自動チューニング機能付き並列スパースダイレクトソルバの性能評価 – チューニング情報からの知識発見 –

Performance Evaluation of an Automatically Tuned Parallel Sparse Direct LU Factorization Routine

大澤 清 [¶]　片桐 孝洋 [¶,†]　黒田 久泰 [‡]　金田 康正 [‡]

OSAWA Kiyoshi　KATAGIRI Takahiro　KURODA Hisayasu　KANADA Yasumasa

[¶] 東京大学大学院理学系研究科情報科学専攻　[†] 日本学術振興会特別研究員

[‡] 東京大学情報基盤センタースーパーコンピューティング部門

Department of Information Science, Graduate School of Science, the University of Tokyo

Research Fellow of the Japan Society for the Promotion of Science

Computer Centre Division, Information Technology Center, the University of Tokyo

Abstract: In this paper, we report functions and performance for parallel LU factorization routine (ILIB_RLU) as a system of linear equations solver. It is one of I-LIB (Intelligent LIBrary) routines which include auto-tuning facilities. In the use of ILIB_RLU, users do not need to care precisions of answers, since coefficient matrices are treated as dense matrices and the equations are solved by a direct method. ILIB_RLU reduces computation area in order to attain speed-up. The auto-tuning facilities optimize performance without specifying parameters. Both HITACHI SR8000 and HITACHI SR2201 which are distributed memory parallel machines are used in our performance evaluation. From the experimental results, we found that ILIB_RLU routines attains 72.5% efficiency to theoretical peak performance and 39.0 times speed-up to normal LU factorization for the real problem size of 41296 by our auto-tuning facilities. In addition, we investigate knowledge found on ILIB_RLU and discuss how to apply the knowledge for auto-tuning facilities.

我々は科学技術計算用ライブラリ群，特に並列数値計算において，以下に示す特徴・機能を有する数値計算ライブラリの開発を目指している．

- ユーザが指定するパラメタが少ないこと
- 演算カーネルに関するチューニング機能があること
- 通信処理に関するチューニング機能があること（並列計算機を用いる場合）
- 利用するアルゴリズムに関する自動選択機能があること

これらの設計思想に基づいて開発された I-LIB[1] は，チューニング機能を含む性能に関する最適化をユーザが直接指定することなしにライブラリ自体が行うという概念を実現したライブラリである．今回開発した ILIB_RLU では，これらの機能に加えて係数行列の構造を調べることにより演算量を削減し，計算の高速化を図っている．なお ILIB_RLU の R は演算量の減少 (Reduced) に由来している．

大澤 清 東京大学 情報基盤センター 113-8658 Tel (03)5841-2717 Fax (03)5841-2708 e-mail: ohsawa@pi.cc.u-tokyo.ac.jp

ILIB_RLU は主に並列スパースダイレクトソルバとして用いられるように設計されている．並列スパースダイレクトソルバに関する研究は数多く行われており [2]，また様々な手法が提案され高速化が図られているが，その性能が計算機の理論ピーク性能に近い値を示しているものは少なく，さらに高速なソルバが要求されている．
ILIB_RLU はそういった要求にできるだけ応えようとして設計された． ILIB_RLU の特徴として

- 自動チューニング機能をもつ
- 疎行列を密行列と同様に直接法によって解くことで計算の高速化を図る
- 密行列インタフェースとなるため利用しやすい
- メモリ利用は非効率的になる

などが挙げられる．

自動チューニングの機能をもつ数値計算ソフトウエアとしては PHiPAC[3] や ATLAS[4] などが挙げられる．しかしこれらのソフトウエアは並列計算を考慮に入れていなかったり，行列積などの比較的簡単な計算の最適化の

みを行うだけである．そこで我々は，実用的な並列数値計算ライブラリを念頭に置き，I-LIB プロジェクトとして並列対称密行列固有値ライブラリ [5]，並列三重対角化ライブラリ [6]，並列疎行列連立一次方程式ライブラリ [7][8] の開発を行ってきた．

本稿の構成は以下の通りである．まず第 1章で，今回開発した自動チューニング機能付き並列 LU 分解ルーチン ILIB_RLU の概要について説明する．次に第 2章で，具体的なチューニング機能を紹介する．第 3章では，分散メモリ型並列計算機である HITACHI SR8000 と HITACHI SR2201 を用いて ILIB_RLU の性能を評価した結果を示す．最後に，第 5章で本論文のまとめを述べる．

