Vol.2009-ARC-186 No.7
Vol.2009-HPC-123 No.7
2009/12/1

複数アーキテクチャ上での疎行列ベクトル積の性能最適化手法

西　田　　晃[†1]

本稿では，現在開発を進めている並列反復解法ライブラリ Lis の複数アーキテクチャ上での性能最適化手法について検討する．反復解法の性能は，行列の形状，並列化手法，計算機のメモリ階層等によって大きく性能が変化する．ここでは，反復解法の要素演算である疎行列ベクトル積の性能を事前に評価することにより，反復解法の性能を最適化するための手法の妥当性について考察する．

Performance Optimization of Sparse Matrix-Vector Product on Multiple Architectures

Akira Nishida [†1]

This study discusses the performance optimization of the the parallel iterative solver library Lis, which is developed in our project. The performance of iterative solvers is affected by the data structure of matrices, the methodology of their parallelization, the memory hierarchy of the computers etc. In this paper, we evaluate the validity of the performance optimization of iterative solvers using the preliminary performance evaluation of sparse matrix vector products.

1. 背　　景

本研究では, 平成 14-19 年度科学技術振興機構 CREST 事業の一環として, 反復解法ライブラリ Lis [⋆1] を開発, 配布し, 様々な並列計算機上で大規模な線型方程式を解くための環境を提供している．また平成 20 年度には九州大学情報基盤研究開発センターにおいて固有値解法の実装を行い, 同年 11 月より疎行列固有値解法に対応した新版を公開している．

本稿では，既に反復解法ライブラリ Lis に実装している疎行列ベクトル積の性能ベンチマークプログラムを発展させ，多様なアーキテクチャ上で評価を行った．

†1 九州大学情報基盤研究開発センター
Research Institute for Information Technology, Kyushu University

⋆1 http://www.ssisc.org/lis/

2. Lis を用いた実装

本研究では, 以前より開発を進めている Lis 上に, Krylov 部分空間法を中心とする多様な反復解法を実装している．同様なライブラリとして，Argonne 米国立研究所の並列反復解法ライブラリ PETSc や Lawrence Berkeley 米国立研究所による並列反復解法ライブラリ Hypre などが知られている[1)]．これらのライブラリでは，いずれもオブジェクト指向に基づいた設計を行っており, 並列化はライブラリのレベルで実現されている．また, すべてのデータは API を用いて処理されている．すなわち, 行列, ベクトルデータ等の生成・廃棄, 及びこれらのオブジェクトに対する操作は, それぞれの操作を記述する API を呼び出すことにより処理されている．

反復解法において，疎行列ベクトル積は計算時間の大半を占めることが多く，性能の最適化が最も重要な処理である．Lis では，多様な行列格納形式について疎行列ベクトル積を実装しており，形式間の変換が可能となっている．以下では，格納形式とそれによって生じる性能の違いについて考察する．

3. 行列格納形式

本節では，Lis で使用できる行列の格納形式について述べる[2)]．行列の行 (列) 番号は 0 から始まるものとする．$n \times n$ 行列 $A = (a_{ij})$ の非零要素数を nnz とする．

3.1 Compressed Row Storage (CRS)

CRS 形式は行列データを 3 つの配列 (`ptr`,`index`,`value`) に格納する．

- 長さ nnz の倍精度配列 `value` は行列 A の非零要素の値を行方向に沿って格納する．
- 長さ nnz の整数配列 `index` は配列 `value` に格納された非零要素の列番号を格納する．
- 長さ $n+1$ の整数配列 `ptr` は配列 `value` と `index` の各行の開始位置を格納する．

CRS 形式での格納方法を図 1 に示す.

図 1 の行列 A を 2 プロセス上に CRS 形式で格納すると図 2 のようになる．

3.2 Compressed Column Storage (CCS)

CCS 形式は行列データを 3 つの配列 (`ptr`,`index`,`value`) に格納する．

Vol.2009-ARC-186 No.7
Vol.2009-HPC-123 No.7
2009/12/1

$$A = \begin{pmatrix} 11 & & & \\ 21 & 22 & & \\ & 32 & 33 & \\ 41 & & 43 & 44 \end{pmatrix}$$

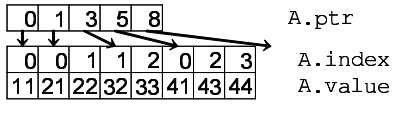

図 1 Data structure of CRS.

