共有メモリ型並列計算機におけるLAPACKの性能評価

西村成司[†1] 舘野諭司[†2] 重原孝臣[†2]
長谷川秀彦[†3] 桧山澄子[†4]

本論文では，代表的な共有メモリ型並列計算機である HITACHI SR8000 1 ノード，SGI Origin2000 上において，連立一次方程式直接解法及び固有値問題を例に，標準的なアルゴリズムに基づく特別なチューニングを施さずに，コンパイラの自動並列化機能のみで並列化したプログラムと，ベンダによってチューニングされた LAPACK/BLAS を用いたプログラムの性能を比較し，ベンダ提供の LAPACK の性能評価をおこなう．その結果に基づき，連立一次方程式直接解法についてはブロック化アルゴリズムの適用および Level 3 BLAS のチューニングにより，LAPACK の高性能化が図られていることを示す．また対称固有値問題については高速アルゴリズムの利用により高性能化が図られていることを示す．一方，非対称固有値問題については既存のアルゴリズムでは共有メモリ型並列計算機の性能は十分に引き出せないことを示す．

Performance Evaluation of LAPACK Tuned for Shared Memory Parallel Computers

Seiji Nishimura,[†1] Satoshi Tateno,[†2] Takaomi Shigehara,[†2]
Hidehiko Hasegawa[†3] and Sumiko Hiyama[†4]

In this paper, we evaluate the performance of LAPACK/BLAS by comparing the runtime performance of LAPACK/BLAS with user programs on HITACHI SR8000 and SGI Origin2000, which are representative shared memory parallel computers. The LAPACK routines were tuned by vendors, whereas the user programs were implemented based on standard algorithms without using any special tuning techniques and parallelized automatically solely by the compilers. For comparison, direct solution of linear systems and eigenvalue problems are considered. The results for performance evaluation show that, in LAPACK, an application of blocked algorithms as well as highly tuned Level 3 BLAS takes a crucial role for high performance computing in direct solution of linear systems, whereas a utilization of a fastest algorithm is efficient for high performance computing in symmetric eigenvalue problems. On the other hand, even the up-to-date algorithms in LAPACK are not efficient in non-symmetric eigenvalue problems on shared memory parallel computers.

1. はじめに

共有メモリ型並列計算機は，分散メモリ型並列計算機とは異なり，コンパイラによる自動並列化が容易におこなえる．そのため，プログラマが MPI (Message Passing Interface)[1] 等のプロセッサ間通信ライブラリによる並列化を意識することなく，並列プログラムを作成できるという利点がある．共有メモリ型並列計算機は従来の CRAY のベクトル型並列計算機のようなバスメモリベースのアーキテクチャ[2] だけでなく，SGI，IBM の並列計算機のように NUMA (Non-Uniform Memory Access) [3] を採用した共有メモリ型並列計算機も開発されており，大規模科学技術計算における利用は今後ますます増大することが予想される．

一般に共有メモリ型並列計算機では，コンパイラによる自動並列化をおこなえばユーザプログラムの性能が容易にしかも飛躍的に向上するものと期待されがちである．しかし，プログラムに内在する並列度はアルゴリズムの性質に大きく依存し，単にコンパイラによる自動並列化だけで共有メモリ型並列計算機の性能を十分に引き出せるアルゴリズムは既存のアルゴリズムのなかでは少数であると言われている[4),5)]．また仮に，

†1 日本 SGI 株式会社
SGI Japan, Ltd.
†2 埼玉大学 大学院 理工学研究科 情報システム工学専攻
Course in Information and Computer Sciences, Graduate School of Science and Engineering, Saitama University.
†3 図書館情報大学
University of Library and Information Science.
†4 埼玉大学 理学部 数学科
Department of Mathematics, Faculty of Science, Saitama University.

コンパイラの自動並列化機能が有効であったとしても，それだけでは計算機の能力を十分に活用しているとは言えない．したがって，コンパイラの自動並列化機能を十分に活用し得る並列度の高い最新のアルゴリズムの利用や，各種アーキテクチャ，ハードウェアの特性に応じた適切なチューニングは，共有メモリ型並列計算機の有効活用のために必要不可欠である．このような観点に立ったとき，現段階において性能面で最も期待できるのはベンダ提供の数値計算ライブラリであろう．公開されている並列度の高い最新アルゴリズムの利用はもちろんのこと，ハードウェア特性やシステムソフトウェアに関する社内内部情報を駆使して，共有メモリ型並列計算機の能力を最大限に引き出し得ることが期待されるからである．

