平成 12 年度 科学技術計算分科会 第 1 回会合資料
平成 12 年 8 月 12 日

# NAS Parallel 1.0 を用いた VPP5000 性能評価

九州大学応用力学研究所
矢木雅敏
yagi@riam.kyushu-u.ac.jp
http://www.riam.kyushu-u.ac.jp/

## １． はじめに

九州大学応用力学研究所（以下応力研）では平成１２年３月に汎用計算機システムを更新し、４月より正式運用を開始した。今回、ベクトル並列演算サーバとして VPP5000 2PE(19.2GFLOPS,メモリ 16GB)を、スカラー並列演算サーバとしてメモリ・チャンネル接続の Compaq AlphaServer ES40 3 台(16GFLOPS,メモリ 9GB)を導入した。ベクトル並列演算サーバの選定にあたっては NAS Parallel 1.0 を用いた性能評価を行ったのでその結果を紹介する。また既存のスペクトルコードを用いた性能評価も合わせて紹介し、問題点を検討する。

## ２． システム構成と運用例

図１に汎用計算機システム構成図を示す。ユーザーは PC(Linux2.2.12+nfsv3 パッチ/Windows98 の multi boot system)あるいはアプリケーションサーバ上でプログラム開発を行い、NQS により PC あるいはアプリケーションサーバ上から VPP5000 及び ES40 へジョブを投入する。ユーザのホーム領域は RAID から NFS により PC 上にマウントされている。ファイルの書き込み速度を向上させるため nfsv3 を採用した（表 2-1 参照）。また PC 上には Linux で動作する Fortran & C Package（富士通製）及び VPP5000 用のクロスコンパイラー（九大応力研仕様　表 2-2 参照）が搭載してある。Linux 側の fortran コンパイラーは VPP5000 が生成するバイナリーファイル（big endian）に対し実行オプションにより互換性を保っている（frt –Wl,-T）。またネットワークライセンス形態で mathematica,matlab の使用が可能である。

**表2-1 Solaris2.6をNFSサーバとしてnfsマウント（オプション　hard,intr ）したディスク上に書き込み。dd if=/dev/zero of=testfile bs=16k count=4096とし、64MBの書き込みテスト。**

| クライアント | プロトコル | 時間（分:秒） |
|---|---|---|
| Linux2.2.5(Laser5 Linux6.0 標準) | NFSv2 | 7:30 |
| Linux2.2.12 | NFSv2 | 7:50 |
| Linux2.2.12+nfsv2 パッチ | NFSv2 | 5:00 |
| Linux2.2.12+nfsv3 パッチ | NFSv3 | 1:30 |
| Linux2.0.36(Redhat Linux5.2 標準) | NFSv2 | 29:30 |
| FreeBSD-4 | NFSv2 | 4:30 |
| FreeBSD-4 | NFSv3 | 1:50 |

**表 2-2 Fortran コンパイラ性能比較**

| | Linux Cross<br>(Pentinum III 500MHz) | VPP Cross<br>(Ultra Sparc II 295MHz) | VPP5000 |
|---|---|---|---|
| Linpack100 元ﾌﾟﾛｸﾞﾗﾑ | 1.58sec | 2.46sec | 2.52 sec |
| ﾘﾊﾞﾓｱ 14 ﾙｰﾌﾟﾌﾟﾛｸﾞﾗﾑ | 1.57sec | 2.32sec | 2.34 sec |
| ANL ﾍﾞｸﾄﾙ化ｺﾝﾃｽﾄﾌﾟﾛｸﾞﾗﾑ | 16.67sec | 25.43sec | 24.48sec |

