NAG Fortran SMP ライブラリ（線形演算）の性能評価
Performance Evaluation of NAG Fortran SMP Library

舘野 諭司，重原 孝臣 (埼玉大学), 長谷川 秀彦 (図書館情報大学), 桧山 澄子 (埼玉大学)
Satoshi Tateno, Takaomi Shigehara, Hidehiko Hasegawa and Sumiko Hiyama

1. はじめに

ライブラリ使用は，ユーザが自ら処理ルーチンを作成する手間が省けるという点だけでなく，精度の保証・汎用性といった面でも優れている．特にライブラリの実効性能は，プログラムの実行に数十時間以上かかる大規模問題を扱うユーザやリアルタイム処理が必要なユーザにとって最も関心のある部分である．しかし，ライブラリの利用によりすべての問題について実効性能が改善されるという保証はない．また，同一の処理を実現できるライブラリは一つとは限らず，施されているチューニングの差異により各ライブラリの実効性能は異なる．特に並列計算機上では，並列化手法も性能に大きな影響を与える．

本研究では HITACHI SR8000 1 ノードを共有メモリ型並列計算機として使用し，LAPACK（Linear Algebra Package）[1] 互換のルーチンを含む並列化手法が異なるライブラリの性能を評価する．比較対象は The Numerical Algorithm Group 社の NAG Fortran SMP ライブラリ Release 2(以下，*NAG* と称す)[2] およびベンダ提供の HITACHI LAPACK V01-03(以下，*Vendor* と称す) である．性能測定をおこなう問題は連立一次方程式の直接解法，対称行列/非対称行列の固有値問題である．性能評価の結果より，NAG Fortran SMP ライブラリの特徴・問題点を明らかにする．

2. 計算環境

2.1. 計算機のスペック

数値実験はすべて HITACHI SR8000 1 ノード (以下，SR8000 と称す) 上でおこなう．SR8000 は演算 CPU 8 台と制御 CPU 1 台で 1 ノードを構成しており，16GByte のメインメモリを共有する．CPU は PowerPC ベースであり，二次キャッシュを利用せずに高い性能を発揮する擬似ベクトル機構 [3, 4] を備えている．また，各 CPU の一次キャッシュは命令 64KByte，データ 128KByte であるが，並列処理時には一次キャッシュも共有され，容量が CPU 台数倍に増加する．測定に用いた CPU 台数は 1 台および 8 台である．

測定プログラムは FORTRAN77 で作成した．使用したコンパイラのバージョンとコンパイルオプションを表 1 に示す．*NAG* では OpenMP の使用が前提であるが，今回使用した FORTRAN77 コンパイラは OpenMP をサポートしていない．そのため，*NAG* の測定には OpenMP をサポートし，性能が FORTRAN77 コンパイラとほとんど変わらない FORTRAN90 コンパイラを用いた．

2.2. 比較対象のライブラリ

NAG，*Vendor* ともに演算処理には BLAS(Basic Linear Algebra Subprograms) を用いている．BLAS のチューニングによる性能変化を調査するため，Netlib[5] で配布されている未チューニングの BLAS (以下，**NetlibBLAS** と称す) ，ベンダがチューニングした BLAS(以下，**VendorBLAS** と称す) の 2 種類の BLAS を用いて性能評価をおこなう．また，Netlib で配布されている特別なチューニングの施されていない LAPACK Version 3.0(以下，*Netlib* と称す) も参考として比較対象に加える．

LAPACK (*Netlib*, *Vendor*)

並列化は BLAS ルーチン内部等の局所的な演算処理に対して重点的におこなわれている．*Netlib*，*Vendor* 共に，並列処理をおこなうかどうかで並列版とシングル版の 2 種類に分かれており，並列版ライブラリを利用する場合には並列化された BLAS をリンクする必要がある．*Vendor* はベンダによるチューニングが施されており，*Netlib* はコンパイラによる最適化・自動並列化機能のみ用いている．

NAG Fortran SMP ライブラリ (*NAG*)

