ハイパフォーマンス
コンピューティング 77−21
（1999. 8. 3）

SR8000向け固有値計算の高速化手法

直野 健*、猪貝 光祥**、山本 有作*、平山 裕之**
*(株)日立製作所中央研究所
**(株)日立超LSIシステムズ

スーパーテクニカルサーバSR8000向けに固有値計算の高速化手法を提案し、評価した。中でも処理の重いハウスホルダ変換の部分に対し、ブロック化によるデータ分割、対角ブロック部の正方演算化、ループ融合による対演算ロード・ストア回数の低減による高速化を行った。これにより1ノードの性能では4000次元において約4.0Gflop/sと、同じピーク性能8Gflop/sのS-3800向け製品MATRIX/HAPの約2.9Gflop/sに比べ大きく性能を向上できた。

High Performance Eigenvalue Computation on the HITACHI SR8000

Ken Naono*, Mitsuyoshi Igai**, Yusaku Yamamoto*, Hiroyuki Hirayama**
*Central Research Laboratory, Hitachi, Ltd.
**Hitachi ULSI Systems Co., Ltd.

The methods of high performance eigenvalue computation on the HITACHI SR8000 are described and evaluated. To achieve high performance computation of the Householder transformation procedure, we adopted the blocked data distribution, the rectangular computation in the diagonal blocks, and the loop integration for reducing the number of load/store. On the 1 node of the SR8000, we achieved about 4.0 Gflop/s in the 4000-dim Householder transformations. This is much better than the 2.9 Gflop/s of the Matrix Library's on the HITACHI S-3800 which has the same peak performance with 1 node of the SR8000.

1. はじめに

固有値計算は他の線形計算に比べ処理量が重く、高速化が極めて重要である。対称密行列の固有値計算で処理が重いハウスホルダ変換について、近年、キャッシュマシン向けブロック化手法[1]が研究されてきた。最近 SMP マシン(RS/6000 SP)向けの研究[2]が報告されているが、十分な性能が引き出せていない。

本報告では、ハウスホルダ変換に対し、対角ブロックでの正方演算化と2つの行列ベクトル積の融合により、SR8000 のノード内8プロセッサ向けに効果的なループアンロールを行い、高速化した結果を報告する。

2. ハウスホルダ変換

ハウスホルダ変換は、標準固有値問題 Av=evをハウスホルダ法によって計算する場合に必須である。その計算量は N を行列 A の次元数として必ず$(4/3)N^3$となり、求める固有値、固有ベクトルの個数によらず常に計算負荷のかかる部分である。本報告でのハウスホルダ変換では行列Aの右上半分のデータを変換する。

ハウスホルダ変換はまず図 2-1 の最右側列のハウスホルダ変換を行い、最右側列の対角要素と副対角要素以外を 0 にする。そして図 2－1 と同様の処理を最右から2番目の列から左側の列へ順次行うことで三重対角行列にしていく。ここで処理量が大きいのは(4)の行列ベクトル積、(7)の RANK2 更新処理である。