表 2-3 に現在運用している VPP5000 のジョブクラスを示す。ここで C0,C1,P は夜間ジョブである。運用は 8:00-0:00 までは A0,A1,B0,B1 が走りジョブスワップにより 0:00-8:00 までは C0,C1,P が優先して走るように設定してある。LinuxPC 上に提供された jstat コマンド (usage: jstat [L|A0|A1|B0|B1|C0|C1|P|D|E|S|M|vpp|es] )により VPP5000,ES40 のジョブの実行状況を確認できるようになっている（表 A-1)。　現在運用している CPU 課金は VPP5000 1PE 0.5 円/sec, 2PE 0.75 円/sec である（ただし、各研究室に対し、課金が 100 万円を越えた場合、それ以上の課金はその研究室から徴収していない)。表 2-4 に平成１２年度４月から６月までのクラスごとの使用状況を示す。６月のベクトルとスカラーの CPU 合計は 1448.17H であり 24Hx30(day)x2(PE)=1440H を目安にすると３ヶ月にして使用率はほぼ100%に達していることがわかる。

**表 2-3 VPP5000 のジョブクラス**

| パイプキュー名<br>（LinuxPC） | バッチキュー名<br>（VPP5000） | 多重度 |
|---|---|---|
| L | L (Compile & Linkage 専用） | 4 多重 P-PE |
| A0 | A0 (0.5GB,15 分) | 1 多重 P-PE |
| A1 | A1 (0.5GBx2PE,15 分) | 1 多重 P-PE,S-PE |
| B0 | B0 (1.5GB,8 時間) | 4 多重 P-PE |
| B1 | B1 (1.5GB,24 時間) | 4 多重 S-PE |
| C0 | C0 (3.0GB,4 時間） | 1 多重 P-PE |
| C1 | C1 (3.0GB,4 時間） | 1 多重 S-PE |
| P | P (4.0GB×2PE,4 時間） | 1 多重 P-PE,S-PE |

**表 2-4　各クラスの使用状況**

| クラス | 4月 | | | 5月 | | | 6月 | | |
|---|---|---|---|---|---|---|---|---|---|
| | ジョブ件数 | ﾍﾞｸﾄﾙCPU(H) | ｽｶﾗCPU(H) | ジョブ件数 | ﾍﾞｸﾄﾙCPU(H) | ｽｶﾗCPU(H) | ジョブ件数 | ﾍﾞｸﾄﾙCPU(H) | ｽｶﾗCPU(H) |
| A0 | 56 | 1.05 | 1.01 | 165 | 0.78 | 0.59 | 308 | 8.36 | 1.53 |
| A1 | 39 | 2.14 | 0.37 | 28 | 0.05 | 0.23 | 115 | 2.42 | 1.02 |
| B0 | 196 | 283.68 | 25.31 | 399 | 592.12 | 61.69 | 206 | 600.46 | 93.01 |
| B1 | 169 | 343.13 | 31.4 | 452 | 612.55 | 54.66 | 76 | 669.53 | 53.28 |
| C0 | 17 | 19.47 | 11.95 | 21 | 7.31 | 2.27 | 16 | 7.13 | 2.51 |
| C1 | 8 | 12.66 | 4.95 | 1 | 0.03 | 0.01 | 4 | 7.67 | 1.21 |
| L | 92 | 0 | 0.17 | 312 | 0 | 1.11 | 26 | 0 | 0.04 |
| P | 13 | 0.31 | 0.85 | 0 | 0 | 0 | 0 | 0 | 0 |
| 合計 | 590 | 662.44 | 76.01 | 1378 | 1212.84 | 120.56 | 751 | 1295.57 | 152.6 |

## 3．　NAS Parallel Benchmark (Version 1.0) によるパフォーマンス評価

Numerical Aerodynamic Simulation(NAS)プログラムは NASA Ames Research Centerで開発されたベンチマークプログラムであり、さまざまな並列スーパコンピュータシステムの性能評価のために使用されている。現在 http://www.nas.nasa.gov/Software/NPB/においてNPB2.3が提供されているが、最新バージョンはスカラー並列用にチューニングされており、ベクトル長が十分とれないこともあって、ベクトル並列計算機システムの性能を評価する場合は通常 NPB 1.0 を用いる。NPB1.0 は８種類のベンチマークプログラムから構成されており、５種類の kernel と３種類の simulated computational fluid dynamics(CFD)アプリケーションからなる。さらに問題の大きさに応じて Class A, ClassB, Class C に区分される。表 3-1 にそれぞれのクラスの問題の大きさを示す（NAS Parallel Benchmarks (Version 1.0) Results 11-96, S. Saini and D. H. Bailey, Report NAS-96-18, Nov. 1996, Table 1 より抜粋）。