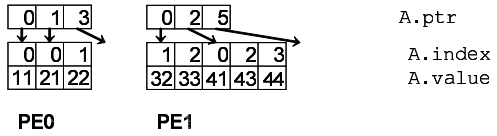

図 2 Data structure of CRS on two processes.

- 長さ nnz の倍精度配列 `value` は行列 A の非零要素の値を列方向に沿って格納する．
- 長さ nnz の整数配列 `index` は配列 `value` に格納された非零要素の行番号を格納する．
- 長さ $n+1$ の整数配列 `ptr` は配列 `value` と `index` の各列の開始位置を格納する．

CCS 形式での格納方法を図 3 に示す．

$$A = \begin{pmatrix} 11 & & & \\ 21 & 22 & & \\ & 32 & 33 & \\ 41 & & 43 & 44 \end{pmatrix}$$

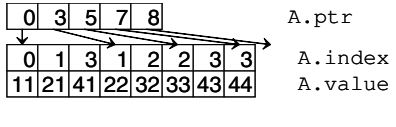

図 3 Data structure of CCS.

2 プロセス上への CCS 形式での格納方法を図 4 に示す．

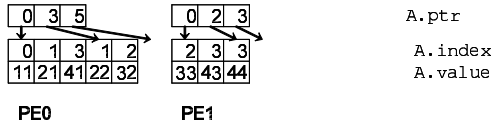

図 4 Data structure of CCS on two processes.

3.3 Modified Compressed Sparse Row (MSR)

MSR 形式は CRS 形式を修正したものである．その違いは対角部分を分けて格納しているところである．MSR 形式は行列データを 2 つの配列 (`index,value`) に格納する．ndz を対角部分の零要素数とする．

- 長さ $nnz + ndz + 1$ の倍精度配列 `value` は n 番目までは行列 A の対角部分を格納する．$n+1$ 番目の要素は使用しない．$n+2$ 番目からは行列 A の対角以外の非零要素の値を行方向に沿って格納する．
- 長さ $nnz + ndz + 1$ の整数配列 `index` は $n+1$ 番目までは行列 A の非対角部分の各行の開始位置を格納する．$n+2$ 番目からは行列 A の非対角部分の配列 `value` に格納された非零要素の列番号を格納する．

MSR 形式での格納方法を図 5 に示す．

$$A = \begin{pmatrix} 11 & & & \\ 21 & 22 & & \\ & 32 & 33 & \\ 41 & & 43 & 44 \end{pmatrix}$$

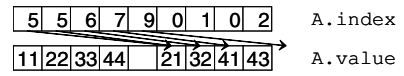

図 5 Data structure of MSR.

2 プロセス上への MSR 形式での格納方法を図 6 に示す．

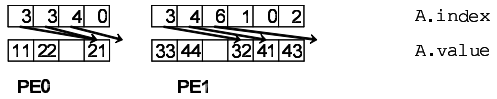

図 6 Data structure of MSR on two processes.

3.4 Diagonal (DIA)

DIA は行列データを 2 つの配列 (`index, value`) に格納する．nnd を行列 A の非零な対角の本数とする．

- 長さ $nnd \times n$ の倍精度配列 `value` は行列 A の非零な対角を格納する．
- 長さ nnd の整数配列 `index` は主対角から各対角へのオフセットを格納する．

OpenMP 版では以下のように修正している．

Vol.2009-ARC-186 No.7
Vol.2009-HPC-123 No.7
2009/12/1

DIA は 2 つの配列 (index, value) に格納する．$nprocs$ をスレッド数とする．nnd_p を行列 A を行ブロック分割した部分行列の非零な対角の本数とする．$maxnnd$ を nnd_p の値の最大値とする．

- 長さ $maxnnd \times n$ の倍精度配列 value は行列 A を行ブロック分割した部分行列の非零な対角を格納する．
- 長さ $nprocs \times maxnnd$ の整数配列 index は主対角から各対角へのオフセットを格納する．

DIA 形式での格納方法を図 7 に示す.

$$A = \begin{pmatrix} 11 & & & \\ 21 & 22 & & \\ & 32 & 33 & \\ 41 & & 43 & 44 \end{pmatrix}$$

-3	-1	0										
0	0	0	41	0	21	32	43	11	22	33	44	

A.index
A.value

図 7 Data structure of DIA.

