P1–3 行列計算ライブラリに対する計算環境に依存しないインタフェースの開発

Development of a computing environment independent interface to matrix computation libraries

梶山民人（JST）　額田彰（JST）　須田礼仁（東京大学）
長谷川秀彦（筑波大学）　西田晃（東京大学）

1 はじめに

行列計算ライブラリは固有の応用プログラムインタフェースを有し，利用者は使用するライブラリの定める関数名や引数，データ形式に従ってユーザプログラムを作成しなければならない．計算環境の特性を活かすには環境毎に最適化されたライブラリを用いる必要があり，計算環境が変わるとユーザプログラムを書き換えなければならない．そのため，計算環境に依存しない形でユーザプログラムを開発する手法に対する要求が高い．

これに対して我々は，行列計算処理をデータ授受と計算指示に分け，計算を文字列（数式）で指示することにより，ライブラリとユーザプログラムの間の独立性を高める枠組み SILC（Simple Interface for Library Collections）を提案している [1].

本稿では，SILC の設計と逐次計算環境向けシステムの実現，および命令記述言語について述べる．

2 SILC の設計と実現

SILC のユーザプログラムは，(1) 入力データの預け入れ，(2) 数式による計算の指示，および (3) 計算結果の受け取りの 3 つのステップでライブラリの機能を利用する．SILC はユーザプログラムから独立したメモリ空間を備え，入力データの預け入れと計算結果の受け取りのために SILC のメモリ空間とユーザプログラムの間でデータの授受を行なう．ライブラリ関数の呼び出しにともなうデータ形式の変換やメモリの確保は SILC 側で自動的に処理する．

クライアント・サーバ方式に基づく逐次計算環境向け SILC のシステム構成を図 1 に示す．ユーザプログラムは，PUT（入力データの預け入れ），SILC（計算指示），および GET（計算結果の受け取り）の 3 つのクライアントルーチンを通じて SILC サーバが管理するライブラリ群を利用する．SILC サーバはインタフェーススレッドと実行スレッドの 2 つのスレッドから成る．インタフェーススレッドはユーザプログラムと同期して 3 つのクライアントルーチンに対応する処理を行なう．文字列で与えられた計算指示は構文解析を経てリクエストキューに追加され，実行スレッドにより順次処理される．2 つのスレッドはリクエストキューを介して非同期に動作する．

SILC の提供するインタフェースによりライブラリ間の差異が隠蔽されるので，利用者は個々のライブラリに依存しない形でユーザプログラムを構築できる．ライブラリからの独立性により，逐次環境から並列環境に移ってもソースコードの変更なしにユーザプログラムを利用できる．ライブラリプログラムの並列化手法やアルゴリズムを変更した場合にもユーザプログラムの変更は不要である．

3 命令記述言語

SILC では行列計算に特化したプログラム言語独立の命令記述言語を備える．命令の基本単位は文であり，文には代入文と手続き呼び出し文がある．制御構文はなく，反復や条件分岐はユーザプログラムの記述言語により実現する．代入文の左辺には変数名を型宣言なしで指定する．変数の値を格納するメモリおよび値のデータ型はサーバにより管理される．データ型には密行列，疎行列，ベクトル，スカラーなどがある．代入文の右辺には式を記述する．式を構成する演算子には加減乗除，内積，共役転置，複素共役，連立一次方程式の求解，関数呼び出しなどがある．行列とベクトルには添字を付けることができる．例えば B = A[1:5,1:5] という文は行列 A の 5 行 5 列の部分行列を変数 B に代入することを表し，a[1:5] = b はベクトル $\boldsymbol{a}$ の最初の 5 つの要素をベクトル $\boldsymbol{b}$ の対応する要素で置き換えることを表す．

