部分情報からの自己組織的計算機構などに興味をもつ．ACM，ソフトウェア科学会各会員．



坂田　明治（正会員）

1958 年生．1982 年東京理科大学理学部数学科卒業．1984 年北海道大学大学院修士課程修了．同年(株)日立製作所中央研究所に入所．現在大型計算機関係の仕事に従事．



高橋　政美（正会員）

1949 年生．1972 年東北大学工学部原子核工学科卒業．現在，(株)日立製作所ソフトウェア開発本部にてデータベース管理システムの開発に従事．