2 スレッド上への DIA 形式での格納方法を図 8 に示す.

-1	0		-3	-1	0						
0	21	11	22			0	41	32	43	33	44

A.index
A.value

図 8 Data structure of DIA on two threads.

2 プロセス上への DIA 形式での格納方法を図 9 に示す.

-1	0			-3	-1	0				
0	21	11	22	0	41	32	43	33	44	
PE0				PE1						

A.index
A.value

図 9 Data structure of DIA on two processes.

3.5 Ellpack-Itpack generalized diagonal (ELL)

ELL は行列データを 2 つの配列 (index, value) に格納する．$maxnzr$ を行列 A の各行での非零要素数の最大値とする．

- 長さ $maxnzr \times n$ の倍精度配列 value は行列 A の各行の非零要素を列方向に沿って格納する．最初の列は各行の最初の非零要素からなる．ただし，格納する非零要素数がない場合は 0 を格納する．
- 長さ $maxnzr \times n$ の整数配列 index は配列 value に格納された非零要素の列番号を格納する．ただし，第 i 行目の非零要素数を nnz_i とすると index[nnz_i*n + i] にはその行番号 i を格納する．

ELL 形式での格納方法を図 10 に示す.

$$A = \begin{pmatrix} 11 & & & \\ 21 & 22 & & \\ & 32 & 33 & \\ 41 & & 43 & 44 \end{pmatrix}$$

0	0	1	0	0	1	2	2	0	1	2	3
11	21	32	41	0	22	33	43	0	0	0	44

A.index
A.value

図 10 Data structure of ELL.

2 プロセス上への ELL 形式での格納方法を図 11 に示す.

0	0	0	1		1	0	2	2	2	3
11	21	0	22		32	41	33	43	0	44
PE0					PE1					

A.index
A.value

図 11 Data structure of ELL on two processes.

3.6 Jagged Diagonal (JDS)

JDS は最初に各行の非零要素数の大きい順に行の並び替えを行い，各行の非零要素を列方向に沿って格納する．JDS は行列データを 4 つの配列 (perm, ptr, index, value) に格納する．$maxnzr$ を行列 A の各行での非零要素数の最大値とする．

- 長さ n の整数配列 perm は並び替えた行番号を格納する．
- 長さ nnz の倍精度配列 value は並び替えられた行列 A のぎざぎざ対角を格納する．最初のぎざぎざ対角は各行の最初の非零要素からなる．次のぎざぎざ対角は各行の 2 番目の非零要素からなる．これを順次繰り返していく．
- 長さ nnz の整数配列 index は配列 value に格納された非零要素の列番号を格納する．
- 長さ $maxnzr + 1$ の整数配列 ptr は各ぎざぎざ対角の開始位置を格納する．

OpenMP 版では以下のように修正を行っている．
JDS は 4 つの配列 (perm, ptr, index, value) に格納する．$nprocs$ をスレッド数とする．

Vol.2009-ARC-186 No.7
Vol.2009-HPC-123 No.7
2009/12/1

$maxnzr_p$ を行列 A を行ブロック分割した部分行列の各行での非零要素数の最大値とする．$maxmaxnzr$ は配列 $maxnzr_p$ の値の最大値である．

- 長さ n の整数配列 `perm` は行列 A を行ブロック分割した部分行列を並び替えた行番号を格納する．
- 長さ nnz の倍精度配列 `value` は並び替えられた行列 A のぎざぎざ対角を格納する．最初のぎざぎざ対角は各行の最初の非零要素数からなる．次のぎざぎざ対角は各行の 2 番目の非零要素からなる．これを順次繰り返していく．
- 長さ nnz の整数配列 `index` は配列 `value` に格納された非零要素の列番号を格納する．
- 長さ $nprocs \times (maxmaxnzr + 1)$ の整数配列 `ptr` は行列 A を行ブロック分割した部分行列の各ぎざぎざ対角の開始位置を格納する．

JDS 形式での格納方法を図 12 に示す．

$$A = \begin{pmatrix} 11 & & & \\ 21 & 22 & & \\ & 32 & 33 & \\ 41 & & 43 & 44 \end{pmatrix}$$

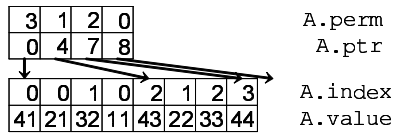

図 12 Data structure of JDS.