並列化は演算処理 (BLAS) に対してではなく，互いにデータ依存性のない独立なルーチンを繰り返し呼び出すループやセクション処理等，ルーチン本体の並列性のある部分に対して OpenMP を用いておこなう．*NAG* を利用するプログラムは並列版・シングル版の区別はなく，並列処理に利用する CPU 台数を環境変数に指定することで並列処理をおこなう．また，並列化された処理の内部で呼び出される BLAS は 1 つの CPU 上で処理される．そのため，BLAS は並列化されていないものをリンクする．LAPACK では提供されていないルーチンも含まれているが，今回はアルゴリズムによる差をできるだけ少なくするため *NAG* に含まれる LAPACK 互換のルーチンを用いる．

3. 実験結果

数値実験は連立一次方程式，対称/非対称行列の固有値問題に対しておこなう．すべての問題において行

表 1: コンパイラのバージョンとコンパイルオプション

Compiler	Compile Option	
Optimizing FORTRAN 90 Compiler V01-03-/A	-nolimit -noscope -64 -omp -save -O4 -parallel +SBTLB	
Optimizing FORTRAN 77 Compiler V01-03-/A	共通	-nolimit -noscope -64 -pvfunc=3
	8CPU	-W0,'OPT(O(4)),MP(P(3))' -procnum=8 -parallel
	1CPU	-W0,'OPT(O(4))' -noparallel

列サイズ n は $n = 5000$ とする．OS の time コマンドを用いてプログラムの経過時間 (real) と各プロセッサの CPU 時間の合計 (user) を計測し，主に real の比較により性能を評価する．表 2〜5 中，real の単位は時:分:秒，user の単位は秒である．また，Speed-Up は 1CPU の real を 8CPU の real で割った値である．なお，実験に用いたプログラムのソースコードは [6] にて公開する．

3.1. 連立一次方程式

本節では，部分軸選択を伴う LU 分解による連立一次方程式の直接解法について議論する．係数行列は区間 $[0, 1)$ の一様乱数を成分に持つ密行列とする．LAPACK ルーチンは dgetrf,dgetrs を用いる．

表 2 に実験結果を示す．LU 分解後の求解部分の演算量 $[O(n^2)]$ は分解の演算量 $[O(n^3)]$ よりはるかに少なく，実質的に LU 分解に要する時間を反映している．VendorBLAS を用いた場合，1CPU の real は，*NAG*(0:01:25)，*Vendor*(0:05:31)，*Netlib*(0:07:15) の順で性能が良い．この結果から *NAG* は *Vendor* よりも高度なチューニングが施されていることがわかる．8CPU の real も *NAG*(0:00:18) が最も速く，次いで *Vendor*(0:00:23)，*Netlib*(0:00:28) の順となっている．*NAG* の測定には 1CPU と 8CPU とで同一のオブジェクトを用いていることから，LU 分解ルーチンに施されたチューニングは 8CPU の性能にも影響を与えていると考えられる．これらのことから，ルーチン本体の並列性のある部分に対して並列化をおこなう *NAG* の手法は，基盤となるルーチン (この場合 LU 分解のルーチン) のチューニングが重要であるといえる．

NetlibBLAS を用いた場合においても性能の順序は同じであり，Speed-Up も 6.7〜6.8 と，並列化の効果が十分に現れている．これは，LU 分解ルーチンにブロック化アルゴリズム [1, 7] が採用されアルゴリズムの並列性が一層高められていることが主な要因である．NAG の OpenMP による並列化はルーチンのアルゴリズムに対しておこなわれるため，ブロック化アルゴリズムの採用により *NAG* は並列化されやすくなると考えられる．また，ブロック化アルゴリズムでは核演算にチューニング・並列化の効果が現れやすい Level 3 BLAS が用いられている．表 3 に示す BLAS の性能比較から Level 3 BLAS のチューニング・並列化の効果は他の Level よりも大きいことが読み取れる．そのため，BLAS の並列化を重視する *Vendor*，*Netlib* でもブロック化アルゴリズムの利用は効果的である．