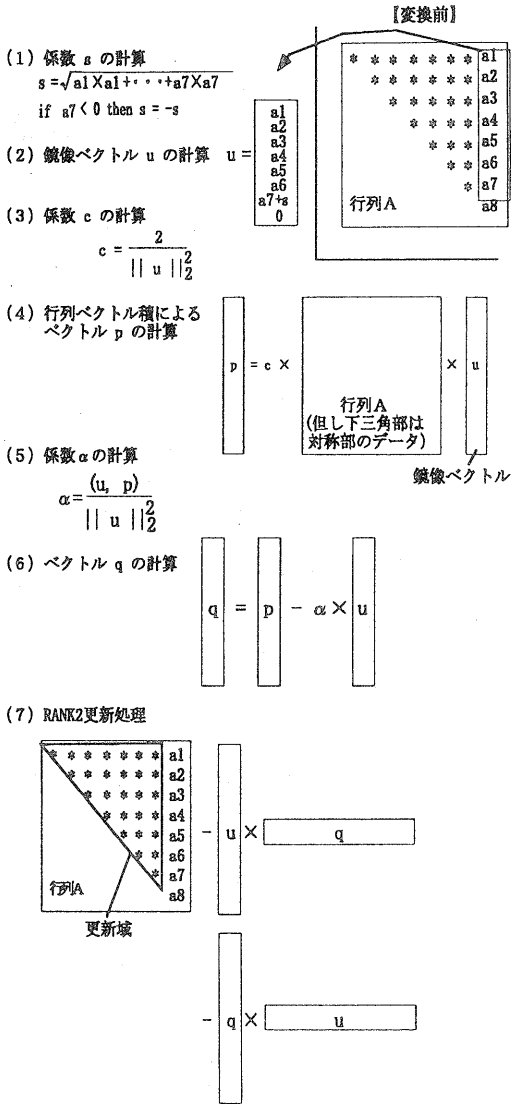


図 2-1 最右側列のハウスホルダ変換

3. 高速化手法

ハウスホルダ変換をSR8000向けに高速化するため、ブロック化、対角ブロックの正方演算化、2つの行列ベクトル積の融合を行った。

3.1 ブロック化ハウスホルダ変換

簡単のため図 2－2 に2列分のブロック化を施した例を示す。2列分のブロック化では、まず「部分ハウスホルダ変換」によって、u と q による RANK2 更新を、行列 A の最右側から2つ目の列(長方形のデータ更新領域)に対してのみ行う。そして、変換された行列 A の最右側から2つ目の列から生成される次のベクトル x と y とともに2列分の RANK2 更新を、行列 A の最右側から3つ目の列から最左側の列まで(三角形のデータ更新領域)行う。1列づつ RANK2 更新すると、1列のハウスホルダ変換の度に行列Aの要素をロード／ストアする必要があったのが、2列分まとめることでロード

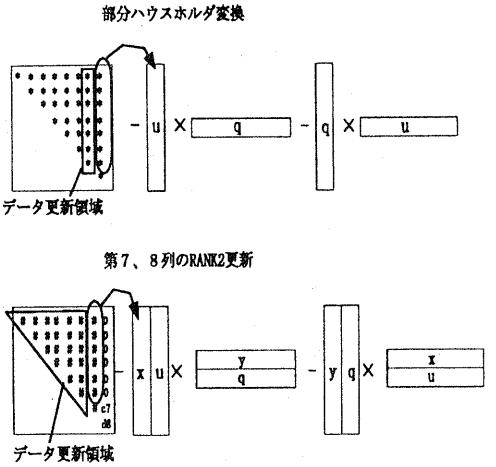


図 3-1 2列分のブロック化ハウスホルダ変換

／ストアを大幅に削減できる。

図 3-1 に示したブロック化は2列分であったが、実際にはこれを40列分ブロック化し高速化を行う。従って、図 2-1 の(1)から(6)までを40回行い、その回数毎のベクトルu、qを保持しU(N,40)、Q(N,40)とし、これを用いてRANK2 更新を行う。

複数ノードでは 40 を1ブロックサイズとして行列Aを 40×40 の正方行列毎に1つのノードに縦・横両方向にサイクリックに分散配置する。4ノードの場合を図 2-3 に示す。N-0 はノード0の担当領域である。

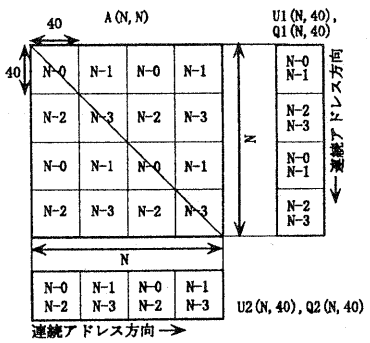


図 3-2 4ノードでの分散配置

U(N,40)と Q(N,40)は、図 3-2 のように U1(N,40)と Q1(N,40)という連続アドレス向けと、U2(N,40)、Q2(N,40)という(行列Aで見た)非連続アドレス向けの2種類を持つようにし、N について40づつ各ノードにサイクリックに分割する。また、U1(N,40)は列方向ノードグループ内で共有し、U2(N,40)は行方向ノードグループ内で共有するようにしておく。

本来は U1 のみであるところを U2 も用意するのは、図 3-3 の行列ベクトル積2に示すように、行列ベクトル積での、行列Aの対角要素から最終列と対応するベクトル要素との内積が同一ノード内で実行できるからである。前半の行列ベクトル積を Gemv-1、後半を Gemv-2 と呼ぶことにする。