2 スレッド上への JDS 形式での格納方法を図 13 に示す．

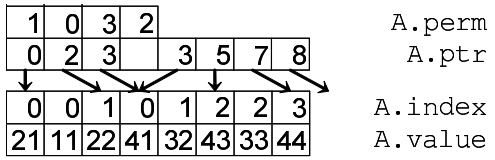

図 13 Data structure of JDS on two threads.

2 プロセス上への JDS 形式での格納方法を図 14 に示す．

3.7 Block Sparse Row (BSR)

BSR では行列を $r \times c$ の大きさの部分行列（ブロックと呼ぶ）に分解する．BSR は CRS と同様の手順で非零ブロック（少なくとも 1 つの非零要素が存在する）を格納する．$nr = n/r$, $nnzb$ を A の非零ブロック数とする．BSR は行列データを 3 つの配列 (`bptr, bindex, value`) に格納する．

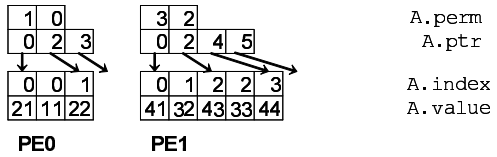

図 14 Data structure of JDS on two processes.

- 長さ $nnzb \times r \times c$ の倍精度配列 `value` は非零ブロックの全要素を格納する．
- 長さ $nnzb$ の整数配列 `bindex` は非零ブロックのブロック列番号を格納する．
- 長さ $nr + 1$ の整数配列 `bptr` は配列 `bindex` のブロック行の開始位置を格納する．

BSR 形式での格納方法を図 15 に示す．

$$A = \left(\begin{array}{cc|cc} 11 & & & \\ 21 & 22 & & \\ \hline & 32 & 33 & \\ 41 & & 43 & 44 \end{array}\right)$$

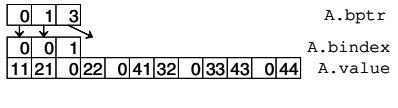

図 15 Data structure of BSR.

2 プロセス上への BSR 形式での格納方法を図 16 に示す．

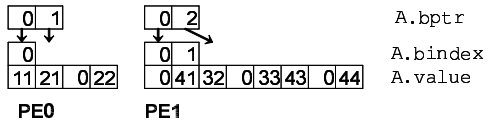

図 16 Data structure of BSR on two processes.

3.8 Block Sparse Column (BSC)

BSC では行列を $r \times c$ の大きさの部分行列（ブロックと呼ぶ）に分解する．BSC は CCS と同様の手順で非零ブロック（少なくとも 1 つの非零要素が存在する）を格納する．$nc = n/c$, $nnzb$ を A の非零ブロック数とする．BSC は行列データを 3 つの配列 (`bptr, bindex, value`) に格納する．

- 長さ $nnzb \times r \times c$ の倍精度配列 `value` は非零ブロックの全要素を格納する．

Vol.2009-ARC-186 No.7
Vol.2009-HPC-123 No.7
2009/12/1

- 長さ $nnzb$ の整数配列 `bindex` は非零ブロックのブロック行番号を格納する．
- 長さ $nc+1$ の整数配列 `bptr` は配列 `bindex` のブロック列の開始位置を格納する．

BSC 形式での格納方法を図 17 に示す．

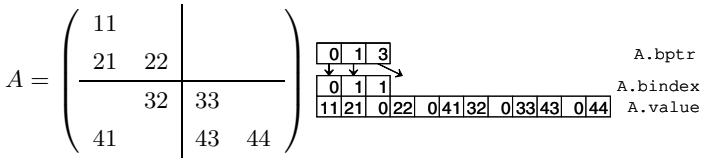

図 17 Data structure of BSC.