本論文では，共有メモリ型並列計算機である HITACHI SR8000 1 ノードと SGI Origin 2000 を対象にベンダ提供の線形計算ライブラリの評価をおこない，その長所，問題点を明らかにする．また，実験データに基づき，共有メモリ型並列計算機の性能を十分に引き出す数値計算アルゴリズムが満たすべき要件について考察する．ベンダが提供する共有メモリ型並列計算機向け線形計算ライブラリは，いずれも LAPACK (Linear Algebra Package)[6),7)] および核演算部分に BLAS（Basic Linear Algebra Subprograms)[8)~12)] を用いている．以下，ベンダ提供の線形計算ライブラリを，BLAS 部分も含めて，単にベンダ版 LAPACK と呼ぶ．本論文では実際的観点から，並列化に特別な配慮がなされていない標準的アルゴリズムに基づくプログラム[13)]（以下，ユーザ版と呼ぶ）をコンパイラの自動並列化機能のみを用いて並列実行した場合との比較・評価をおこなう．なお，ユーザ版においては BLAS を用いない．対象とする問題は，密行列を係数行列とする連立一次方程式および対称/非対称行列に対する固有値問題である．これらの問題に用いられるアルゴリズムの一部またはその変形版は一般固有値問題や特異値分解のアルゴリズムにおいても多用されるものであり，本論文で示す実験データや知見はこれらを含む線形問題全般に関わるものである．また，ソースコードが非公開であるベンダ版 LAPACK に対して施されているチューニングの効果を明らかにするために，Netlib[14)] で公開されている LAPACK および BLAS との比較・検討をおこなう．これらに基づき，ベンダ版 LAPACK の長所・問題点，および，今後開発が期待される高性能線形計算プログラムが具備すべき要件について議論する．

本論文の構成は以下の通りである．数値実験に用いた計算環境を 2 節に示す．3 節では，まず実験方法および採用したアルゴリズムを提示し，次に連立一次方程式，対称行列の固有値問題，非対称行列の固有値問題の順に実験データの解析・検討をおこない，ベンダ版 LAPACK の長所・問題点を明らかにする．4 節で本論文をまとめ，今後の課題について検討する．

2. 実験環境

表 1 に数値実験に用いた計算環境を示す．

SR8000 1 ノード（以下，SR8000 と呼ぶ）は，演算 CPU 8 台と制御 CPU 1 台からなる共有メモリ型並列計算機である．各プロセッサは擬似ベクトル機構を備えており，二次キャッシュメモリを利用せずに高い性能を発揮することができる[15),16)]．このため，SR8000 はベクトルプロセッサに近い性能を示す[17)]．また，並列処理時には一次キャッシュメモリが共有され，容量が CPU 台数倍になる．並列処理では各 CPU で扱われるデータが分割されて小さくなるため，容量の大きくなったキャッシュメモリ上にデータが載る可能性が高くなり，より効率的に処理を行えるようになる．

今回実験に用いた Origin2000 は 64 ノードから成る．1 ノードは 1 プロセッサであるが，cc-NUMA（Cache-Coherent Non-Uniform Memory Access)[3)] を採用することにより，物理的分散・論理的共有メモリ型並列計算機を実現している．

プログラミング言語は FORTRAN77 を用いる．使用したコンパイラ・ライブラリのバージョンを表 1 に，各プログラムに対するコンパイラの自動並列化・最適化オプションを**表 2** に示す．なお，Origin2000 の自動並列化オプションは複数存在し，各々のオプションで性能が異なる．実験では，ユーザ版とベンダ版 LAPACK それぞれに対し最も高い性能が得られるオプションを選択した．また，SR8000，Origin2000 共に，実験で使用するプロセッサ数は 1CPU/8CPU であり，ベンダ提供の LAPACK は Netlib で公開されている LAPACK の Version 3.0 に相当する．ベンダ版 LAPACK の測定プログラムにリンクする LAPACK および BLAS のオブジェクトは 1CPU/8CPU で異なる．

3. 実験および解析

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．各実験では，**表 3** に示すアルゴリズム/LAPACK ルーチンに基づき，ユーザ版/ベンダ版 LAPACK のプログラムを作成した．ユーザ版はメモリアクセスパターンを FORTRAN77 の仕

表 1 数値実験に用いるマシンのスペック.
Table 1 Machine Specification

	SR8000	Origin2000
Vendor	HITACHI	SGI
Processor	PowerPC base	MIPS R12000/IP27
Clock [MHz]	375	300
L1 Cache (Inst./Data) [KB/proc.]	64/128	32/32
L2 Cache [KB/proc.]	–	8192
Main Memory	16[GB/node]	0.5 [GB/proc.]
Number of Processors	48[proc.] (6 [node] × 8[proc./node])	64[proc.]
Theoretical Peak Perf. [GFLOPS/proc.]	1.5	0.6
Architecture	Pseudo-Vector Processor	cc-NUMA
OS	HI-UX/MPP 03-02	IRIX 6.5
FORTRAN77 Compiler	OFORT77 V01-02-/B	MIPSpro 7.3.1.2m
Library	HITACHI LAPACK V01-03	SCSL Scientific Library1.3.0.0

表 2 FORTRAN77 コンパイラの自動並列化・最適化オプション.
Table 2 Compiler Options.

Machine	Programs	#CPU	Compiler Options
SR8000	User & Vendor LAPACK	1CPU	-64 -nolimit -noscope -W0,'OPT(O(4))' -pvfunc=3 -noparallel -l(LAPACK/BLAS for Single Processing)
		8CPU	-64 -nolimit -noscope -W0,'OPT(O(4)),MP(P(3))' -procnum=8 -pvfunc=3 -parallel -l(LAPACK/BLAS for Multi Processing)
Origin2000	User	1CPU	-64 -mips4 -O3
		8CPU	-64 -mips4 -O3 -apo
	Vendor LAPACK	1CPU	-64 -mips4 -O3
		8CPU	-64 -mips4 -O3 -mp

表 3 ユーザ版プログラムのアルゴリズムと LAPACK ルーチン.
Table 3 Algorithms of user program and LAPACK routines for each problem.

