Ｌｉｓｐからのベクトルプロセッサの利用

1Ｇ－9

植松尚士，小林一隆，安井 裕
(大阪大学・工学部)

1，はじめに

我々は，Ｌｉｓｐ環境でのより高速な処理を目的として，ベクトルプロセッサ，すなわちスーパーコンピュータのベクトル演算パイプライン[1]に着目し，スーパーコンピュータ上で機能するＬｉｓｐインタプリタを試作した．この処理系では，配列型データに対する演算の拡張を行なう一方，配列処理用の関数を追加して並列性の明確なプログラミングができるように言語仕様の拡張を行ない，配列演算のみならずＬｉｓｐ処理系の一部をベクトルパイプライン上で処理することを試みている．我々はこの処理系をＶｅｃｔｏ-Ｌｉｓｐと名付けており，NEC SX-2N上に試作し，実験した．

本稿では，試作処理系の概要及びそのベクトル演算機能の利用による効果について述べる．

2．設計方針

本研究では，

・配列型に対する組み込み関数はできる限りベクトルパイプラインを使用する．
・配列型に対する有効な演算を強化する．

の二つの方針に従い，Ｌｉｓｐ処理系を試作した．

試作処理系では，以下のような規則で配列に対する演算の拡張を行なった．

1. 引数の両方がスカラの時にはそのまま演算した結果を返す．
2. 引数の一方がスカラで他方が配列の場合は，配列の各要素に対しスカラをそれぞれ演算する．そして，それぞれの演算結果を要素とする新たな行列を生成して返す．
3. 引数の両方が配列の時はさらに引数の配列の各次元の寸法により分類され，
   3．1 両方の配列の各次元の寸法が同じ時には，対応する要素同士の演算をして，その演算結果を要素とする新たな配列を生成して返す．
   3．2 一方の配列が１次元でその寸法が他方のある次元の寸法と等しい時，１次元の配列が他方配列の別の次元方向に繰り返し演算される．
   3．3 上記以外の場合は，エラーである．

このように，もともと適用が許されなかった演算が四則についてＡＰＬ的に拡張され，ベクトルパイプラインで処理される．

また，比較関数（＝，＜，＞，＜＝，＝＞）についても同様に配列に対してベクトル化が行なえる．非数値に関しては，”＝”のみ配列に対して拡張することができる．配列対スカラの比較関数は五種類用意した．

また，本処理系ではベクトルパイプラインで高速に処理される配列を対象とした関数を多数用意しているが，以下にその一部の関数を示す．

(XLAT ＜ベクトル＞ ＜配列＞ ＜倍率＞ ＜定数＞)
　＜機能＞ 配列の各要素に倍率を掛け，定数を足した位置にあるベクトルの要素で置き換えた配列を返す．

(MAKE-ARRAY ＜大きさ宣言リスト＞ ＜初期値＞)
　＜機能＞ 配列を生成し，初期値を設定する．

(COPY-ARRAY ＜配列＞)
　＜機能＞ 配列をコピーする．

(CHKARRAY ＜配列＞)
　＜機能＞ 配列の次元と大きさを調べる．

(V-EXTEND ＜配列＞ ＜拡張数＞)
　＜機能＞ ベクトルの転置と拡張をする．

(SUM-VECTOR ＜配列＞ ＜次元＞)
　＜機能＞ 配列をある次元方向に足し合わせる．

(SORT-VECTOR ＜ベクトル＞)
　＜機能＞ ベクトルの要素をソートし大きい順の位置を返す．

(VECTOR-LENGTH ＜ベクトル＞)
　＜機能＞ ベクトルの要素数を求める．

(COMPRESS-VECTOR ＜ベクトル１＞ ＜ベクトル２＞)
　＜機能＞ ベクトルの要素を選び，取り出す．

(EXTEND-VECTOR ＜ベクトル１＞ ＜ベクトル２＞)
　＜機能＞ COMPRESS-VECTORの逆の操作をする．

(TRANSPOSE ＜配列＞ ＜次元１＞ ＜次元２＞)
　＜機能＞ 任意の次元で配列の転値をする．

なお，入出力及び要素へのアクセスに対しては配列の要素にシンボル，数値，リスト，ストリング等Ｌｉｓｐのオブジェクトを全てとることができる．また，配列の要素に配列をとることもできる．

現在，配列は３次元まで使用可能である．しかし，これについては，容易により多次元に拡張することができる．ただし，ある次元方向の大きさが０である配列は許していない．

3，インプリメント

試作処理系では，記憶空間はリストセル空間と配列空

Implementation of Vecto-Lisp System
on the Supercomputer SX-2N.
Takashi UEMATSU, Kazutaka KOBAYASHI, Hiroshi YASUI.
OSAKA University.