2 プロセス上への BSC 形式での格納方法を図 18 に示す．

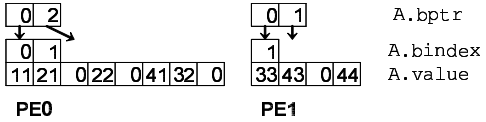

図 18 Data structure of BSC on two processes.

3.9 Variable Block Row (VBR)

VBR 形式は BSR 形式を一般化したものである．行と列の分割位置は配列 (`row, col`) で与えられる．VBR は CRS と同様の手順で非零ブロック（少なくとも 1 つの非零要素が存在する）を格納する．nr, nc をそれぞれ行分割数，列分割数とする．$nnzb$ を A の非零ブロック数，nnz を非零ブロックの全要素数とする．VBR は行列データを 6 つの配列 (`bptr, bindex, row, col, ptr, value`) に格納する．

- 長さ $nr+1$ の整数配列 `row` はブロック行の開始行番号を格納する．
- 長さ $nc+1$ の整数配列 `col` はブロック列の開始列番号を格納する．
- 長さ $nnzb$ の整数配列 `bindex` は非零ブロックのブロック列番号を格納する．
- 長さ $nr+1$ の整数配列 `bptr` は配列 `bindex` のブロック行の開始位置を格納する．
- 長さ nnz の倍精度配列 `value` は非零ブロックの全要素を格納する．
- 長さ $nnzb+1$ の整数配列 `ptr` は配列 `value` の非零ブロックの開始位置を格納する．

VBR 形式での格納方法を図 19 に示す．

2 プロセス上への VBR 形式での格納方法を図 20 に示す．

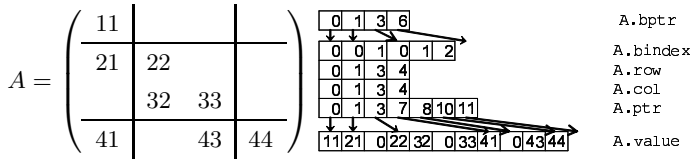

図 19 Data structure of VBR.

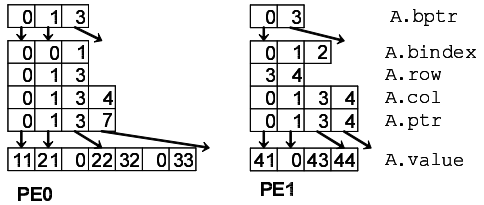

図 20 Data structure of VBR on two processes.

3.10 Coordinate (COO)

COO は行列データを 3 つの配列 (`row, col, value`) に格納する．

- 長さ nnz の倍精度配列 `value` は非零要素を格納する．
- 長さ nnz の整数配列 `row` は非零要素の行番号を格納する．
- 長さ nnz の整数配列 `col` は非零要素の列番号を格納する．

COO 形式での格納方法を図 21 に示す．

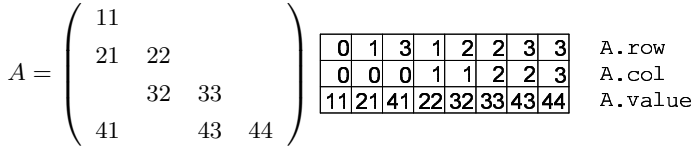

図 21 Data structure of COO.

2 プロセス上への COO 形式での格納方法を図 22 に示す．

3.11 Dense (DNS)

DNS は行列データを 1 つの配列 (`value`) に格納する．

- 長さ $n \times n$ の倍精度配列 `value` は列優先で要素を格納する．

Vol.2009-ARC-186 No.7
Vol.2009-HPC-123 No.7
2009/12/1

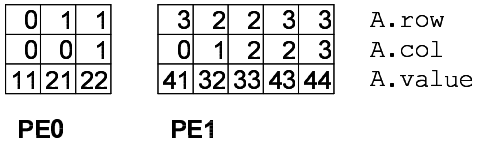

図 22 Data structure of COO on two processes.