Problems	Algorithms of User Program	LAPACK Routines
LU Factorization	1) LU factorization with partial pivoting 2) Solve linear equation by forward and backward substitutions	dgetrf dgetrs
Eigenvalue Problem (Symmetric Matrix)	1) Tridiagonalization by Householder transformation 2) Find all eigenvalues and all eigenvectors by bisection method and inverse iteration method	dsyevr (dsyev)
Eigenvalue Problem (Non-Symmetric Matrix)	1) Transform to Hessenberg form by modified Gaussian elimination 2) Find all eigenvalues by QR Method with implicit shifts	dgeev

様に合わせてあるが，ループアンローリングやブロック化といった特別なチューニングは何も施していない．なお，実験に用いた全プログラムは 13) にて公開する．

時間計測には OS の time コマンドを用い，プログラムの経過時間（real time）と各プロセッサの CPU 時間の合計（user time）を計測した．並列プログラムでは，real time を用いて並列化の効果・性能の差異が，また，real time と user time の比較によりシステムのオーバヘッドがわかる．表 4〜10 中，real は real time，user は user time を表す．単位はともに時:分:秒である．また，Speed-Up は 1CPU の real time を 8CPU の real time で割った値を表し，n は問題サイズ（行列の次元）を表す．

ベンダ版 LAPACK に施されているチューニングの効果を明らかにするため，問題毎にベンダ提供の LAPACK，BLAS コードを Netlib で配布されている LAPACK，BLAS のサンプルコードで置き換え，比較実験をおこなう．表 5,8,10 中，LAPACK，BLAS に用いたコードの種類を

（LAPACK のコード，BLAS のコード）

の形式で区別する．v はベンダコード，n が Netlib コードを示す．例えば，(v,n) は，LAPACK にベンダコー

ド，BLAS に Netlib コードを用いた場合に対応する．なお，1CPU，8CPU いずれの場合も計算結果の正確性は確認した．

3.1 連立一次方程式

本節では，ピボットの部分選択を伴う 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．

表 4 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．表 4 からわかるように，ベンダ版 LAPACK はユーザ版に比べて実行時間が短く，Speed-Up が大きい．理由を明らかにするために，SR8000 8CPU，$n = 5000$ の場合に，LAPACK/BLAS にベンダコード/Netlib コードを組み合わせて実行した実験例を**表 5** に示す．表 5 より，LAPACK，BLAS の両方に Netlib コードを用いた場合でも real time 0:00:49 とユーザ版の real time 0:03:21 に勝ることがわかる．これは，LAPACK の LU 分解ルーチン `dgetrf` ではブロック化アルゴリズム[6),18)] を適用し，核演算部分で Level 3 BLAS[12)] を利用しているためである．Netlib の Level 3 BLAS ではループ交換可能な密多重のループ構造が用いられており，コンパイラによる自動並列化の効果が大きい．また LAPACK は，ブロック化パラメータ，計算機における演算方式等の環境依存のパラメータを `ilaenv` ルーチンによって取得し，適切なチューニングをおこなうことができる[6)]．このため，ブロック化アルゴリズムによる LU 分解は適切なブロック化サイズの適用により Netlib コードを用いても大幅に高速化される．表 5 より，LAPACK にベンダコードを使用しても real time 0:00:49 であり，ほとんど性能向上はみられないこともわかる．したがって，ベンダ版 LU 分解ルーチンの高速化はもっぱら BLAS のチューニングによる．特に，行列-行列演算を扱う Level 3 BLAS のチューニングは Level 1,2 BLAS のチューニングよりも大きな効果が得られる．**表 6** に示すとおり，SR8000 上での行列/ベクトルサイズ $n = 5000$ に対する代表的な BLAS ルーチンの実験例では，Level 3 BLAS の Netlib/Vendor (Netlib の real time を Vendor の real time で割った値) は 8CPU の場合に 10.11 であり，Level 1 BLAS の 1.50，Level 2 BLAS の 4.42 と比較して非常に大きい．ブロック化アルゴリズムを適用したルーチンでは Level 3 BLAS が核演算部分に用いられるため，Level 3 BLAS のチューニングが全体の性能に大きな影響を与える．以上より，ベンダ版 LAPACK では，ブロック化アルゴリズムを適用し，Level3 BLAS に対してループアンローリング，ブロック化，等価なアルゴリズムへの変換等，計算機に特化したチューニングをおこなうことによって，コンパイラの自動最適化機能だけでは得られない高い実効性能を実現していることがわかる．Origin2000 でも SR8000 と同様の傾向を示す．なお，表 6 において Level 3 BLAS 1CPU の Netlib/Vendor が 1.66 と小さいのは，ベンダ版の BLAS の real time が 02:23:11 と遅いためである．user time 01:00:57 との差が 01:21:42 であることから，この原因はシステムの負荷が著しく大きくなったためと考えられる．user time を基にした Netlib/Vendor は 3.91 であるので，システムの負荷を除けば 8CPU の場合と同様にチューニングの効果が最も大きい．