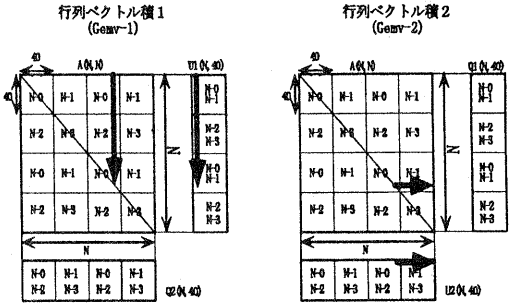


図 3-3 2つの行列ベクトル積

3.2 対角ブロックの正方演算化

行列 A (*,*)とベクトル U1(*,LPIV)(LPIV は固定)の積である Gemv-1 の上三角部分の処理は図 3-4 のようになる。

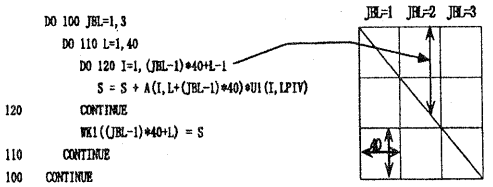


図 3-4 対角ブロックでの Gemv-1

このループはループ 120 の終値が 110 のインデックス Lに依存するため、ループ 110 でのアンロールができない。そのため、およそ 430Mflop/s とピーク性能の5%

程度となってしまう。

そこで、まずハウスホルダ変換の処理の前に、対角ブロック部分のみ上三角部分を下三角部分にコピーを行っておく。

次に、対角ブロック部分で Gemv-1 と Gemv-2 を併せてD-Block という1つの内積計算にする。図 3-5 の変更前のように、Gemv-1 は第 1 要素から対角要素の一つ前まで、Gemv-2 は対角要素から最終要素までだったのを、図 3-5 の変更後のようにそのまま全要素分を U1 と内積をとる。こうすると、Aのアクセスが全て連続になり、また列方向(J)についてアンロールが可能になる。

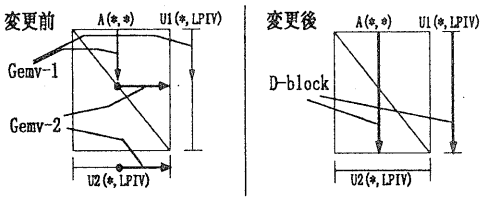


図 3-5 対角ブロックの下三角処理の正方化

RANK2 では対角ブロックでの更新処理を上半分だけではなく下半分も行う。これによって、演算量は増えるが、RANK2 での対角ブロックでの図 3-4 と同様なループが除外できる。

3.3 Gemv-1 と Gemv-2 のループ融合

図 3-6 に示すように、上記の正方化によって Gemv-1 と Gemv-2 でロードされる行列要素 A のデータは、アクセスする方向こそ違うが、全く同じになった。

そこで、2つの行列ベクトル積を融合することで行列データのロード数を削減する。

まず、ループ融合の前に Gemv-1 のアンロールを行う。A を列方向に 8 つアンロールを行うため、図 3-7 のように幅が8の帯ごとにサイクリックに各プロセッサに割り振る。この帯のインデックスがプログラム中のKKであり、これが do 300 において5×ブロック数

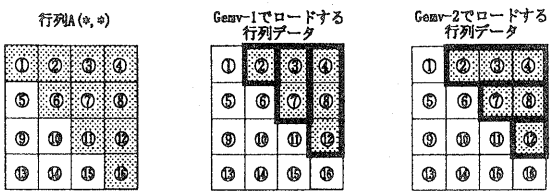


図 3-6 Gemv-1、Gemv-2 の両者でロードされる行列要素 A のブロック部分(1ノード)

(JBLMAX)まで動く。また、do 100 での関数 ILENGTH(JJ)は、第JJブロックでの行方向の長さを表している。

Gemv-2 では行列 A の列方向のインデックスとベクトルセット U2 で内積計算を行うが、A のロードする配列がプロセッサ毎に異なるため、ストアすべきデータを最後に和をとる必要がある。そこで WKP(N,0:7)という配列を用意し、プロセッサ-0 の結果は WKP(*,0)に、プロセッサ-1 の結果は WKP(*,1)に入る、などとして Gemv-2 の演算主体を行い、後で総和計算を行い WK2 を求める。Gemv-2 にこの実装を行ったプログラムを図 3-8 に示す。

なお、Gemv-2 のループでは、各プロセッサの終値保証の必要がないのため、INDEP(WKP) というディレクティブを用い不要なバリア同期を発生させないようにしてある。