2 プロセス上への DNS 形式での格納方法を図 23 に示す.

$$A = \begin{pmatrix} 11 & & & \\ 21 & 22 & & \\ & 32 & 33 & \\ 41 & & 43 & 44 \end{pmatrix}$$

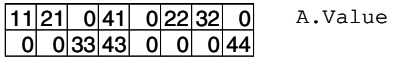

図 23 Data structure of DNS.

2 プロセス上への DNS 形式での格納方法を図 24 に示す.

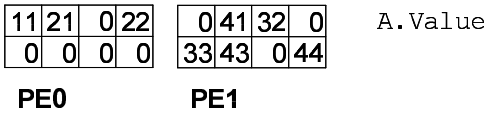

図 24 Data structure of DNS on two processes.

4. 性 能 評 価

Lis 上に実装した固有値解法の特性について調べるため, 九州大学情報基盤研究開発センターに設置された Intel Xeon 5570 サーバ (2.93GHz クアッドコアプロセッサ ×2), 日立 SR16000 サーバ (4.7GHz IBM デュアルコア POWER6 プロセッサ ×16) の DDR InfiniBand クラスタ, 及び東北大学サイバーサイエンスセンターの SX-9 1 ノードを用い,

今回は, Lis 上に実装した疎行列ベクトル積に関するベンチマークプログラム `spmvtest1`, `spmvtest2` にて性能評価を行った. `spmvtest1`, `spmvtest2` の仕様は以下の通りである.

4.0.1 spmvtest1

次数 `n` の三重対角行列

$$A = \begin{pmatrix} 2 & 1 & & & \\ 1 & 2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 2 & 1 \\ & & & 1 & 2 \end{pmatrix}$$

とベクトル $(1, \ldots, 1)^T$ との積を実行可能な行列格納形式について `iter` で指定された回数実行し, MFLOPS 値を算出する.

4.0.2 spmvtest2

2 次元 Poisson 方程式を 5 点中心差分で離散化した行列とベクトル $(1, \ldots, 1)^T$ との積を実行可能な行列格納形式について `iter` で指定された回数実行し, MFLOPS 値を算出する. `m` と `n` はそれぞれ垂直方向と水平方向の格子点数である.

1-6 に, 上記アーキテクチャにおいて高い性能を示した CRS, DIA, ELL, JDS 形式での MPI 版, OpenMP 版 spmvtest1, spmvtest2 の評価結果を示す.

これらのデータから, 計算機アーキテクチャによって最適な格納形式, 並列化手法が異なること, またこれらの性能が行列データの構造にも依存することなどが見て取れる.

表 1 Performance of spmvtest1 on Xeon 5570 server.

Problem Size	40,000	80,000	160,000	320,000
# MPI processes	1	2	4	8
CRS (MFLOPS)	884	1740	3460	5790
DIA (MFLOPS)	1510	2990	5470	6500
ELL (MFLOPS)	1200	2370	4500	4960
JDS (MFLOPS)	840	1660	3150	3080
Problem Size	40,000	80,000	160,000	320,000
# threads	1	2	4	8
CRS (MFLOPS)	884	1690	3370	4890
DIA (MFLOPS)	1510	2970	5470	6440
ELL (MFLOPS)	1200	2350	4570	5270
JDS (MFLOPS)	840	1660	3150	2450

Vol.2009-ARC-186 No.7
Vol.2009-HPC-123 No.7
2009/12/1

表 2 Performance of spmvtest2 on Xeon 5570 server.

Problem Size	40,000	80,000	160,000	320,000
# MPI processes	1	2	4	8
CRS (MFLOPS)	1110	2190	4140	4440
DIA (MFLOPS)	1630	3190	5010	4370
ELL (MFLOPS)	1210	2400	4270	3770
JDS (MFLOPS)	1030	2030	3370	3040
Problem Size	40,000	80,000	160,000	320,000
# threads	1	2	4	8
CRS (MFLOPS)	1190	2380	2160	3780
DIA (MFLOPS)	1640	3130	6040	4940
ELL (MFLOPS)	1250	2440	4350	3560
JDS (MFLOPS)	1020	2030	3070	2860

表 3 Performance of spmvtest1 on Hitachi SR16000.