一方，ユーザ版ではピボットの部分選択のために最内側ループにメモリへの間接参照が現れ，キャッシュミスが多発すると同時にコンパイラによる最適化が適切におこなわれない．特に，Origin2000 ではアーキテクチャ上，キャッシュミスペナルティが大きく[3)]，大規模問題でその影響が顕著に現れる．表 4 からわかるように，実際，問題サイズを $n = 1000$ から $n = 5000$ に増加すると，real time は 1CPU の場合で 580 倍，8CPU の場合で 557 倍に増加し，演算量の増加率 (約 125 倍) よりはるかに大きくなっている．ベンダ版 LAPACK では，ブロック化アルゴリズムの採用によりキャッシュミスを大幅に減少させ，Origin2000 上でも高い実効性能が得られる．なお，SR8000 では擬似ベクトル化によりキャッシュミスの影響は生じていない[17)]．実際，ユーザ版で問題サイズを 1000 から 5000 に増加させた場合でも real time の増加率は 1CPU の場合で 109 倍，8CPU の場合で 50 倍に抑えられ，性能はむしろ向上していることがわかる．

なお，表 4 の SR8000 8CPU において，$n = 5000$ のベンダ版 LAPACK の Speed-Up が CPU 台数倍 (8 倍) よりも大きい 15.00 となっている．これは，ベンダ提供の 8CPU 版 BLAS は 1 ノード上で 1 プログラムのみの動作を前提とした高度な最適化が施されており，ワークメモリの取り方・メモリアクセスの仕方も MPI 等でも利用される 1CPU 版と異なるためである．加えて，8CPU におけるキャッシュメモリの共有による一次キャッシュメモリの容量増加も Speed-Up が CPU 台数倍以上になる要因であると考えられる．

3.2 対称行列の固有値問題

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

表 4 LU 分解におけるユーザ版とベンダ版 LAPACK の実行時間 (単位は時:分:秒). n は問題サイズ.
Table 4 Real (elapsed) and user times of LAPACK routines and user programs for linear system (hour:minute:second). n is problem size.

		SR8000			Origin2000		
LU Factorization		1CPU	8CPU	Speed-Up	1CPU	8CPU	Speed-Up
User	n=1000	real 0:00:06 user 0:00:06	real 0:00:04 user 0:00:28	1.50	real 0:00:37 user 0:00:38	real 0:00:16 user 0:01:36	2.31
	n=5000	real 0:10:54 user 0:10:53	real 0:03:21 user 0:26:40	3.25	real 5:57:29 user 5:55:47	real 2:28:31 user 11:56:21	2.41
Vendor LAPACK (dgetrf,dgetrs)	n=1000	real 0:00:04 user 0:00:02	real 0:00:01 user 0:00:05	4.00	real 0:00:01 user 0:00:02	real 0:00:01 user 0:00:05	1.00
	n=5000	real 0:05:30 user 0:02:58	real 0:00:22 user 0:02:52	15.00	real 0:03:11 user 0:03:02	real 0:01:01 user 0:03:54	3.13

表 5 SR8000 8CPU 上でのベンダコードと Netlib コードでの LU 分解の実行時間 (単位は時:分:秒). 問題サイズは $n = 5000$. LAPACK ルーチンは dgetrf,dgetrs.
Table 5 Comparison of real (elapsed) and user times for LU factorization with vendor code (v) and Netlib code (n) on SR8000 8CPU (hour:minute:second); (lapack code, blas code)=(v/n,v/n). Problem size is $n = 5000$. LAPACK routines are dgetrf,dgetrs.

Code	(v,v)	(n,v)	(v,n)	(n,n)	User
real time	00:00:22	00:00:23	00:00:49	00:00:49	00:03:21
user time	00:02:52	00:02:55	00:06:27	00:06:29	00:26:40

表 6 SR8000 上での各 BLAS レベルにおけるベンダコードと Netlib コードの実行時間 (単位は時:分:秒). 行列サイズ/ベクトル長は $n = 5000$.
Table 6 Comparison of real (elapsed) and user times for a typical BLAS routine at each level between vendor code and Netlib code on SR8000 (hour:minute:second). Matrix size/vector length is $n = 5000$.

BLAS routines	Level 1 BLAS (daxpy) 2×10^5 iterations		Level 2 BLAS (dgemv) 2000 iterations		Level 3 BLAS (dgemm) 10 iterations	
	1CPU	8CPU	1CPU	8CPU	1CPU	8CPU
Netlib BLAS	real 00:00:10 user 00:00:11	real 00:00:03 user 00:00:19	real 00:09:34 user 00:09:32	real 00:01:55 user 00:15:04	real 03:59:06 user 03:58:09	real 00:30:10 user 03:59:58
Vendor BLAS	real 00:00:04 user 00:00:04	real 00:00:02 user 00:00:12	real 00:02:57 user 00:02:56	real 00:00:26 user 00:03:20	real 02:23:11 user 01:00:57	real 00:02:59 user 00:23:35
Netlib / Vendor	2.50	1.50	3.24	4.42	1.66	10.11

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

である. ユーザ版では, ハウスホルダー変換により行列を三重対角化し, 二分法と逆反復法により全固有値・固有ベクトルを求める. LAPACK ルーチンは, LAPACK Version 3 より導入され, 使用が推奨されている dsyevr を用いる. 固有値の精度はユーザ版, ベンダ版 LAPACK 共に同じオーダー (相対残差が 10^{-12} 未満) に設定した.