1 ILIB_RLU の概要

1.1 自動チューニング

I-LIB における自動チューニング機能としては以下のものが挙げられる．

(1) 機種，アーキテクチャに依存するチューニング（アンローリング段数，通信方式など）

(2) 問題に依存するチューニング

(3) アルゴリズムに関するチューニング

これらのうち ILIB_RLU によって実装されているのは (1) と (2) に分類されるチューニングである．

1.2 並列疎ソルバとしての ILIB_RLU

三次元物体の構造解析などを行う際に得られる大規模連立一次方程式を解くために，高速なソルバの需要が高まっている．我々が開発している並列連立一次方程式ライブラリは，式（1）に示される連立一次方程式の解ベクトル x を求めることができる．

$$Ax = b \tag{1}$$

ここで，係数行列 $A \in \Re^{n \times n}$ は実数の非対称密行列もしくは非対称疎行列である．また右辺ベクトル b は $b \in \Re^n$ であり，解ベクトル x は $x \in \Re^n$ である．式（1）の解法として直接解法と反復解法の 2 種類が存在するが，今回は密行列，疎行列とも直接解法で解くこととする．

構造解析など実際の応用問題で得られる連立一次方程式の係数行列は疎行列になることが多く，その場合は反復解法により解が求められることが多い．しかし係数行列が対角優位でなく悪条件の場合などは反復解法を用いると解が収束しなかったり，また収束したとしても反復回数が非常に増大する場合がある．一方で最近の計算機のメモリの増加を考慮すると，反復解法に比べてメモリ利用の点で不利ではあるが，かなり大規模なサイズの問題が精度の点で有利な直接解法でも解けるようになってきているため，直接解法で疎行列を扱うことにした．解法が直接解法のため，当然 ILIB_RLU は密行列も扱うことができる．

2 ILIB_RLU のもつチューニング機能の紹介

並列 LU 分解ルーチンでのチューニングとして，以下の機能を用意した．

- ブロックアルゴリズムにおける最適ブロック幅（多段多列同時消去法における段数 BH と列数 BW）の探索
- 係数行列の非零構造からの演算範囲限定
- 零要素による更新演算の省略
- データ分散方法の最適化

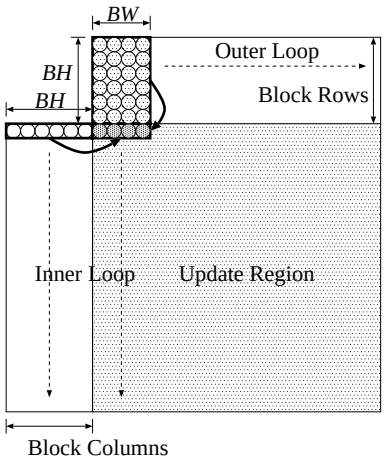


図 1: 多段多列同時消去法

2.1 多段多列同時消去法

ILIB_RLU では LU 分解を外積形式ガウス法により行っている．その更新演算にはブロックアルゴリズム [9] とよばれる手法を用い，レジスタの同時利用によりその性能向上を図っている．そのためこの手法は機種，アーキテクチャに依存するチューニングに分類される．図 1はブロック幅を示し，それらは BH，BW の 2 種類のパラメタで表されている．

実際の演算は図 1の Block Rows 内の $BH \times BW$ で示される長方形ブロック内の要素と Block Columns 内の BH 個の要素との積和演算となり，Update Region 内の各要素が自身から BH 個の積を減算している．これによりメモリへのストア回数が減ることになり，同時参照可能なレジスタ数に応じて演算速度が向上する．

最も演算効率を高める BH，BW の組合せは使用する並列計算機のアーキテクチャやコンパイラの最適化機能に大きく依存するため，実際にプログラムを実行して決定している．具体的には BH，$BW = \{1, 2, 3, ..., 20\}$ とし（SR2201 の場合は $BH, BW = \{1, 2, 3, ..., 16\}$），この $20 \times 20 = 400$ 通り（同，256 通り）の組合せから最適な（BH_{opt}, BW_{opt}）を決定する．ここで，この自動チューニングはライブラリをインストールする時に 1 度行えば，利用するプロセッサ台数や並列計算機環境が変わらなければ再度行う必要がないことに注意しておく．すなわち，自動チューニングは静的（ライブラリ実行前）に行われる．