Problem Size	40,000	80,000	160,000	320,000	640,000	1,280,000
# MPI processes	1	2	4	8	16	32
CRS (MFLOPS)	310	484	976	1960	3730	7760
DIA (MFLOPS)	1800	1940	3900	7800	15600	31200
ELL (MFLOPS)	1530	1490	2130	6070	12100	24000
JDS (MFLOPS)	1150	1090	1480	4270	8450	17100
# threads	1	2	4	8	16	32
CRS (MFLOPS)	310	607	1210	2420	4790	9380
DIA (MFLOPS)	1800	3270	6470	12600	25900	43000
ELL (MFLOPS)	1530	2880	5620	11100	21000	40700
JDS (MFLOPS)	1150	2090	4080	7810	15100	29300

表 4 Performance of spmvtest2 on Hitachi SR16000.

Problem Size	40,000	80,000	160,000	320,000	640,000	1,280,000
# MPI processes	1	2	4	8	16	32
CRS (MFLOPS)	580	851	1680	3270	6430	13400
DIA (MFLOPS)	2130	2370	4000	7730	14900	30600
ELL (MFLOPS)	1790	1750	2440	4860	9490	19100
JDS (MFLOPS)	1390	1330	2020	4020	7960	16200
# threads	1	2	4	8	16	32
CRS (MFLOPS)	580	1090	2210	4390	8740	15100
DIA (MFLOPS)	2130	3930	7500	15000	28700	44100
ELL (MFLOPS)	1790	2930	5900	11500	23000	33800
JDS (MFLOPS)	1390	2250	4550	9060	17600	27300

表 5 Performance of spmvtest1 on NEC SX-9.

Problem Size	40,000	80,000	160,000	320,000	640,000
# MPI processes	1	2	4	8	16
CRS (MFLOPS)	5.39	10.8	21.6	43.3	-
DIA (MFLOPS)	9780	11600	20800	42600	-
ELL (MFLOPS)	1320	2380	4730	9270	-
JDS (MFLOPS)	835	1510	3010	5860	-
# threads	1	2	4	8	16
CRS (MFLOPS)	5.44	11.0	14.4	43.7	87.4
DIA (MFLOPS)	8930	9460	18000	35600	64700
ELL (MFLOPS)	1280	2300	4470	9150	18000
JDS (MFLOPS)	828	1550	1040	6160	12100

Vol.2009-ARC-186 No.7
Vol.2009-HPC-123 No.7
2009/12/1

表 6 Performance of spmvtest2 on NEC SX-9.

Problem Size	40,000	80,000	160,000	320,000	640,000
# MPI processes	1	2	4	8	16
CRS (MFLOPS)	13.2	26.5	52.7	106	-
DIA (MFLOPS)	10400	12100	16000	29600	-
ELL (MFLOPS)	1310	2410	4450	8720	-
JDS (MFLOPS)	1070	1890	3460	6340	-
# threads	1	2	4	8	16
CRS (MFLOPS)	13.4	27.0	53.6	-	212
DIA (MFLOPS)	11700	15500	28300	-	100000
ELL (MFLOPS)	1310	2530	4900	-	19500
JDS (MFLOPS)	1080	2050	4090	-	16000

5. 考　　察

本稿では，現在開発を進めている並列反復解法ライブラリ Lis への適用を目的として，複数アーキテクチャ上での疎行列ベクトル積の事前性能評価の妥当性について検討した．疎行列ベクトル積の性能は，行列の形状，並列化手法，メモリの階層構造等によって大きく性能が変化するため，すべての場合に最適な解法を見出すのは難しい．したがって，現実的な解として，本研究で試みた局所的な性能解析を一般化し，より多様なデータ格納手法について評価を可能にすることが考えられる．また，解析手法の自動化により，性能可搬性の確保を容易にすることも重要である．本研究では，今後これらの手法について，実装，評価を行っていく予定である．

参 考 文 献

1) Dongarra, J.: Freely Available Software for the Solution of Linear Algebra Problems, http://www.netlib.org/utk/people/JackDongarra/la-sw.html (2009).
2) The Scalable Software Infrastructure Project: Lis 1.2.15 User's Manual, http://www.ssisc.org/lis/ (2009).