1Q-2

スーパーコンピュータ用拡張記憶の拡張主記憶としての高度利用

大西重行　岡部寿男　津田孝夫
京都大学工学部情報工学教室

1．はじめに

今日、自然科学の各種の分野における大規模かつ複雑な科学技術計算を高速に実行するために、スーパーコンピュータが広く利用されている。これらの計算における膨大な数値データを高速に処理するためには、ベクトルプロセッサの演算処理性能の向上は勿論のこと、大容量でかつ高速な記憶装置の実現が不可欠である。しかしながら、処理対象となるデータ量が膨大になればなるほど、主記憶のメモリ空間のみでデータ処理を行うことは実質上困難になり、実記憶方式で対処するには限界がある。

スーパーコンピュータの中には、HITAC S-820シリーズのように従来の磁気ディスクの数百倍の転送速度をもつＧＢ級の高速半導体補助記憶装置（以下拡張記憶と呼ぶ）を備えているものがある。この拡張記憶装置は通常のFortranのREAD/WRITE文でアクセス可能であるため、従来ディスクとデータ転送を行っていた入出力処理を拡張記憶とのデータ転送に置き換えることができれば、プログラム全体の処理の高速化が期待される。

しかし、スーパーコンピュータにおける多くのプログラムは実記憶方式で書かれているため、これを拡張記憶とのデータ転送を行うように書き換えることはユーザには大きな負担となる。そこで我々は現在、与えられたFortarnプログラムを拡張記憶を主記憶の延長（拡張主記憶）として利用する形のプログラムへ変換するFortranプリプロセッサを開発中である。これによって、実際には主記憶と拡張記憶との間でデータ転送（ページ交換）を行いながらプログラムが実行されるわけであるが、ユーザからすれば拡張記憶の膨大なメモリ空間があたかも主記憶空間であり、巨大な配列データが常に主記憶上に存在していながらプログラムが進行しているかのように思える。このような変換を自動的に行うことにより、拡張記憶を仮想的な拡張主記憶としての高度利用が可能になると考えられる。

2．プリプロセッサによる実現

拡張記憶を仮想的な拡張主記憶として利用するためには、まず、主記憶に入りきらないような巨大な配列データをプログラム中から抽出して、拡張記憶へ割り付ける。そして、主記憶上にその配列データ用作業領域を確保する。つまり、巨大配列宣言を主記憶に確保できる作業領域の宣言に置き換える。

次に、拡張記憶に割り付けられた配列に対する参照を検出し、参照が行われる前に拡張記憶とのデータ転送命令（READ/WRITE文）を挿入する。そして、その参照を主記憶上に確保する作業領域に対する参照に置き換える。実際には、データ転送を行う必要があるか否かを判定した上で、必要な場合のみ転送を行うルーチン（データ転送ルーチン）を追加して、そのルーチンを呼び出すCALL文を挿入する。このルーチンは、次に参照される配列要素が主記憶上に存在するか否かを調べて、存在しなければまず、データ転送命令を実行して必要な配列要素を主記憶上に用意しておく。そして、元の配列要素の添字から作業領域における添字を計算しなおす。

このように我々は、プログラム変換をFortranのソースプログラムレベルにおいて行うアプローチを採っている。

3．拡張記憶とのデータ転送

拡張記憶とのデータ転送は、ある程度まとまったデータブロック（ページ）を単位に行われるのが効率的である。したがって、拡張記憶へ割り付けた配列データをデータブロックの大きさごとに分割して管理する必要がある。そして、主記憶上に確保する作業領域としては、使用できる主記憶容量に依存するが、データブロックの複数個を一度に格納できるだけの大きさを確保すればよい。よって、実際にユーザプログラムが実行される際には、このデータブロック単位の交換を行いながら、処理を進行させていくことになる。

また、各配列要素は参照される前に主記憶上に転送しておくためには、必要な配列要素がどのデータブロックに属しているのかを調べなければならない。２次元配列を例に取り上げると、複数の「列」を一つのデータブロックに対応させるから、各配列要素の２次元目の添字式（例えば、A(I,J)ならJ）の値が確定すれば、必要なデータブロックの番号がわかる。これは、Fortranにおいては配列データが列優先でメモリ上に割り付けられるからである。

参照がDOループ内で行われている場合を考えると、ループ内で参照される配列要素の添字式には各ループの制御変数がよく現れる。この場合、各

Advanced Use of Semiconductor Extended Storage as Extended Main Storage for Supercomputing