なお，*Vendor*，*Netlib* では Speed-Up が CPU 台数倍 (8 倍) 以上になる場合がある．主な原因は SR8000 の並列処理時には共有により一次キャッシュの容量が増加すること，ワークメモリの取り方やメモリアクセスの仕方が異なるライブラリ・BLAS のオブジェクトを並列処理時に使用していることである．

3.2. 対称固有値問題

本節では，フランク行列を用いて実対称行列の全固有値・固有ベクトルの計算を評価する．n 次フランク行列の (i, j) 成分は

$$a_{ij} = n - \max(i, j) + 1, \quad i, j = 1, \cdots, n$$

である．LAPACK ルーチンは，Relatively Robust Representation[8] による dsyevr，分割統治法による dsyevd，QR 法による dsyev を用いる．*NAG* では dsyevd と同等のルーチンを用いる．これは，dsyevr,dsyev に相当する LAPACK 互換ルーチンが *NAG* に含まれていないためである．

表 4 に実験結果を示す．VendorBLAS を用いた場合，1CPU の real は *NAG*(0:09:05)，dsyevr の *Vendor*(0:14:01)，dsyevd の *Vendor*(0:18:29)，dsyevr の *Netlib*(0:21:06)，dsyevd の *Netlib*(0:24:28) の順に性能が良く，ライブラリ自体のチューニングは *NAG* が優れていることがわかる．しかし，8CPU の real は dsyevd を用いた *Vendor*(0:01:24)，dsyevr の *Vendor*(0:01:27)，dsyevd の *Netlib*(0:01:39)，dsyevr の *Netlib*(0:01:43)，*NAG*(0:02:21) となり，並列化された BLAS を利用する *Netlib*，*Vendor* が *NAG* を上回る．これは，アルゴリズムの並列性が低い対称固有値問題では，ルーチンが十分にチューニングされていても *NAG* の手法では並列化の効果を十分に得られないことを意味する．また，dsyev を用いた *Vendor*，*Netlib* の real は 8CPU でそれぞれ 0:04:43，0:04:54 と，他のルーチンより大幅に劣る結果を得た．これは dsyev の演算量が dsyevr，dsyevd よりも多く，アルゴリズムの逐次性が高いためである．このことから固有値問題では逐次性が低く，演算量の少ないアルゴリズムを用いることが重要であるといえる．

NetlibBLAS を用いた場合には，8CPU の場合は dsyevr の *Netlib*(0:10:47) が最速，1CPU の場合は

表 2: 連立一次方程式の解法の実行時間 (単位は real 時:分:秒, user 秒)．行列サイズは $n = 5000$.

Library	VendorBLAS			NetlibBLAS		
	1CPU	8CPU	Speed-Up	1CPU	8CPU	Speed-Up
NAG	real 0:01:25 user 670	real 0:00:18 user 140	4.72	real 0:08:07 user 3870	real 0:01:11 user 140	6.86
Vendor	real 0:05:31 user 180	real 0:00:23 user 176	14.39	real 0:08:25 user 503	real 0:01:15 user 592	6.73
Netlib	real 0:07:15 user 276	real 0:00:28 user 213	15.35	real 0:08:22 user 499	real 0:01:15 user 591	6.69

表 3: Level 1,2,3 BLAS の 100 回反復時の実行時間 (単位は 秒)．行列サイズは $n = 500$.