表 7 に実験結果を示す. 問題サイズ $n = 5000$ での real time は, SR8000 8CPU の場合, ベンダ版 LAPACK で 0:01:26, ユーザ版で 0:04:14, Origin2000 8CPU の場合, ベンダ版 LAPACK で 0:19:21, ユーザ版で 0:51:04 となり, ベンダ版 LAPACK はユーザ版に比べて 2.5 倍以上の性能を示す. dsyevr では Relatively Robust Representation[19] を用いて, 固有値が密集すると最悪 $O(n^3)$ の演算量が必要となる三重対角化後のプロセスを $O(n^2)$ の演算量に抑えている. 加えて, Relatively Robust Representation では逆反復法の逐次性を低減することができ, LU 分解の場合に大幅な性能向上をもたらしたブロック化アルゴリズムも利用できる. そのため, ベンダ版 LAPACK はユーザ版を上回る性能を発揮する. LAPACK には, 三重対角化後に QR 法を用いる標準的手順に従った dsyev というルーチンもある. しかし, dsyev を用いたベンダ版 LAPACK の real time は SR8000 8CPU, $n = 5000$ の場合で 0:04:29 であり, ユーザ版よりも劣る[20]. IBM の Power アーキテクチャ上でも dsyevr は dsyev に比べて 2.5 倍から 3 倍の性能を示してお

表 7 対称行列の固有値・固有ベクトル問題の実行時間 (単位は時:分:秒). n は問題サイズ.
Table 7 Real (elapsed) and user times of LAPACK routines and user programs for symmetric eigenvalue/eigenvector problem (hour:minute:second). n is problem size.

		SR8000			Origin2000		
Symmetric Matrix		1CPU	8CPU	Speed-Up	1CPU	8CPU	Speed-Up
User	n=1000	real 0:00:17 user 0:00:16	real 0:00:04 user 0:00:31	4.25	real 0:00:25 user 0:00:25	real 0:00:40 user 0:01:39	0.62
	n=5000	real 0:27:18 user 0:27:11	real 0:04:14 user 0:33:36	6.45	real 2:58:13 user 2:51:53	real 0:51:04 user 6:52:43	3.49
Vendor LAPACK (dsyevr)	n=1000	real 0:00:11 user 0:00:07	real 0:00:03 user 0:00:18	3.67	real 0:00:10 user 0:00:10	real 0:00:08 user 0:00:56	1.25
	n=5000	real 0:13:56 user 0:09:52	real 0:01:26 user 0:11:17	9.72	real 0:32:06 user 0:31:55	real 0:19:21 user 2:12:19	1.66

表 8 SR8000 8CPU 上でのベンダコードと Netlib コードでの対称固有値問題の実行時間 (単位は時:分:秒). 問題サイズは $n = 5000$. LAPACK ルーチンは dsyevr.
Table 8 Comparison of real (elapsed) and user times for symmetric eigenvalue problem with vendor code (v) and Netlib code (n) on SR8000 8CPU (hour:minute:second); (lapack code, blas code)=(v/n,v/n). Problem size is $n = 5000$. LAPACK routine is dsyevr.

Code	(v,v)	(n,v)	(v,n)	(n,n)	User
real time	00:01:26	00:01:41	00:08:11	00:07:43	00:04:14
user time	00:11:17	00:13:15	01:05:07	01:01:27	00:33:36

り[6], SR8000, Origin2000 上でも同程度かそれ以上の性能改善がみられる.

SR8000 8CPU, $n = 5000$ の場合に, LAPACK/BLAS に対してベンダコード/Netlib コードを組み合わせて実行した実験例を**表 8** に示す. BLAS に Netlib コードを用いた場合の real time は, (v,n) で 00:08:11, (n,n) で 00:07:43 と, BLAS にベンダコードを用いた場合よりも大幅に性能が低下する. この事から, LU 分解と同様に dsyevr でも BLAS のチューニングが性能向上に大きな影響を与えていることがわかる. これは, dsyevr において Level 3 を含む多くの BLAS ルーチンが演算の主要部を占めるためである. また, LU 分解の場合と異なり, BLAS に Netlib コードを用いたものはユーザ版の半分程度の実効性能しか得られていない. 原因は, LU 分解ではアルゴリズムの書き換えやブロック化の効果で Netlib コードの性能が著しく向上したのに対し, 対称固有値問題では dsyevr でのブロック化の効果が LU 分解ほど顕著でないためである. なお, Origin2000 においても, 表 8 と同様の傾向の結果が得られた.