4. 性能評価

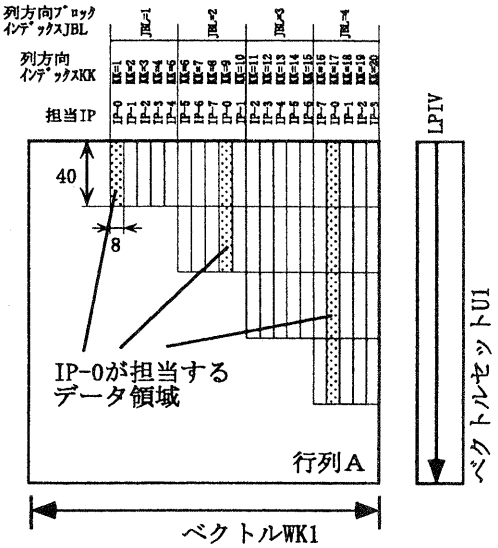
ループ融合の効果、各主要ループの性能評価、1ノード、複数ノードでの性能評価をそれぞれ示す。

4.1 ループ融合の効果

ループ融合を行わず各ループを高速化した結果と、ループ融合を行った結果を表 4.1 に示す。10Gflop 分の演算時間では Gemv-1 と Gemv-2 の融合化によって性能は約 1.5 倍になったことがわかる。

4.2 各主要ループの性能結果

各主要ループの性能を表 4.2 にまとめる。

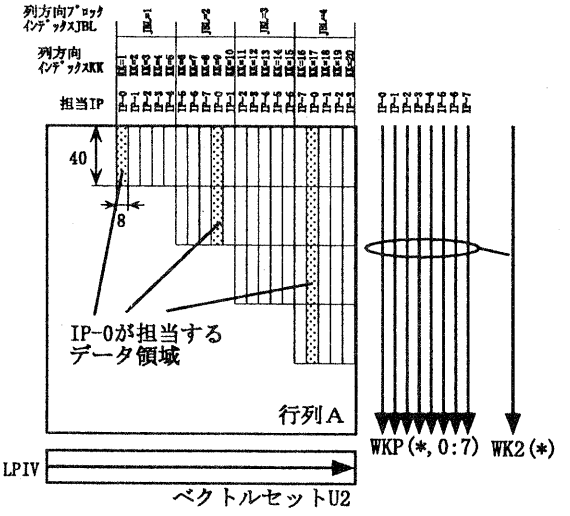



```
*POPTION CYCLIC(1)
      DO 300 KK = 1, 5*JBLMAX
         KKK = (KK-1)*8+40
         S1R = 0.0D0; S2R = 0.0D0;
         S3R = 0.0D0; S4R = 0.0D0;
         S5R = 0.0D0; S6R = 0.0D0;
         S7R = 0.0D0; S8R = 0.0D0;
         DO 100 I=1,ILENGTH((KK-1)/5+1)
            S1R=S1R+A(I,KKK+1)*U1(I,LPIV)
            S2R=S2R+A(I,KKK+2)*U1(I,LPIV)
            S3R=S3R+A(I,KKK+3)*U1(I,LPIV)
            S4R=S4R+A(I,KKK+4)*U1(I,LPIV)
            S5R=S5R+A(I,KKK+5)*U1(I,LPIV)
            S6R=S6R+A(I,KKK+6)*U1(I,LPIV)
            S7R=S7R+A(I,KKK+7)*U1(I,LPIV)
            S8R=S8R+A(I,KKK+8)*U1(I,LPIV)
  100    CONTINUE
         WK1(KKK+1) = S1R; WK1(KKK+2) = S2R
         WK1(KKK+3) = S3R; WK1(KKK+4) = S4R
         WK1(KKK+5) = S5R; WK1(KKK+6) = S6R
         WK1(KKK+7) = S7R; WK1(KKK+8) = S8R
  300 CONTINUE
```

図 3-7 Gemv-1 部分の 8 段アンロール
(1ノード)