BLAS Level	VendorBLAS			NetlibBLAS		
	1CPU	8CPU	Speed-up	1CPU	8CPU	Speed-up
Level 1 (daxpy)	0.000263	0.000249	1.06	0.000561	0.000510	1.10
Level 2 (dgemv)	0.0929	0.0197	4.72	0.2913	0.2765	1.05
Level 3 (dgemm)	95.506	10.702	8.92	145.272	18.469	7.87

NAG(0:33:55) が最速となり，dsyev を用いた *Netlib* よりも劣る性能しか発揮できていない．これは，BLAS の高速化が特に重要であることを示している．

3.3. 非対称固有値問題

本節では，非対称行列の固有値問題について検討する．固有値は全て求めるが，固有ベクトルは求めない．非対称固有値問題の一例として，実験には

$$b_{ij} = \mathrm{mod}(n + j - i, n) + 1, \quad i, j = 1, \cdots, n$$

を (i, j) 成分とする巡回行列を用いた．LAPACK ルーチンは dgeev を使用する．*NAG* には dgeev に相当するルーチンが実装されていないため，dgeev 内部で呼び出されるヘッセンベルグ行列への変換ルーチン (dgehrd 相当)，ヘッセンベルグ行列に対して QR 法により固有値を求めるルーチン (dhseqr 相当) を組み合わせて測定をおこなう．しかし，NAG ではこれらのルーチンは並列化されていない [2]．なお dgeev では，固有値の収束を速めるため，ヘッセンベルグ型に変換する前にバランシング [1] をおこなうが，実験に用いた巡回行列ではバランシングは効かない．

表 5 に実験結果を示す．*NAG* のルーチンは並列化されておらず，8CPU の並列処理時においても 1CPU と同じ処理をおこなうため，*NAG* の Speed-Up は 1.00 となる．VendorBLAS を用いた場合，1CPU の real は *NAG*(0:24:31) が最も速く，次いで *Netlib*(0:47:05)，*Vendor*(1:00:07) の順となる．8CPU の real では *Vendor*(0:13:12) は *Netlib*(0:10:01) に劣る．連立一次方程式や対称固有値問題の結果と比較すると，非対称固有値問題の性能は著しく低く，並列化も大した効果を発揮できていない．このことから，非対称固有値問題では各ライブラリとも計算機の特性を十分に活用しているとはいえず，性能改善にはアルゴリズムの改良もしくは新しいアルゴリズムの開発が不可欠であるといえる．

また，NetlibBLAS を用いた場合の real は，8CPU の場合 *Vendor* は 0:22:41，*Netlib* は 0:20:02，1CPU の場合 *NAG* は 0:48:35，*Vendor* は 1:24:47，*Netlib* は 1:12:39 となる．これは，VendorBLAS を用いても最大で 2 倍程度しか性能が改善されないことを示している．連立一次方程式や対称固有値問題の場合は VendorBLAS を用いると最大約 8 倍，平均で約 3 倍性能が向上するため，非対称固有値問題では BLAS のチューニングだけでは大幅な性能向上は期待できない．

4. まとめ

本稿では SR8000 1 ノード上において，BLAS の並列化に頼らずライブラリルーチン本体の並列性のある部分を OpenMP を用いて並列化する NAG Fortran SMP ライブラリ，BLAS 内部等の局所的な演算処理を重点的に並列化するベンダ提供の LAPACK について性能比較をおこない，NAG Fortran SMP ライブラリの特徴・問題点を明らかにした．数値実験の結果より，NAG Fortran SMP ライブラリの並列化手法は，連立一次方程式の場合のようにルーチン本体が十分にチューニングされ適度な並列性を備えているようなアルゴリズムに対しては計算機の性能を十分に引き出せる最適な方法であるが，固有値問題のように並列性が低いアルゴリズムでは最適な方法とはいえないことが明らかとなった．同時に，ベンダ提供の LAPACK はアルゴリズムの並列性が低い場合でも十分な並列化の効果を得られるが，性能の大部分が演算処理すなわち BLAS のチューニングによるものであるため，アルゴ

表 4: 対称固有値問題の実行時間 (単位は real 時:分:秒, user 秒)．行列サイズは $n = 5000$.