実際のプログラムにおいては，BH，BW はアンローリング段数として扱われる．例として，3 段 2 列（$BH = 3, BW = 2$）同時消去を行うカーネル部分を以下に示す．N は行列サイズ，bc は Block Columns 内の要素を示している．ここで，N は BH で割り切れるものとする．

```
do j=1, N, BH
  (Pivoting)
  jj=j+BH
  nblock=(N-(jj-1))/BW
  do jb=1, nblock
    (Updating elements in Block Rows)
    do i=jj, N
      bc1=bc(i, 1)
      bc2=bc(i, 2)
      bc3=bc(i, 3)
      a(i, jj  )=a(i, jj  )
```

```
&            -bc1*a(j  , jj  )
&            -bc2*a(j+1, jj  )
&            -bc3*a(j+2, jj  )
       a(i, jj+1)=a(i, jj+1)
&            -bc1*a(j  , jj+1)
&            -bc2*a(j+1, jj+1)
&            -bc3*a(j+2, jj+1)
     enddo
     jj=jj+BW
   enddo
   (Updating elements in remainder region)
 enddo
```

以上のようなカーネル部分をもつコードをアンローリング段数に応じて (BHの候補数) × (BWの候補数) 通りも手作業で書くのは大変な手間がかかる．そのためその部分のコード生成は ILIB_RLU 用のカーネル部分コード自動生成ルーチンを用いて行っている．以下にその概要を示す．

2.1.1 コード自動生成ルーチン

コード自動生成ルーチン
ILIB_LUkergen は ILIB_RLU 内のカーネル部分とアンローリング段数，データ分散幅の定数宣言部を自動生成する．実際には ILIB_RLU の本来カーネル部分，定数宣言部が記述されるべき位置に，それらの代わりに指示行が記述されたファイルを用意しておき，各パラメタを指定することで上に示したような BH と BW を反映したコードが生成される． ILIB_LUkergen の呼び出しは

ILIB_LUkergen $PLAINFILE$ BH BW DW $OUTFILE$

で行われる． $PLAINFILE$ は指示行が記述されたファイル名， BH と BW は前述のアンローリング段数， DW はデータ分散幅， $OUTFILE$ は出力ファイル名である．
具体的には， $PLAINFILE$ の同時多段多列消去を開始する位置 ((Updating elements in Block Rows) にあたる位置) に指示行として

```
!ILIB_LUkergen_main
```

と記しておくことで先に示したような Block Rows 内と Update Region 内の更新演算が自動生成される．また，Update Region のブロック幅 BW で割り切れなかった領域を更新するために

```
!ILIB_LUkergen_remd
```

という指示行を (Updating elements in remainder region) の位置に記述する．

2.2 非零構造からの演算範囲限定

構造解析の分野でしばしば用いられる疎行列は対角要素に非零要素が集中していることが多く，その場合対角要素以外の要素に対して乗算を行うとほとんどの場合零要素との乗算になり，無駄な演算を行っていることになる．そこで非零要素の構造を特定するインデックスとして，各行，各列の非零要素の終端位置を表す配列を用意した．具体的には

$jlast(i) =$ (第i行の非零要素の最大列番号)
$ilast(j) =$ (第j列の非零要素の最大行番号)

とした．実際には計算誤差を抑えるためのピボット処理により計算が進むにつれて非零要素の構造は変化し，その際のインデックスの変化を最小にするために $jlast(i) \geq jlast(i-1), ilast(j) \geq ilast(j-1)(i,j = 2,3,...,$ 行列サイズ$N)$ とする． $jlast(i), ilast(j)$ で示される要素が第 i 行，第 j 列の計算対象領域の終端となる（図 2）．

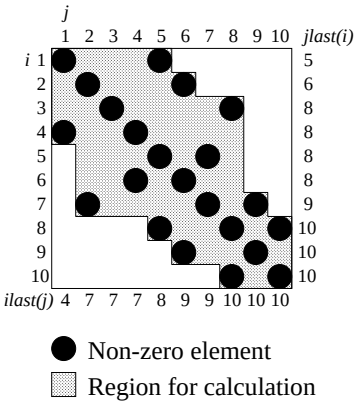


図 2: 非零構造からの演算範囲限定

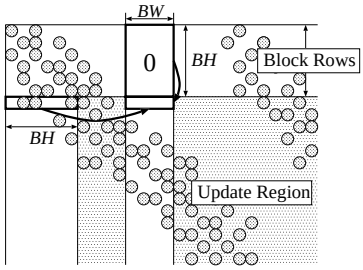


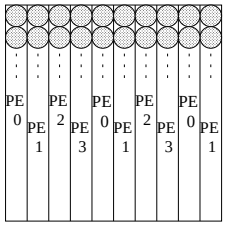
図 3: 零要素による更新演算の省略

ピボット処理により第 i 行と第 k 行が交換される場合，$jlast(I) = jlast(k)(I = i+1, i+2, ..., k-1)$ と更新する． $ilast(j)$ についてはこのような更新を行う必要はない．以上のようにして無駄な領域の演算を行わないようにする．この手法は問題に依存するチューニングに分類される．