**表 3-1　NPB における各クラスの問題の大きさ**

| Benchmark Name | Abb. | Class A Nominal Size (Memory Size) | Class B Nominal Size (Memory Size) | Class C Nominal Size |
|---|---|---|---|---|
| Embarrassingly Parallel | EP | $2^{28}$ (1Mw) | $2^{30}$ (18Mw) | $2^{32}$ |
| Multigrid | MG | $256^3$ (57Mw) | $256^3$ (59Mw) | $512^3$ |
| Conjugate Gradient | CG | $14x10^3$ (10Mw) | $75x10^3$ (97Mw) | $1.5x10^5$ |
| 3-D FFT PDE | FT | $256^2$x128 (59Mw) | $512x256^2$ (162Mw) | $512^3$ |
| Integer Sort | IS | $2^{23}x2^{19}$ (26Mw) | $2^{25}x2^{21}$ (114Mw) | $2^{27}$ |
| LU Simulated CFD Appl. | LU | $64^3$ (30Mw) | $102^3$ (122Mw) | $162^3$ |
| SP Simulated CFD Appl. | SP | $64^3$ (6Mw) | $102^3$ (22Mw) | $162^3$ |
| BT Simulated CFD Appl. | BT | $64^3$ (24Mw) | $102^3$ (96Mw) | $162^3$ |

ベンチマークはクラスCの問題の大きさを選び、kernelとしてMG, CG, FTを、CFDとしてLU, SP, BTを使用した。 ここでMGは3-D Poisson PDEを解くためのmultigrid kernelであり、CGは大きなスパース対称正定値行列の最小固有値の近似解を計算するための勾配共役法、FTは3-D PDEを解くためのFast Fourier Transform、LUにおいては正規スパースブロック 5x5 下、上三角行列を解くために lower-upper diagonal factorization ではなく symmetric successive over-relaxation(SSOR)スキームを用いている。SPは scalar pentadiagonal solverであり、BTは block tridiagonal solver である。 Kernelの詳細な説明は THE NAS PARALLEL BENCHMARKS, D. Bailey, E. Barszcz, J. Barton, et al., RNR Thechnical Report RNR-94-007, March 1994に記述されているのでそちらを参照していただきたい。最近ではCFDとして新しい計算手法も提案されておりsp,btのような手法は古典的になりつつある。

表3-2にベンチマーク環境、表3-3にコンパイルオプション、表3-4にコンパイルオプションの説明、表3-5にクラスCに対するベンチマークテストの結果（平成１１年８月１７日）を示す。 N社の共有型ベクトル並列マシンでもほぼ同等の性能が得られている。これは1CPUあたりの理論性能はVPP5000の方がN社のマシンより20%程度すぐれていたが通信においては共有型の方が有利であり総合性能として見るとほぼ同程度ということである。しかし共有型マシンでもマルチノード間で通信を行う場合は通信のパフォーマンスは当然落ちる。（この結果はNASAにsubmitしたものではないので公式なデータとして使用できないことに注意してほしい。）。

**表3-2　ベンチマーク実施環境**

| | 富士通 |
|---|---|
| ハード | VPP5000/2(2PE)<br>分散型ベクトル並列計算機<br>理論演算性能 9.6GFLOPSx2<br>SDRAMメモリ 16GB(8GB/PE) |
| OS | UXP/V V20L10 |
| コンパイラ | UXP/V Fortran V20L10<br>UXP/V Fortran/VPP V20L10 |
| ライブラリ | UXP/V SSL2/VP V20L10 |