Library	VendorBLAS			NetlibBLAS		
	1CPU	8CPU	Speed-Up	1CPU	8CPU	Speed-Up
NAG	real 0:09:05 user 4319	real 0:02:21 user1107	3.87	real 0:33:55 user 16205	real 0:11:41 user 5550	2.90
Vendor dsyevr	real 0:14:01 user 594	real 0:01:27 user 670	9.67	real 0:35:09 user 2097	real 0:11:20 user 5384	3.10
Vendor dsyevd	real 0:18:29 user 705	real 0:01:24 user 642	13.20	real 0:43:16 user 2583	real 0:12:04 user 5752	3.59
Vendor dsyev	real 0:27:26 user 1472	real 0:04:43 user 2237	5.82	real 0:45:31 user 2710	real 13:40 user 6507	3.33
Netlib dsyevr	real 0:21:06 user 961	real 0:01:43 user 788	12.29	real 0:34:06 user 2030	real 0:10:47 user 5130	3.16
Netlib dsyevd	real 0:24:28 user 1043	real 0:01:39 user 764	14.83	real 0:42:20 user 2521	real 0:11:32 user 5495	3.67
Netlib dsyev	real 0:31:08 user1687	real 0:04:54 user 2309	6.35	real 0:45:09 user 2688	real 0:13:11 user 6275	3.42

表 5: 非対称固有値問題の実行時間 (単位は real 時:分:秒, user 秒)．行列サイズは $n = 5000$.

Library	VendorBLAS			NetlibBLAS		
	1CPU	8CPU	Speed-Up	1CPU	8CPU	Speed-Up
NAG	real 0:24:31 user 1464	real 0:24:33 user 1464	1.00	real 0:48:35 user 2903	real 0:48:43 user 2903	1.00
Vendor	real 1:00:07 user 3598	real 0:13:12 user 6271	4.55	real 1:24:47 user 5074	real 0:22:41 user 10830	3.74
Netlib	real 0:47:05 user 2817	real 0:10:01 user 4765	4.70	real 1:12:39 user 4347	real 0:20:02 user 9566	3.63

リズムを改良しなければ大幅な性能改善が期待できないことが明らかとなった．これらのことから，状況に応じて適切な方法を使い分けることで計算機を最大限に活用できるといえる．また，非対称固有値問題の結果は，現在のライブラリではプログラムの性能がほとんど改善されない一例といえる．

本研究で得た結論を踏まえると，今後並列処理を扱うライブラリに期待されることは，並列性の高いアルゴリズムを採用し常に並列計算機の特性を活用できる並列化手法を備えていること，逐次性が高く並列化が難しい問題に対しても効率良く処理できるようにチューニングが十分に施されていることである．

参考文献

[1] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J. J., DuCroz, J., Greenbaum, A., Hammarling, S., McKenney, A. and Sorensen, D.: *LAPACK Users' Guide (3rd Ed.)*, SIAM (2000).

[2] NAG Fortran SMP Library Documentation: `http://www.nag.co.uk/numeric/FL/manual/html/FSlibrarymanual.asp` .

[3] Nishiyama, H., Motokawa, K., Kyushima, I. and Kikuchi, S.: Pseudo-vectorizing Compiler for the SR8000, *Proc. Euro-Par 2000, LNCS 1900*, Springer-Verlag, pp. 1023–1027 (2000).

[4] Brehm, M., Bader, R., Heller, H. and Ebner, R.: Pseudovectorization, SMP, and Messege Passing on the Hitachi SR8000-F1, *Proc. Euro-Par 2000, LNCS 1900*, Springer-Verlag, pp. 1351–1361 (2000).

[5] Netlib: `http://www.netlib.org/`.

[6] プログラムのソースコード: `http://www.me.ics.saitama-u.ac.jp/~sigehara/hps/hps.html`.

[7] Dongarra, J. J., Duff, I. S., Sorensen, D. C. and van der Vorst, H. A.: *Numerical Linear Algebra for High-Performance Computers*, SIAM (1998).

[8] Dhillon, I. S.: *A New $O(n^2)$ Algorithm for Symmetric Tridiagonal Eigenvalue/Eigenvector Problem*, PhD Thesis, University of California at Berkeley (1997).