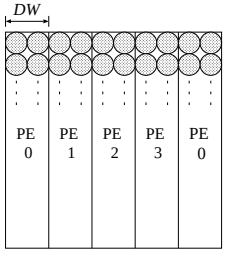
非対称行列のサイズを N とすると，通常の LU 分解では $2 \times N^3/3$ の演算量を必要とするが，演算範囲を限定した場合，用いる行列を帯行列とみなして半帯幅 $\max_i(jlast(i) - i)$ を hbw とすると演算量は最悪でも $4 \times hbw^2 \times N$ であり，例えば $hbw = N/5$ とするとその演算量は $4 \times N^3/25$ に抑えられ，通常の場合の 24% にまで減少することになる．

2.3 零要素による更新演算の省略

疎行列において，ある行の対角要素とその右端の要素との間に非零要素がほとんど存在しない場合を考える．このときブロックアルゴリズムで LU 分解を行うと，ブロック行（Block Rows）内の $BH \times BW$ の領域がすべて 0 になり，更新演算がすべて零要素との乗算になるため更新領域（Update Region）内の演算が不要になる場合がある（図 3）． ILIB_RLU では，更新演算を開始する前に図 3の $BH \times BW$ の長方形ブロックがすべて 0 であるかを調べて，すべて 0 の場合には更新領域内の演算を行わないようにして速度向上を図っている．この手法は問題依存のチューニングに分類される．



Column-cyclic Distribution



Block-column-cyclic Distribution

図 4: データ分散方法

2.4 データ分散方法の最適化

データ分散方式として列サイクリック分散（Column-cyclic Distribution）（図 4）を採用して LU 分解を行ったとき，一列の消去演算ごとに通信を行う必要があり，問題が比較的小さく実行時間内に通信時間の占める比率が比較的大きな場合に不利になる．そこで列サイクリックに幅を持たせたブロック列サイクリック方式で分散を行った場合，単純に計算すれば通信回数は 1/(分散ブロック幅) に減少し性能が向上すると考えられる．分散ブロック幅は図 4のブロック列サイクリック分散における DW で表されている．列サイクリック分散に比べてロードバランスの点でブロック列サイクリック分散は不利になるが，通信回数削減の効果に比べてその影響は小さいと考えられる．

しかし最適な DW を一概に決定するのは難しく，多段多列消去段数の決定と同様に実際に実行して決定する必要がある．この場合データの分散ブロック幅 DW とブロックアルゴリズムとしてのブロック幅 BH，BW とを独立に決定するとなると，分散ブロック幅の候補それぞれについて先の 400 通り (SR2201 では 256 通り) の組み合わせを試すことになり，膨大な量の時間とプログラムが必要になる．そこで同時参照可能なレジスタの数を予想するなどして明らかに無駄と判断される BH，BW の組合せのブロックアルゴリズムは省略して決定するのが現実的である．2000 年 9 月 1 日現在の ILIB_RLU では，DW は可変ではあるが最適化は行われていない．

3 性能評価

この章では，ILIB_RLU を HITACHI SR8000 および HITACHI SR2201 を用いて評価した結果を示す．

本評価で使用した SR8000 の各ノードの理論ピーク性能は 8GFLOPS である．その各ノードは 1GFLOPS の IP (Instruction Processor)8 台の共有メモリ構成となっている．ノード間は三次元ハイパクロスバ網で結合されており，その最大転送性能は片方向で 1Gbyte/ 秒，双方向で 2Gbyte/ 秒である．また，SR2201 の各 PE の理論ピーク性能は 300MFLOPS である．PE 間は三次元ハイパクロスバ網で結合されており，その最大転送性能は 300Mbyte/ 秒である．なお，通信ライブラリとしては MPI(Message Passing Interface) を利用した．現在はデータ分散幅 DW は 40 に固定している．表 1，表 2と図 5に使用した係数行列の情報を示す．この行列は人工心臓の流体解析に用いられたものであり，これより得られる連立一次方程式を科学技術用並列行列計算ライブラリ PETSc[10] の GMRES(m) 法ルーチン (Additive Schwarz Method 前処理，リスタート周期 768，停止残差 0.855e-7，問題サイズ 11608) によって SR2201 で解いた場合，23040 反復で 3358.0 秒を必要とした [11]. したがってこの行列は非常に収束の悪い行列といえる．ILIB_RLU によって得られた残差ベクトルの 2- ノルムは 0.451e-13 となった．