**表3-3　コンパイルオプション**

| | UXP/V Fortran/VPP V20L10 |
|---|---|
| MG | -Wx –Of,-x –Wv,-ilfunc –Wv,-Gf –Wv,-m3 –Ps –Pt -Z |
| CG | -Wx –Wv,-md –Ab –Sw –Oe –Ps -Z |
| FT | -Wx –O5 –Ne –Wv,-ilfunc –Wv,-P255 –Wv,-m3 –Ps –Pt -Z |
| LU | -Wx –O5 –Wv,-Gf –Wv,-P255 –Wv,-ilfunc –Wv,-preload -Ps |
| SP | -Wx –O5 –Wv,-r163 –Wv,-P100 –Ad –NI –Ps –Pd -Z |
| BT | -Wx –Of –Wv,-P255 –Ad –Os,-x |

**表 3-4　オプションの説明**

| UXP/V Fortran/VPP V20L10 | |
|---|---|
| **オプション** | **説明** |
| -Wx | VPP機能を有効にするXOCL行をVPP用拡張最適化制御行と認識し、VPP Fortran向けのオブジェクトを生成する。 |
| -Os,-x | 最適化の内容として「演算の評価方法を変更する最適化機能を更に強化した最適化」を追加実施する。 |
| -Of,-x | 最適化オプション-Of に追加して、「演算の評価方法を変更する最適化機能を更に強化した最適化」を追加実施する。 |
| -Oe | 標準的な最適化を実施する。演算評価方法の変更、不変式の先行評価や総和演算のベクトル化などが実施される。 |
| -Of | 最適化オプション-Oeで行われる最適化に加え、同一ファイル内のインライン展開、多重DOループの構成変更、最適化の繰り返し実行などが実施される。 |
| -O5 | 最適化レベルの指定-Ofで行われる最適化に加え、強化した演算評価方法の変更などが実施される。 |
| -Wv,-Gf | ベクトルレジスタのグローバル化可能な配列要素を全てグローバル化する。 |
| -Wv,-P<num1> | ベクトル命令スケジューリングの範囲を<num1>命令単位とする。 |
| -Wv,-ilfunc | ベクトル化された組み込み関数のインライン展開を行う。 |
| -Wv,-preload | ベクトルデータのプリロードを行う。 |
| -Wv,-r<num2> | DOループの繰り返し回数の最大値が<num2>であることを指示する。 |
| -Wv,-m3 | ベクトル化できなっかた原因およびベクトル化の状況を通知するメッセージを出力する。 |
| -Wv,-md | -Wv,-m3と等価である。 |
| -Ab | 全ての仮引数を位置どりにする。 |
| -Ad | 単精度から倍精度に精度拡張する。 |
| -Ne | 実行文数が３０以下の利用者定義の外部手続きをその引用箇所にインライン展開する。 |
| -NI | 利用者定義の外部手続きをその引用箇所にインライン展開する最適化を抑止する。 |
| -Ps | 原始プログラムリストを出力する。 |
| -Pd | INCLUDE行で取り込まれた原始プログラムリストを出力する。 |
| -Sw | Iレベルのメッセージの出力を抑止する。 |
| -Z<filename> | 指定されたファイル名<filename>に翻訳情報を出力する。 |

**表 3-5　クラス C に対するベンチマークテストの結果**

| ベンチマーク名 | 実行時間 VPP5000 |
|---|---|
| MG | 19.48sec |
| CG | 38.57sec |
| FT | 39.02sec |
| LU | 183.69sec |
| SP | 157.76sec |
| BT | 218.67sec |