Shigeyuki OHNISHI, Yasuo OKABE and Takao TSUDA
Kyoto University

ループ制御変数の値は、それに対応するループに入った直後に確定するので、ループ内で必ずデータ転送ルーチンを呼び出すことになる。このルーチンの呼び出しはスカラ処理されるため、この呼び出しはできる限りループレベルの浅い所へ移動させることが望ましい。なぜならば、多重ループの最内側で１次元目の添字ではなく２次元目の添字が変化するような場合を考えると、最悪の場合最内側ループの繰り返しごとにデータ転送が必要となり、データ転送に要するCPUコストは非常に大きくなることが予想されるからである。

また、２．で述べたデータ転送ルーチンでは、２次元目の添字式の値からそれが含まれているデータブロックの番号を求めて、次に参照される配列要素が属しているデータブロックが作業領域上に存在するかどうかの判定を行っている。もし、作業領域のどのブロックにも必要とするデータブロックが存在しなければ、主記憶上のあるデータブロックを拡張記憶へ書き出して、必要なデータブロックを新たに読み込まなければならない。

４．おわりに

拡張記憶の拡張主記憶として利用に関する有効性を確かめるために、以上述べてきた手法にもとづき、典型的なＬＵ分解のプログラム[1]（1000×1000の行列の分解）を例に取り上げ、これを手作業で変換した場合にどの程度の性能が得られるか実験した。その測定結果を表1に示す。この結果より、ある程度仮想記憶方式を意識したプログラムは、拡張記憶を拡張主記憶として利用すると実用的なデータ転送コストで元のプログラムが実行されることがわかった。

現在、上述したFortranプリプロセッサのプロトタイプの開発を進めている。このプロトタイプでは、巨大配列データの抽出やデータ転送ルーチンの挿入する位置に関する情報をユーザがあらかじめコメントの形をしたユーザ指示行で記述できるようにしている。今後の課題としては、多くのプログラム例に対してこの手法の有効性を検証し、より複雑なプログラム変換が必要となる場合の処理に関する考察およびそれに対する対策を検討する必要がある。変換例を図1.1（変換前）、図1.2（変換後）に挙げておく。

謝辞

御討論頂いた本学大久保英嗣助教授、国枝義敏助手をはじめ津田研究室の諸氏に感謝致します。

参考文献

[1]森 正武：FORTARN77数値計算プログラミング，岩波コンピュータサイエンス，(岩波書店,1986)

表１　拡張記憶を用いたプログラムの実行結果

主記憶上の作業領域(KB)	500	1000	2000
CPU時間(sec)	13.32	11.35	10.43
VPU時間(sec)	3.80	3.77	3.77
転送時間（sec）	6.71	4.98	4.17
転送速度（MB/sec）	1154.6	1497.3	1622.3

（1000×1000の行列分解）
東京大学大型計算機センター HITAC S-820/80
1988年6月測定（拡張記憶：8 MB 使用）

```
      PROGRAM TEST
      PARAMETER (LEN = 10000)
      COMMON /BLOCK1/A(LEN,LEN)

      DO 40 K=1,N
        DO 10 J=K+1,N
          A(K,J)=A(K,J)/A(K,K)
10    CONTINUE
        DO 30 J=1,N
          DO 20 I=1,N
            IF (I.NE.K) THEN
              A(I,J)=A(I,J)-A(I,K)*A(K,J)
            ENDIF
20        CONTINUE
30      CONTINUE
40    CONTINUE
      END
```

図1.1　入力プログラム（変換前）

```
      PROGRAM TEST
      PARAMETER (LEN¥ = 10000)
      PARAMETER (WID¥ = 500, BLK¥ = 2)
      COMMON /BLOCK1/A¥(LEN¥,WID¥*BLK¥)
      COMMON /PARAM/LEN¥,WID¥
      COMMON /TABLE/ITBL¥(BLK¥)

      DO 40 K=1,N
        CALL TRANS(K,K¥,･･･)
        DO 10 J=K+1,N
          CALL TRANS(J,J¥,･･･)
          A¥(K,J¥)=A¥(K,J¥)/A¥(K,K¥)
10      CONTINUE
        DO 30 J=1,N
          CALL TRANS(J,J¥,･･･)
          DO 20 I=1,N
            IF (I.NE.K) THEN
              A¥(I,J¥)=A¥(I,J¥)-A¥(I,K¥)*A¥(K,J¥)
            ENDIF
20        CONTINUE
30      CONTINUE
40    CONTINUE
  END
```

- ITBL¥(BLK)
  元の配列データと作業領域上の
  データブロックの管理
- データ転送ルーチンの引数
  最後の次元の添字（KやJ）
  作業領域上の添字（K¥,J¥）

図1.2　出力プログラム（変換後）