表 1: 使用した係数行列，次元数 $N = 41296$

	非零要素の幅（対角要素からの距離）	一行当たりの非零要素数
平均	2334.0	67.6
最大	4007	107
最小	43	7

表 2: 使用した係数行列，次元数 $N = 11608$

	非零要素の幅（対角要素からの距離）	一行当たりの非零要素数
平均	1254.2	63.3
最大	2179	107
最小	33	7

3.1 多段多列同時消去の効果

ILIB_RLU における多段多列同時消去に関するパラメタは

- 同時消去段数 BH={1,2,3,...,20}
- 同時消去列数 BW={1,2,3,...,20}
  (SR2201 は BH, BW={1,2,3,...,16})

の 2 種類とした．表 3，表 4は SR8000，SR2201 における，自動チューニングにより最適化したパラメタの組合せを用いた実行時間と，文献 [9] より妥当と思われるパラメタの組合せ（BH, BW）＝（8, 4）（SR2201 の場合は（5, 2））を設定した場合の実行時間との比較を示している．表 3より ILIB_RLU は SR8000 の 1 ノード (8IP) の

東京大学情報基盤センターが所有している 128 ノード（8IP/1 ノード）の SR8000 のうち 2 ノード –16 ノードを使用した．また，コンパイラとして日立の最適化 FORTRAN90V01-00，オプションとして，-W0,'opt(o(ss),expand(0)),mp(p(4),diag(1))' を指定した．

東京大学情報基盤センターが所有している 1024PE の SR2201 のうち 16PE を使用した．またコンパイラとして FORTRAN では，日立の最適化 FORTRAN90 V02-06-/D，オプションとして，-W0,'PVEC(PVFUNC(1),VERCHK(0),DIAG(1)), opt(o(s),expand(0),fold(2),prefetch(1),rapidcall(1),ischedule(3), reroll(1),scope(1),split(2),uinline(2))' を指定した．

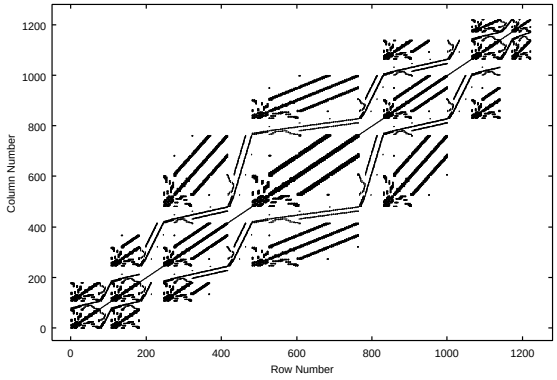


図 5: 使用した係数行列 (粗いメッシュで構成したモデル, 問題サイズ $N = 1222$, 非対称行列)

表 3: 多段多列同時消去の効果 (SR8000, 2 ノード (16IP), $DW = 40$)

(a) 問題サイズ: 41296

最適化	段数 BH	列数 BW	実行時間 [秒]	GFLOPS /PE	効率 [%]
あり	10	9	84.18	5.80	72.5
なし	8	4	129.47	3.77	47.1

(b) 問題サイズ: 11608

最適化	段数 BH	列数 BW	実行時間 [秒]	GFLOPS /PE	効率 [%]
あり	10	9	8.05	4.98	62.2
なし	8	4	11.27	3.55	44.4

ピーク性能 (8GFLOPS) に対する効率 72.5% を達成しており, スパースソルバとしては計算機の性能を十分に引き出しているといえる. なお, ここで効率は理論最大性能に対する実効性能である. 図 6, 図 7に SR8000 で最適な BH, BW を求めるために全探索を行った結果を示す. これらの図より, $BH \times BW = 100$ となる双曲線付近に最適に近いパラメータが分布していることが見てとれる.

3.2 演算範囲限定の効果

ILIB_RLU の演算範囲限定による高速化の結果を示す. 範囲を限定しない場合については ILIB_LU(通常の LU 分解ルーチン) を用いて計算を行った. 表 5にその結果を示す.