3.3 非対称行列の固有値問題

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. ユーザ版では, ピボット選択つきガウス消去法と同等の手順により係数行列をヘッセンベルグ行列へ変換し, シフト付き QR 法により全固有値を求める[21]. LAPACK ルーチンは dgeev を使用する. dgeev では, ヘッセンベルグ行列への変換にハウスホルダー変換を用いるため, 変換に要する演算量はユーザ版の約 2 倍である[21]. 固有値の精度はユーザ版, ベンダ版 LAPACK ともに同じオーダー (相対残差が 10^{-12} 未満) に設定した. 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[6] 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 9 に実験結果を示す. 問題サイズ $n = 5000$ に対する real time は, SR8000 8CPU でベンダ版 LAPACK は 00:11:44, ユーザ版は 00:08:16, 一方, Origin2000 8CPU でベンダ版 LAPACK は 02:40:55, ユーザ版は 02:08:12 となり, SR8000, Origin2000 ともにベンダ版 LAPACK の性能はユーザ版に劣るという結果を得た. 1CPU の場合は, SR8000 でベンダ版 LAPACK は 01:04:34, ユーザ版は 00:25:13 でありベンダ版 LAPACK の性能が劣る. 一方 Origin2000 では, ベンダ版 LAPACK が 03:09:46, ユーザ版が 07:45:04 となり, ユーザ版の性能が劣る. ユーザ版はベンダ版 LAPACK に比べて約半分の演算量でヘッセンベルグ行列へ変換していることを考慮しても, ベンダ版 LA-

表 9 非対称行列の固有値問題の実行時間 (単位は時:分:秒). n は問題サイズ.
Table 9 Real (elapsed) and user times of LAPACK routines and user programs for non-symmetric eigenvalue problem (hour:minute:second). n is problem size.

		SR8000			Origin2000		
Non-Symmetric Matrix		1CPU	8CPU	Speed-Up	1CPU	8CPU	Speed-Up
User	n=1000	real 0:00:14 user 0:00:14	real 0:00:12 user 0:01:29	1.17	real 0:01:05 user 0:01:02	real 0:01:03 user 0:06:19	1.03
	n=5000	real 0:25:13 user 0:25:07	real 0:08:16 user 1:05:42	3.05	real 7:45:04 user 7:28:39	real 2:08:12 user 16:29:12	3.63
Vendor LAPACK (dgeev)	n=1000	real 0:00:33 user 0:00:28	real 0:00:19 user 0:02:27	1.74	real 0:00:45 user 0:00:44	real 0:00:39 user 0:01:19	1.15
	n=5000	real 1:04:34 user 0:57:31	real 0:11:44 user 1:33:13	5.50	real 3:09:46 user 3:03:00	real 2:40:55 user 4:06:18	1.18

表 10 SR8000 8CPU 上でのベンダのコードと Netlib のコードでの非対称固有値問題の実行時間 (単位は時:分:秒). 問題サイズは $n = 5000$. LAPACK ルーチンは dgeev.
Table 10 Comparison of real (elapsed) and user times for non-symmetric eigenvalue problem with vendor code (v) and Netlib code (n) on SR8000 8CPU (hour:minute:second); (lapack code, blas code)=(v/n,v/n). Problem size is $n = 5000$. LAPACK routine is dgeev.

Code	(v,v)	(n,v)	(v,n)	(n,n)	User
real time	00:11:44	00:09:26	00:15:09	00:11:26	00:08:16
user time	01:32:13	01:14:58	02:00:28	01:30:58	01:05:42

PACK の性能が勝るとは言えない.

ベンダ版 LAPACK の実験結果を Speed-Up について対称固有値問題の結果と比較すると, $n = 5000$ の場合, 対称固有値問題において SR8000 では 9.72, Origin2000 では 1.66 であった Speed-Up が, 非対称固有値問題においては SR8000 では 5.50, Origin2000 では 1.18 と, SR8000, Origin2000 ともに小さくなっている. 非対称行列の固有値問題では既存のアルゴリズム自身の並列性が低いため, ベンダ版 LAPACK でも核演算部分を Level 3 BLAS に帰着できず, BLAS のチューニングの効果が十分に得られない. Speed-Up の減少はユーザ版にも見られる. したがって, 非対称固有値問題においては, ベンダ版 LAPACK・ユーザ版ともに共有メモリ型計算機の特性を十分に活用しているとは言えない. なお, Origin2000 では, ユーザ版がベンダ LAPACK 版よりも大きな Speed-Up を得ているが, これはユーザ版, 1CPU の性能が低いためで, 見かけ上のものである.

LAPACK/BLAS に対してベンダコード/Netlib コードを組み合わせ SR8000 8CPU で実行した実験例を**表 10** に示す. 問題サイズは $n = 5000$ である. LAPACK にベンダコードを用いた場合, real time は, (v,v) で 00:11:44, (v,n) で 00:15:09 となり, 一方 LAPACK に Netlib コードを用いた場合には (n,v) で 00:09:26, (n,n) で 00:11:26 となり, LAPACK に Netlib コードを用いた方が性能が良くなることもわかる. また, LU 分解や対称固有値問題の場合と同様に BLAS のチューニングが有効であることがわかる. Origin2000 でも同様の結果を得た. この結果からも, これまで知られている非対称固有値問題に対するアルゴリズムは演算の逐次性が高いために並列化および高速化が難しいこと, BLAS のチューニングが LU 分解や対称固有値問題の場合ほど十分な効果を発揮していないことがわかる.