## ４．スペクトルコードによるパフォーマンス評価および今後の課題

次にスペクトルコードを用いたパフォーマンス評価を行った。このコードでは３重対角行列と５重対角行列で構成される連立一次元方程式を解いている。すなわち NPB の SP, BT に対応する CFD を必要とする。Sampler による解析結果を表 A-2 に示す。ここで３重対角行列の solver は RECUR であり、５重対角行列の solver は RECUR2 である。これらのドライバーがこのコードのコストを決めているがあまりベクトル化されていない（この場合をケース(a)とする）。表 A-3 に RECUR のソースコードを示す。RECUR 及び RECUR2 では漸化式を用いているためにベクトル化ができていない。コードサイズが大きくなるとそれでも他の部分のベクトル化の効果により若干速度の向上が見られる。問題の大きさを 128x288,128x972,128x1088,128x2048 と変化させたときの ES40 1CPU (f90)と VPP5000 1PE (frt)との比較を表 4-1 に示す（時間ステップ 2 回）。ここで 128x2048 の場合のコードの大きさは約 310MB 程度であり、128x6144 は約 1.33GB 程度である。円柱の体系を考え、r 方向は有限差分でθ及び z 方向はフーリエ分解しているが、128 は r 方向のメッシュサイズ 2048,6144 はレゾナンスモードの数を表している。ただしレゾナンスモードの数が３倍になってもコードの大きさは単純に３倍にはならない。CPU 時間はモード数に対して線形ではなく mode 数の 2-2.5 程度のべきで増加している。128x3788 の場合を比較すると VPP5000 の方が約 4 倍程度速いことがわかる。VPP5000 のスカラ性能は Alpha 667MHz(1.33GFLOPS) の約 50%強なのでこの差はベクトル化によるものと考えられるだろう。

さらにこれらを解くドライバーをガウスの掃き出し法へ変えたものをケース(b)とする。ソースコードは Computational Techniques for Fluid Dynamics Volume 1, Fundamental and General Techniques, Second Edition, C. A. J. Fletcher, Springer P.185,186 のサブルーチン BANFAC, BANSOL を修正した。コードの一部を表 A-4 に示す。これらのサブルーチンもベクトル化されていない。また参考のため LAPACK を使用し５重対角行列を LU 分解で解き、３重対角行列をガウスの掃き出し法で解くドライバーZGBSV, ZGTSV を使用した（ケース(c)）。ここでは問題の大きさを 128x972 とした。表 4-1 の場合と入力パラメータが異なるので実行時間は一致しない（時間ステップ 10 回）。むしろそれぞれのドライバーのパフォーマンスに注目してほしい。表 4-2 にベンチマークの結果を示す。LAPACK(数値計算ライブラリ)は期待したほど性能がでていない。LAPACK では一般の行列の演算に関してはベクトル化の効果が期待できるが３重対角とか５重対角のような行列に対しては逆にオーバーヘッドが現れ遅くなってしまうので注意が必要である。

今後の課題として富士通へ要望したいことは SSL II の拡張機能でサポートされている線形計算のためのサブルーチンは実行列のみであり、複素行列には対応していないのでそれを拡張すること、NPB で使用されている基本 kernel 及び CFD は頻繁に使用されるものであり、部分的には SSL II でサポートされているものの完全とは言い難い。特にここで問題となった bt, sp に対応するソルバーの高速化は必要であり（このコードのパフォーマンスが最低でも ES40 の 10 倍程度はほしい）、ぜひ SSL II でサポートしていただきたい。

**表 4-1　問題のサイズを変化させた場合のベンチマークの結果**

| 問題のサイズ | ES40 | VPP5000 |
|---|---|---|
| 128x288 | 4.7sec | 1.60sec |
| 128x972 | 65sec | 15sec |
| 128x1088 | 78.7sec | 15.52sec |
| 128x2948 | 368.1sec | 96.24sec |
| 128x3788 | 1900.5sec | 464.75sec |

**表 4-2　さまざまなドライバに対するベンチマークの結果（128x972）**

| | ES40 | VPP5000 |
|---|---|---|
| case (a) | 150.1sec | 31.06sec |
| case (b) | 151.5sec | 33.02sec |
| case (c) | 156.2sec | 55.77sec |

**表 A-1 jstat コマンドの出力結果**

[yagi@riam202 ~]$ jstat vpp

L@vpp;  type=BATCH;  [ENABLED, INACTIVE];  pri=40

  0 exit;  0 run;  0 queued;  0 wait;  0 hold;  0 arrive;