ILIB_RLU は FLOPS 値の点で ILIB_LU に及ばない. これは範囲を限定したことによってベクトル長が短くなったり, インデックス参照が増えたことが原因と考えられる. しかし, 全体の演算量が削減されるため当然通常の場合に比べて大幅に計算時間は短縮されている.

3.3 更新演算省略の効果

零要素との乗算を行う更新演算の省略を行った場合と, 行わなかった場合の結果を表 6に示す. SR2201 でこの機能を用いた場合は速度向上比にして 3.34% の高速化が図られたが, SR8000 ではほとんど差が現れず, むしろ機能を使わない場合が高速の場合もあった. このチューニング機能は問題の性質に依存するため, 使用するかしないかをパラメタとして選択できるようにすることが必要と考えられる.

表 4: 多段多列同時消去の効果 (SR2201, 16PE, $DW = 40$)

問題サイズ: 11608

最適化	段数 BH	列数 BW	実行時間 [秒]	MFLOPS /PE	効率 [%]
あり	6	2	33.73	148.4	49.5
なし	5	2	34.50	145.1	48.4

表 5: 演算範囲限定の効果 ($DW = 40$)

SR8000, 2 ノード (16IP), 問題サイズ: 41296, $BH = 10, BW = 9$

	実行時間 [秒]	GFLOPS /PE	効率 [%]	速度向上比 (対 ILIB_LU)
ILIB_RLU	84.18	5.80	72.5	39.0
ILIB_LU	3286.19	7.14	89.3	1.0

SR8000, 2 ノード (16IP), 問題サイズ: 11608, $BH = 10, BW = 9$

	実行時間 [秒]	GFLOPS /PE	効率 [%]	速度向上比 (対 ILIB_LU)
ILIB_RLU	8.05	4.98	62.2	9.7
ILIB_LU	77.90	6.69	83.7	1.0

SR2201, 16PE, 問題サイズ: 11608, $BH = 6, BW = 2$

	実行時間 [秒]	MFLOPS /PE	効率 [%]	速度向上比 (対 ILIB_LU)
ILIB_RLU	33.73	148.4	49.5	9.5
ILIB_LU	319.18	204.2	68.1	1.0

4 ILIB_RLU における知識発見

この章では ILIB_RLU が行った自動チューニングにより定められたパラメタ情報から得られる知識と, その知識の活用法について議論する.

3.1章でも述べたように, 図 6, 図 7より $BH \times BW = k$(一定) となる双曲線上に効率的な LU 分解を行うパラメタ群が存在することが理解できる. 表 7に高い実行効率をえたパラメタを示す. これより SR8000 では $BH \times BW = 100$ 程度の値を超えない場合に良い性能を示し, 逆にこの値を超えると急激に性能が低下している. $BH \times BW$ の値は図 1にも示したように多段多列同時消去を行う際に参照される要素の個数であり, その個数だけ値が計算機内の物理的なレジスタに固定される. レジスタの数は有限であるので, その数を超えて値を割り当てようとすれば当然性能低下を引き起こすことが予想される. すなわち

表 6: 更新演算省略の効果 ($DW = 40$)

SR8000, 2 ノード (16IP), 問題サイズ: 41296, $BH = 10, BW = 9$

	実行時間 [秒]	速度向上比 (対・通常)
省略	83.81	1.0044
通常	84.18	1.0000

SR8000, 2 ノード (16IP), 問題サイズ: 11608, $BH = 10, BW = 9$

	実行時間 [秒]	速度向上比 (対・通常)
省略	8.06	0.9988
通常	8.05	1.0000

SR2201, 16PE, 問題サイズ: 11608, $BH = 6, BW = 2$

	実行時間 [秒]	速度向上比 (対・通常)
省略	32.64	1.0334
通常	33.73	1.0000

各計算機に依存してこのパラメタ (上限値) は決定されることが予想される. しかしそうして得られたパラメタの範囲に収まる BH と BW の組み合わせのうち最も高い性能を得るパラメタを特定するにはコンパイラのレジスタ割り当てなどの検討を行わねばならず, さらに細かい議論が必要になるため, 現在は BH と BW の組合せを全探索によって決定している. 以上より得られた知識をまとめると

1. ILIB_RLU(ブロックアルゴリズムを用いた LU 分解) において高い性能を得る $BH \times BW$ の値にはある上限値が存在する
2. 上限となる $BH \times BW$ の値は主に計算機に依存し, 問題に依存する部分は小さい
3. 具体的に最適な BH と BW それぞれの値を予測することは困難で, 最終的には発見的手法によりパラメタを決定しなければならない