4. ま と め

本論文では, 共有メモリ型並列計算機 HITACHI SR8000 1 ノードと SGI Origin2000 上で, ベンダにより最適化された LAPACK (ベンダ版 LAPACK) とユーザの作成した標準的なアルゴリズムに基づくプログラム (ユーザ版) の比較をおこなった. LU 分解および対称固有値問題の実験結果から, ベンダ版 LAPACK の高速化は, LAPACK 本体ではなく, BLAS の高速化によっていることが明らかになった. したがって, 一般ユーザによるプログラミングにおいても核演算部分を BLAS に帰着させれば, ベンダ提供の BLAS を用いることによってプログラムの高速化が十分に期待できる. 特に, 1 ロード/ストア命令当りの演算命令実行比率が高い Level 3 BLAS の活用がプログラムの高速化に有効であり, 可能ならば核演算部分で Level 3

BLAS を使用できるようにアルゴリズム自身をチューニングすることが重要である．LU 分解のブロック化による高速化はその典型例である．

従来の QR 法によるルーチンではユーザ版に劣る性能しか得られなかった対称固有値問題では，新しいアルゴリズムを用いたルーチンを使うことにより性能が改善された．この実験結果より，共有メモリ型並列計算機の性能を十分に引き出すためには，演算の並列性が高いアルゴリズムを採用することが重要であることがわかる．また，Relatively Robust Representation のような高速で新しいアルゴリズムを積極的に扱っているという点において，LAPACK の利用は有益である．

非対称固有値問題では，ベンダ版 LAPACK がユーザ版に比べ性能が劣る，ないしは，せいぜい同等という意外な結果を得た．主な理由は，ユーザ版ではヘッセンベルグ行列への変換の際にピボット選択つきガウス消去法と同等な変換を用いたためにハウスホルダー変換を用いる場合に比べて演算量を約 5 割削減できたことによる．一方，数値計算ライブラリとして精度を追求するなら，演算量は多くても安定なハウスホルダー変換を用いる必要がある．いずれにしても今回の実験結果は性能・並列度の両面からみて，核演算部分のチューニングだけでは共有メモリ型並列計算機においても非対称固有値問題を効率よく解くことは困難であることを示している．

本論文で対象とした共有メモリ型並列計算機向け線形計算ライブラリは核演算部分の並列化を促進することによってプログラムの高速化を実現するものである．この方向を一層発展させるためには，共有メモリ型並列計算機においても基本線形演算自動チューニングライブラリ[22] の開発が重要であろう．一方これとは逆に，OpenMP[23] など，指示子を用いてループまたはセクションを並列化し，核演算部分には単一プロセッサ用モジュールを用いるアプローチも有力である（この際，核演算部分に対するチューニングが必要となることは言うまでもない）．実際，NAG （Numerical Algorithm Group） SMP ライブラリでは後者のアプローチが採用されている．これら二つのアプローチの得失を明らかにすることは，今後開発すべき共有メモリ型並列計算機向け高性能線形計算プログラムが具備すべき要件をより一層明確にする上で極めて重要な課題である．

謝辞 本研究は，2001 年 5 月におこなわれた数値解析シンポジウムでの発表を母体とし，東京大学 地震研究所 地震予知情報センター 共同利用システムと埼玉大学 総合情報処理センターの計算機資源を用いておこなった．

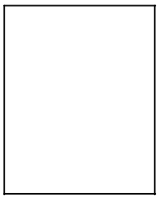
本研究の遂行にあたり，SR8000 上で最新版の LAPACK を利用できるようにしていただき，数々の情報を提供していただいた（株）日立製作所 後 保範 氏，吉村 卓弘 氏，並びに Origin2000 上でのコンパイラとライブラリのバグの原因を敏速に究明・対処していただいた日本 SGI 株式会社 システム技術・サービス本部 金 熙 博士に感謝の意を表する．

参 考 文 献

1) Message Passing Interface Forum: http://www.mpi-forum.org/.
2) Fernbach, S.(ed.): *Supercomputer*, North-Holland (1988).
3) Scott, S. L.: A Cache Coherence Mechanism for Scalable, Shared-Memory Multiprocessors, *Proc. 1991 ISSMM*, pp. 49–59 (1991).
4) 津田孝雄: 並列数値処理の現状, 応用数理, Vol.11, No. 1, pp. 14–32 (2001).
5) Pieper, K. L.: *Parallelizing Compilers: Imprementation and Effectiveness*, PhD Thesis, Stanford University, (1993). http://suif.stanford.edu/.
6) 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).
7) 山本喜一, 榊原進, 野寺隆志, 長谷川秀彦: これだけは知っておきたい数学ツール, 共立出版 (1999).
8) 小国力, 村田健郎, 三好俊郎, Dongarra, J. J., 長谷川秀彦: 行列計算ソフトウエア, 丸善 (1991).
9) Dongarra, J. J., Duff, I. S., Sorensen, S. C. and van der Vorst, H. A.: *Solving Linear Systems on Vector and Shared Memory Computers*, SIAM (1991).
10) Lawson, C., Hanson, R., Kincaid, D. and Krogh, F.: Basic Linear Algebra Subprograms for FORTRAN Usage, *ACM Trans. Math. Software*, Vol. 5, pp. 308–325 (1979).
11) Dongarra, J. J., DuCroz, J., Hammarling, S. and Hanson, R.: An Extended Set of FORTRAN Basic Linear Algebra Subprograms, *ACM Trans. Math. Software*, Vol. 14, pp. 1–17 (1988).
12) Dongarra, J. J., DuCroz, J., Duff, I. S. and Hammarling, S.: A Set of Level 3 Basic Linear Algebra Subprograms, *ACM Trans. Math. Software*, Vol. 16, pp. 1–17 (1990).
13) プログラムのソースコード: http://www.me.ics.saitama-u.ac.jp/~sigehara/hps/hps.html.
14) Netlib: http://www.netlib.org/.