間から構成されている．配列データは配列空間に割り当てられ，その内部表現は図１のようになっている．配列データは全て１次元の配列に展開されている．ポインタが指している最初のワードは，その配列の次元数が整数型の数値として格納されている．次のワードは，実行時型判別の処理のために予約されているが，現在使われていない．その後は，各次元方向の大きさが格納されている．その次のワードから，実際の配列要素のポインタが線形に格納されている．数値データの演算は全て単精度実数で行なっている．

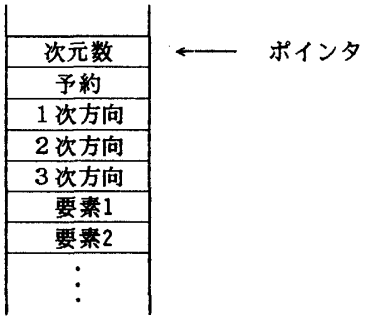

図１．配列データの内部表現

ストリングは１次元のベクトルにされ，その要素の文字コードが整数に変換されて格納されている．

シンボルは，

・７８文字までのＰＮＡＭＥ
・大域値
・関数またはマクロの定義

のための領域が個々に確保されている．

ガーベジコレクションはリストセル空間と配列空間がまとめて行なわれる．セル空間ではマーク後にガーベジがそのまま回収されるが，配列空間では新しい配列をつくるのに連続した領域を確保する必要があるため，生きているセルが集められる．これらガーベジコレクションの処理の一部はベクトルパイプラインで処理される．本処理系は，強化した関数も含めて全てFORTRAN77/SXで記述している．

４．実行結果

四則演算においてベクトルパイプラインを利用した効果を示す一例として，図２にフォームの評価１回あたりの演算”＋”の実行に対してベクトル実行した場合とスカラ実行した場合の比較を示す．

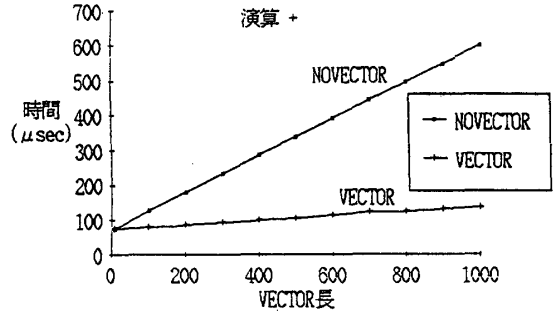

図２．ベクトル化の効果

また，Ｖｅｃｔｏ-Ｌｉｓｐの応用例の一つとして，我々の研究室で提案，実験中のＮｅｕｒｏ-Ｌｉｓｐ[2]環境での研究の一環として，階層型ニューラルネットワークモデルのシミュレーションをＶｅｃｔｏ-Ｌｉｓｐで記述し本処理系上で実行した．ネットワークは３層から成り，入力層，出力層のニューロン数はそれぞれ１０で，中間層のニューロン数を変え，学習及び出力に要する時間を測定した．学習の測定値は１０種類のデータについて学習をそれぞれ２００回行なった時間であり，出力計算の測定値は１０種類のデータを１回ずつ計１０回計算させた時間である．各々の結果を表１，表２に示す．

表１　結合数変化による学習時間

中間層ニューロン数	ベクトル実行 (sec/200回)	スカラ実行 (sec/200回)
１０	１．３７５×１０	１．５１７×１０
２０	１．４２９×１０	１．７０３×１０
３０	１．４８７×１０	１．８８３×１０
４０	１．５４０×１０	２．０６１×１０

表２　結合数変化による出力計算時間

中間層ニューロン数	ベクトル実行 (sec/10回)	スカラ実行 (sec/10回)
１０	1.03×10^{-2}	1.16×10^{-2}
２０	1.10×10^{-2}	1.36×10^{-2}
３０	1.17×10^{-2}	1.57×10^{-2}
４０	1.24×10^{-2}	1.77×10^{-2}

５．おわりに

本研究では，Ｌｉｓｐからスーパーコンピュータのベクトル演算機能を利用するために，Ｖｅｃｔｏ-Ｌｉｓｐインタプリタをスーパーコンピュータ上に試作した．また，Ｌｉｓｐ処理系の一部をベクトルパイプライン上で処理することをも試みた．Ｌｉｓｐの言語仕様を拡張することによって，ベクトルパイプラインを利用するための言語としての記述力の強化を行ない，応用プログラムにおいてベクトルパイプラインを使用した効果として，多量のデータを高速に処理する可能性を示すことができた．今後は，さらにベクトル化率を高めるためのプログラミング及び処理系を探究したい．

［参考文献］

(1)NECｽｰﾊﾟｰｺﾝﾋﾟｭｰﾀ SX-2/SX-1/SX-1E ｼｽﾃﾑ概説書 GAZ 01-3 NEC 日本電気株式会社

(2)安井 裕 他:Lisp並列処理マシン -EVLISマシン- とニューロエンジンの結合システム，情処第39回全大，3W-4(1989)．