ILIB_RLU によって人工心臓の構造解析を行った際は, 同一の係数行列を用いて数千回にわたり連立一次方程式を解いたため, 全探索を行う自動チューニングのオーバヘッドがそれほど問題にならなかった. しかし, 自動チューニングのコストを減らしたい場合は 1. の知識を用いてパラメタの候補を減らすことにより時間を短縮させることができる. さらに 2. の知識を用いて, 大きなサイズの問題を解く際にも先に小さなサイズの問題を解いて $BH \times BW$ の値の見当をつけてからチューニングを行うことにより時間の短縮が図れる. しかし 1.2. の知識よりパラメタ探索の時間を短縮しても, 3. で述べたように最終的には複数回の実行を繰り返してパラメタを決定することになる. そこで, ある連立一次方程式を一回だけ解く, すなわち LU 分解を一回だけ行う場合は, 使用する計算機に依存したパラメタをもつ準最適な LU 分解ルーチンを用意しておくといった方法が考えられる.

表 7: 実行時間上位のパラメタ (SR8000)

	BH	BW	時間 [秒]	$BH \times BW$
1	10	9	8.05	90
2	12	8	8.07	96
3	11	8	8.11	88
4	12	7	8.24	84
5	12	6	8.29	72
6	14	6	8.34	84
7	9	10	8.35	90
8	11	7	8.36	77
9	10	8	8.39	80
9	15	6	8.39	90

5 おわりに

本稿ではユーザによるパラメタ設定の手間を省き, かつ自動チューニング機能により高い性能を得ることができるスパースダイレクトソルバ ILIB_RLU の機能と性能について述べた. 自動チューニング機能を使用することにより, 疎行列の LU 分解において最高で理論性能に対する効率 72.5% および, 通常の LU 分解と比較して最大で 39.0 倍の実行性能の向上を達成し, その有用性が示された.

さらに自動チューニング機能によって得られたパラメタから, 使用する計算機に関する知識が得られ, それはチューニングにかかる時間の短縮に利用できるということについて述べた. 今後はその知識を活用し自動チューニングの手続きを一般化して, 性能を低下させずに汎用的なルーチンの開発を行う予定である.

今後の課題として, ブロック列サイクリック分散におけるデータの分散幅 DW の最適化, 他機種での性能評価や新たなチューニング機能の開発などが挙げられる.

なお I-LIB に関する情報は, http://www.hints.org/ から入手可能である.

参 考 文 献

[1] 片桐孝洋，黒田久泰，大澤清，金田康正： I-LIB：自動チューニング機能付き並列数値計算ライブラリとその性能評価. JSPP2000 論文集. pp. 27–34 (2000).

[2] 山本有作，猪貝光祥，直野健：分散メモリ型並列計算機向けスパース対称行列ソルバの開発と評価. 情報処理学会論文誌. Vol.41,No.5,pp.1567–1576 (2000).

[3] Bilmes, J., Asanović, K., Chin, C.-W. and Demmel, J.: Optimizing Matrix Multiply using PHiPAC: a Portable, High-Performance, ANSI C Coding Methodology, *Proceedings of International Conference on Supercomputing 97*, pp. 340–347 (1997).

[4] Whaley, R. C. and Dongarra, J. J.: Automatically Tuned Linear Algebra Software, ATLAS project, http://www.netlib.org/atlas/index.html.

[5] 片桐孝洋，金田康正：並列固有値ソルバーの実現とその性能， *IPSJ SIG Notes, 97-HPC-69*, pp. 49–54 (1997).

[6] T. Katagiri, H. Kuroda, Y. Kanada:A Methdology for Automatically Tuned Parallel Tridiagonalization on Distributed Memory Vector-Parallel Machines, *Proceedings of VECPAR 2000*, pp. 265–277 (2000).

[7] 黒田久泰，金田康正：自動チューニング機能付き並列疎行列連立一次方程式ソルバの性能， *IPSJ SIG Notes, 99-HPC-76*, pp. 13–18 (1999). p

[8] H. Kuroda, T. Katagiri, Y. Kanada:Performance of Automatically Tuned Parallel GMRES(m) Method on Distributed Memory Machines, *Proceedings of VECPAR 2000*, pp. 251–264 (2000).