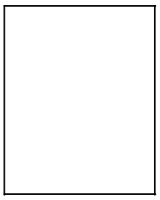
15) Nishiyama, H., Motokawa, K., Kyushima, I. and Kikuchi, S.: Pseudo-vectorizing Compiler for the SR8000, *Proc. Euro-Par 2000, LNCS 1900*, Springer-Verlag, pp. 1023–1027 (2000).
16) Brehm, M., Bader, R., Heller, H. and Ebner, R.: Pseudovectorization, SMP, and Messege Passing on the Hitachi SR8000-F1, *Proc. Euro-Par 2000, LNCS 1900*, Springer-Verlag, pp. 1351–1361 (2000).
17) Nishimura, S., Takahashi, D., Shigehara, T., Mizoguchi, H. and Mishima, T.: A Performance Study on a Single Processing Node of the HITACHI SR8000, *Proc. of NAA2000, LNCS 1988*, Springer-Verlag, pp. 628–635 (2001).
18) Dongarra, J. J., Duff, I. S., Sorensen, D. C. and van der Vorst, H. A.: *Numerical Linear Algebra for High-Performance Computers*, SIAM (1998).
19) Dhillon, I. S.: *A New $O(n^2)$ Algorithm for Symmetric Tridiagonal Eigenvalue/Eigenvector Problem*, PhD Thesis, University of California at Berkeley (1997).
20) 西村成司, 舘野論司, 重原孝臣, 桧山澄子: 共有メモリ型並列計算機向けにチューニングされた LAPACK の有効性, 第 30 回数値解析シンポジウム講演 予稿集, pp. 59–62 (2001).
21) Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P.: *Numerical Recipes in FORTRAN Second Edition*, Cambridge University Press (1992).
22) Whaley, R. C. and Dongarra, J. J.: Automatically Tuned Linear Algebra Software, *Proceedings of the SC98 Conference, Orlando, FL*, IEEE Publications (1998).
`http://www.netlib.org/atlas/`.
23) OpenMP Architecture Review Board:
`http://www.OpenMP.org/`.

(平成 年 月 日受付)
(平成 年 月 日採録)



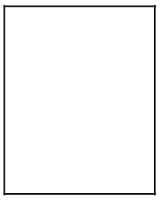
西村 成司（正会員）

昭和 51 年生. 平成 11 年 埼玉大学 工学部 情報工学科 卒. 平成 13 年 埼玉大学 大学院 理工学研究科 情報工学専攻 修了. 平成 14 年 埼玉大学 大学院 理工学研究科 情報数理科学専攻 中退. 同年 日本 SGI 株式会社 入社. 専門分野はハイパフォーマンス コンピューティング, 並列数値計算. 日本応用数理学会, 電子情報通信学会 各会員.



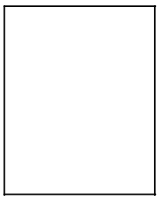
舘野 諭司

昭和 54 年生. 平成 13 年 埼玉大学 工学部 情報システム工学科 卒. 同年より 埼玉大学 大学院 理工学研究科 情報システム工学専攻 在学. 専門分野はハイパフォーマンス コンピューティング, 並列数値計算.



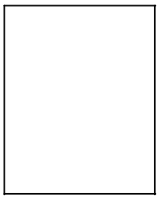
重原 孝臣（正会員）

昭和 35 年生. 昭和 58 年 東京大学 理学部 物理学科 卒. 昭和 63 年 東京大学 大学院 理学系研究科 物理学専攻 修了. 同年より 東京大学 大型計算機センター 研究開発部 助手, 平成 9 年より 埼玉大学 工学部 情報システム工学科 講師を経て, 現在, 同 助教授. 理学博士. 専門分野はハイパフォーマンス コンピューティング, 並列数値計算, 数理物理, 量子カオス. 電子情報通信学会, 日本物理学会 各会員.



長谷川秀彦（正会員）

昭和 33 年生. 昭和 55 年 筑波大学 第 1 学群 自然学類 卒業. 昭和 58 年 筑波大学 大学院 社会工学研究科 中退. 現在, 図書館情報大学 助教授. 博士（工学）. PHASE, LA 研究会を運営する. 専門分野は線形数値計算. 日本応用数理学会, SIAM, ACM 各会員.



桧山 澄子（正会員）

昭和 15 年生. 昭和 38 年 宇都宮大学 教育学部 数学科 卒. 昭和 40 年 東京都立大学 大学院 理学研究科 数学専攻 修了. 昭和 43 年より 東京大学 地震研究所 所属. 平成 13 年同研究所 地震予知情報センター 停年退官. 現在, 埼玉大学 理学部 数学科 非常勤講師. 博士（工学）. 専門分野はハイパフォーマンス コンピューティング, 数値計算. 日本応用数理学会 会員.