表 4.1 Gemv-1と Gemv-2の融合前と後の
2500 次元における実測性能(Gflop/s)、
並びに 10Gflop 演算時間の比較

融合	ループ	演算量	Gf/s	10Gf
しない	Gemv-1	$N^3/3$	6.2	5.6 s
	Gemv-2	$N^3/3$	2.5	
する	Gemv-1&2	$2N^3/3$	5.3	3.7 s




```
*POPTION INDEP(WKP)
      DO 300 KK = 1, JBLMAX*5
         IP = MOD(KK-1,8); KKK = (KK-1)*8+40
         DO 100 I = 1, ILENGTH((KK-1)/5+1)
            WKP(I,IP) = WKP(I,IP)
              + A(I,KKK+1)*U2(KKK+1, LPIV)
              + A(I,KKK+2)*U2(KKK+2, LPIV)
              + A(I,KKK+3)*U2(KKK+3, LPIV)
              + A(I,KKK+4)*U2(KKK+4, LPIV)
              + A(I,KKK+5)*U2(KKK+5, LPIV)
              + A(I,KKK+6)*U2(KKK+6, LPIV)
              + A(I,KKK+7)*U2(KKK+7, LPIV)
              + A(I,KKK+8)*U2(KKK+8, LPIV)
  100    CONTINUE
  300 CONTINUE

      DO I = 1, IBLMAX*40
         WK2(I) = WKP(I,0)+WKP(I,1)+,…,+WKP(I,7)
      ENDDO
```

図 3-8 Gemv-1と融合させる Gemv-2(1ノード)

表 4.2 各主要ループの
演算量、Byte/Flop 値、実測性能(Gflop/s)

ループ名	演算量	B/F 値	実効性能
RANK2	$2N^3/3$	2.0	7.13 Gf/s
Gemv-1&2	$2N^3/3$	3.5	5.34 Gf/s
Dot	$20N^2$	5.0	2.09 Gf/s
Belt-1&2	$40N^2$	3.5	4.44 Gf/s
D-block	$40N^2$	4.5	1.12 Gf/s

表での Dot は部分ハウスホルダ変換、Belt-1 と Belt-2 は、行列 A の 40 列分に満たない右側ブロックの部分での Gemv-1 と Gemv-2 であり、Gemv-1&2 と同様

にループ融合を行った。D-Block は 8 段のアンロールを行った。

4.3 ハウスホルダ変換の性能

SR8000 の1ノード上でのハウスホルダ変換の実行時間などを表 4.3 にまとめる。1000 次元や 2000 次元ではやや性能が悪く、次元数に対する性能の立ち上がりが良好とは言えないが、12000 次元ではおよそ 5.5Gflop/s、ピーク性能のおよそ70%を達成した。また、1ノードと同じピーク性能の S-3800 との比較では、1000次元ではやや劣るものの、4000次元では1．3倍以上の性能を達成した。

表 4.3 SR8000 の 1 ノードでのハウスホルダ変換の実行時間、実行性能、対ピーク比、並びに S-3800 での実効性能との比較

次元数	時間 sec	性能 Gflop/s	対ピーク性能比	S-3800 Gflop/s
1000	1.09	1.23	15.4 %	1.58
2000	4.10	2.60	32.5 %	2.27
4000	21.27	4.02	50.2 %	2.94
8000	137.94	4.95	61.9 %	---
12000	416.36	5.54	69.2 %	---

また、表 4.4 に1ノード当り 4000 次元の行列サイズでの1，4，16ノードの実行時間などを示す。

表 4.4 SR8000 の1，4，16ノードでのハウスホルダ変換の実行時間、実行性能、対ピーク比

ノード数	次元数	時間 sec	性能 Gflop/s	対ピーク性能比
1	4000	21.27	4.02	50.2 %
4	8000	45.41	15.04	47.0 %
16	16000	94.55	57.77	45.1 %

複数ノードにおいても良好な性能を発揮していることがわかる。

5. 結論

スーパーテクニカルサーバ SR8000 向けに固有値計算の高速化手法を処理の重いハウスホルダ変換について検討した。高速化の手法は、ノード内向けチューニングとして、ブロック化、対角ブロック部の正方演算化、2つの行列ベクトル積計算におけるループ融合、を行った。これらによって行列 A のロード／ストア回数の削減、アンローリングを可能にした。

これらの高速化により、1ノードの性能では 4000 次元において約 4.02Gflop/s と、同じピーク性能の S-3800 向け製品 MATRIX/HAP の約 2.94Gflop/s に比べ大きく性能を向上できた。12000 次元においては約 5.54Gflop/s とピーク性能の約 70% という高い性能を達成できた。また、複数ノードの性能においても 16 ノード 16000 次元において 57.77 Gflop/s という高い性能を達成することができた。

[参考文献]

[1] J.J.Dongarra and R.A. van de Geijn: Reduction to condensed form for the Eigenvalue problem on distributed architectures, Parallel Computing Vol.18, No.9, 1992.

[2] S.A. Salvini and L.S. Mulholland、The NAG Fortran SMP Library、9th SIAM Conference Parallel Processing for Scientific Computing 1999.