[9] (株) 日立製作所：スーパーテクニカルサーバ HITACHI SR8000 LINPACK 性能のご紹介，スーパーコンピューティングニュース， Vol. 1, No. 2, pp. 72–79 (1999). 東京大学情報基盤センター (スーパーコンピューティング部門).

[10] Balay, S., Gropp, W., McInnes, L. C. and Smith, B.: PETSc 2.0 Users Manual, ANL-95/11 - Revision 2.0.24, Argonne National Laboratory (1999).

[11] 村田大二郎：並列計算を用いた有限要素法による人工心臓の構造と流体の連成解の効率化，修士論文，東京大学大学院新領域創成科学研究科 (2000).

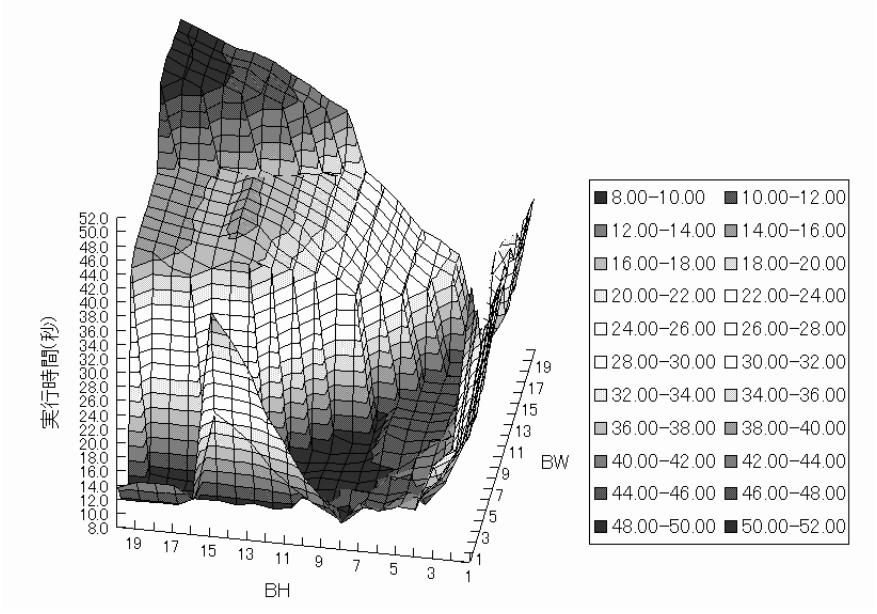


図 6: 最適な BH, BW の探索 (SR8000, 2 ノード (16IP), 次元数 $N = 11608$, $DW = 40$)

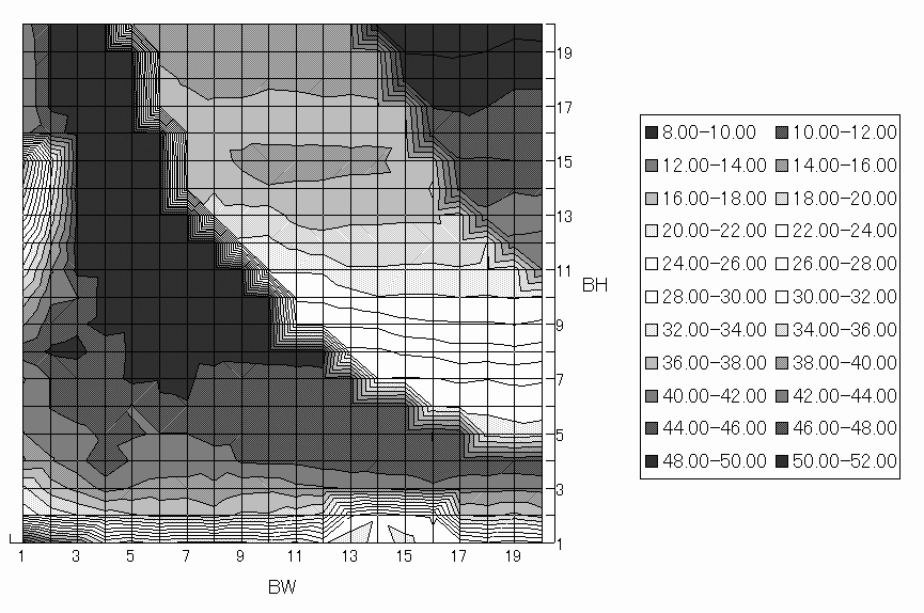


図 7: 最適な BH, BW の探索 (SR8000, 2 ノード (16IP), 次元数 $N = 11608$, $DW = 40$)