A0@vpp;  type=BATCH;  [ENABLED, INACTIVE];  pri=40

  0 exit;  0 run;  0 queued;  0 wait;  0 hold;  0 arrive;

A1@vpp;  type=BATCH;  [ENABLED, INACTIVE];  pri=40

  0 exit;  0 run;  0 queued;  0 wait;  0 hold;  0 arrive;

B0@vpp;  type=BATCH;  [ENABLED, RUNNING];  pri=40

  0 exit;  3 run;  3 queued;  0 wait;  0 hold;  0 arrive;

<3 requests RUNNING>

<3 requests QUEUED>

B1@vpp;  type=BATCH;  [ENABLED, RUNNING];  pri=40

  0 exit;  3 run;  4 queued;  0 wait;  0 hold;  0 arrive;

<3 requests RUNNING>

<4 requests QUEUED>

C0@vpp;  type=BATCH;  [ENABLED, RUNNING];  pri=30

  0 exit;  1 run;  0 queued;  0 wait;  0 hold;  0 arrive;

<1 request RUNNING>

C1@vpp;  type=BATCH;  [ENABLED, INACTIVE];  pri=30

  0 exit;  0 run;  0 queued;  0 wait;  0 hold;  0 arrive;

P@vpp;  type=BATCH;  [ENABLED, INACTIVE];  pri=30

  0 exit;  0 run;  0 queued;  0 wait;  0 hold;  0 arrive;

**表 A-2　Samplerによる解析結果**

```
Status                    : Serial
 Number of Processors     : 1

 Type                     : cpu
 Interval (msec)          : 20

 Synthesis Information
    Count|   Percent(Accum)|    VL| V_Hit(%)| Name
      105|      43.9( 43.9)|     -|     37.1|  RECUR_
       70|      29.3( 73.2)|     -|     24.3|  RECUR2_
       33|      13.8( 87.0)|   319|     42.4|  TMPUS_
       25|      10.5( 97.5)|    31|     76.0|  TMRHS_
        5|       2.1( 99.6)|    31|     60.0|  TMDRV_
        1|       0.4(100.0)|     -|      0.0|  FUNC_

      239|                  |   143|         |  TOTAL
```

**表 A-3　サブルーチン RECUR のソースコード**

```
      C=====================================================================
       2            SUBROUTINE RECUR(XX,AA,BB,CC,DD,NN,N0,N1)
       3       C=====================================================================
       4       C                                       2000/7/11
       5       C                                       RIAM, KYUSHU UNIVERSITY
       6       C                                       MASATOSHI YAGI
       7       C
       8       C     AA(I)*XX(N-1)+BB(N)*XX(N)+CC(N)*XX(N+1)=DD(N)     N=0,---,NN
       9       C             N0=0 : XX( 0)=A0 ;  N0=1 : D(XX( 0))/DY=AN.
      10       C             N1=0 : XX(NN)=B0 ;  N1=1 : D(XX(NN))/DY=BN.
      11       C
      12       C             N0=0 XX(0)  = A0
      13       C             N1=0 XX(NN) = B0
      14       C             N0=1 XX(0)  = DR*AN
      15       C             N1=1 XX(NN) = DR*BN
      16       C=====================================================================
```

```
17            IMPLICIT REAL*8(A-H,O-Z)
18            PARAMETER( IDM = 200 )
19            COMPLEX*16, DIMENSION(0:IDM) :: XX,AA,BB,CC,DD,EE,FF
20            COMPLEX*16 GG
21      C-----------------------------------------------------------------------
22            NM=NN-1
23      C-----
24            SELECT CASE (N1)
25            CASE(0) ! N1=0
26            EE(NM)=0.0D0
27            FF(NM)=XX(NN)
28            CASE(1) ! N1=1
29            GG    =1.0D0/(3.0D0*AA(NM)-CC(NM))
30            EE(NM)=(4.0D0*AA(NM)+BB(NM))*GG
31            FF(NM)= - (DD(NM)-2.0D0*XX(NN)*AA(NM))*GG
32      C##      EE(NM)=(4.0D0*AA(NM)+BB(NM))/(3.0D0*AA(NM)-CC(NM))
33      C##      FF(NM)= - (DD(NM)-2.0D0*XX(NN)*AA(NM))/(3.0D0*AA(NM)-CC(NM))
34            END SELECT
35      C
36  s2        DO N=NM-1,0,-1  ! EE(NN-2),....EE(0)
37  s2        GG=1.0D0/(BB(N+1)+CC(N+1)*EE(N+1))
38  s2        EE(N)=-AA(N+1)*GG
39  s2        FF(N)=(DD(N+1)-CC(N+1)*FF(N+1))*GG
40      C##      EE(N)=-AA(N+1)/(BB(N+1)+CC(N+1)*EE(N+1))
41      C##      FF(N)=(DD(N+1)-CC(N+1)*FF(N+1))/(BB(N+1)+CC(N+1)*EE(N+1))
42  s2        END DO
43      C
44            IF(N0 .EQ. 1) THEN
45            XX(0)=((4.-EE(1))*FF(0)-FF(1))/(3.-(4.-EE(1))*EE(0))
46           & -2.0D0*XX(0)/(3.0D0-(4.0D0-EE(1))*EE(0))
47            END IF
48      C
49  s6        DO  N=1,NN ! XX(1),...XX(NN)
50  s6        XX(N)=EE(N-1)*XX(N-1)+FF(N-1)
51  s6        END DO
52      C
```

```
      53            RETURN
      54            END

 Vectorization messages: program name(RECUR)
  jpc2217i-i   "zztext.f", line 37 - 38: EE is not vectorized since recursive reference takes place.
  jpc2217i-i   "zztext.f", line 39: FF is not vectorized since recursive reference takes place.
  jpc2217i-i   "zztext.f", line 50: XX is not vectorized since recursive reference takes place.
tmrhs.f:
 UXP/V  Fortran V20L20  Mon Aug 14 10:39:05 2000
```

**表 A-4　サブルーチン banfac,bansol のソースコード**

```
      1            subroutine banfac(b,n,int)
      2            complex*16 b(5,n)
      3      C##
      4            select case(int)
      5            case(1)
      6      C
      7      C     int = 1, linear elements = tridiagonal system
      8      C
      9  s4        do j = 2, n
     10  m4        b(2, j) = b(2, j) / b(3, j-1)
     11  s4        b(3, j) = b(3, j) - b(2,j)*b(4,j-1)
     12  v4        end do
     13      C
     14      C##
     15            case(2)
                     .
                     .
                     .
     40            end select
     41            return
     42            end

     Vectorization messages: program name(banfac)
  jpc2306i-i   "banfac.f", line 9: This DO loop is not vectorized since partial vectorization overhead
is too large.
```

```
  jpc2217i-i   "banfac.f", line 10 - 11: b is not vectorized since recursive reference takes place.
  jpc2217i-i   "banfac.f", line 35 - 37: b is not vectorized since recursive reference takes place.

       1            subroutine bansol(r,x,b,n,int)
       2            complex*16 r(n), x(n), b(5,n)
       3            select case(int)
       4            case(1)
       5       C##
       6       C##     int = 1,   tridiagonal system
       7       C##
       8  s6        do  j = 2, n
       9  s6           r(j) = r(j) - b(2,j)*r(j-1)
      10  s6        end do
      11       C##
      12                r(n) = r(n) / b(3, n)
      13  s4            do  j = n-1, 1, -1
      14  m4               x(j) = (r(j) - b(4,j)*x(j+1))/b(3,j)
      15  v4            end do
      16       C##
      17              case(2)
                            .
                            .
                            .
      39              end select
      40                 return
      41                 end
Vectorization messages: program name(bansol)
  jpc2217i-i   "bansol.f", line 9: r is not vectorized since recursive reference takes place.
  jpc2306i-i   "bansol.f", line 13: This DO loop is not vectorized since partial vectorization
overhead is too large.
  jpc2217i-i   "bansol.f", line 14: x is not vectorized since recursive reference takes place.
```