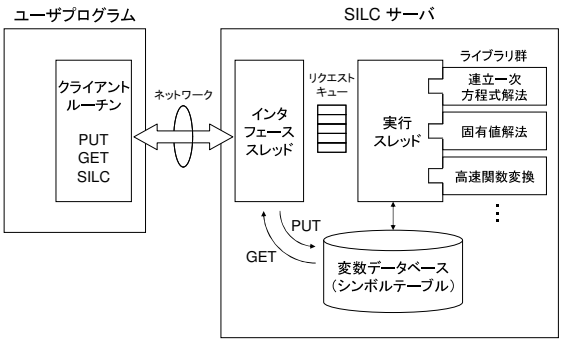


図 1: 逐次計算環境における SILC のシステム構成

```
// （入力）A: 係数行列，P: 補間行列，x:（初期）近似解，
//         b: 右辺
SILC("split(A, L, D, U)");     // 下三角，対角，上三角に分解
SILC("AA = P' * A * P");                  // 粗い格子の係数行列
SILC("x = (L + D) \ (b - U * x)");          // 前スムージング
SILC("y = AA \ (P' * (b - A * x))");      // 粗い格子の修正量
SILC("x += P * y");                          // 修正量を補間
SILC("x = (D + U) \ (b - L * x)");          // 後スムージング
// （出力）x ≈ A^-1 b : 解ベクトル
```

図 2: 2 グリッド法の命令記述例

連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ を 2 グリッド法で解くプログラムを図 2 に示す．入力データは計算指示の前に PUT ルーチンを用いてサーバに預ける．計算指示には SILC ルーチンを用い，引数として命令文を文字列で与える．図 2 の最初の文は手続き split の呼び出し，残り 5 つは代入文である．' は共役転置，\ は連立一次方程式の求解を表す演算子である．これらの手続き呼び出しと行列演算はサーバ側でライブラリ関数の呼び出しに翻訳されて処理される．

4 おわりに

ポスターでは，逐次計算環境向け SILC の設計と実現について示す．また，本システムの命令記述言語によりユーザプログラムを特定のライブラリに依存しない形で記述できることを示す．今後の課題は，既存の行列計算ライブラリの登録，並列計算環境向けシステムの設計と実現，および等価変換に基づく行列計算の最適化である．

謝辞　本研究は，科学技術振興機構（JST）戦略的創造研究推進事業（CREST）「大規模シミュレーション向け基盤ソフトウェアの開発」プロジェクトの一部として実施した．

参考文献

[1] 長谷川秀彦, 須田礼仁, 額田彰, 梶山民人, 中島研吾, 高橋大介, 小武守恒, 藤井昭宏, 西田晃. 計算環境に依存しない行列計算ライブラリインタフェース SILC. 情報処理学会研究報告 2004–HPC–100, pp. 37–42, 2004.

SSI
Scalable Software Infrastructure

ハイパフォーマンスコンピューティングと計算科学シンポジウム
(HPCS2005)
ポスター発表論文

P1－3
行列計算ライブラリに対する計算環境に依存しないインタフェースの開発

梶山民人（JST） 額田彰（JST） 須田礼仁（東京大学）
長谷川秀彦（筑波大学） 西田晃（東京大学）

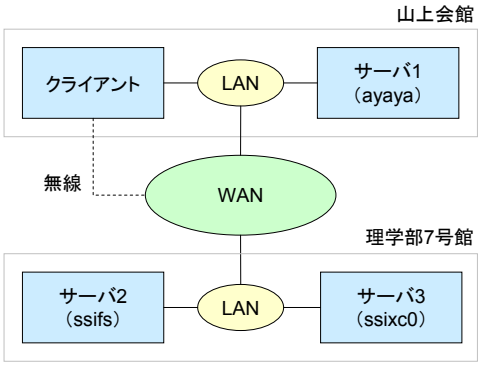
デモの内容

- 連立一次方程式 $Ax = b$ を解く

$$\begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & -1 & 2 & -1 & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{pmatrix}$$

- 入力
  - 係数行列 A
    - 三重対角行列、次数1,000
    - Compressed Row Storage (CRS) 形式で PUT
  - 右辺 $b = (-1, 0, \ldots, 0, -1)^T$
- 出力
  - 解ベクトル $x = (x_1, x_2, \ldots, x_N)^T$
- 解法
  - CG 法（OpenMP で並列化）
  - ガウスの消去法（密行列に自動変換）

- 計算環境
  - クライアント: ノートPC
    （MMX Pentium 266MHz、メモリ 96MB）
  - サーバ1（ayaya）: デスクトップPC
    （Pentium 4 1.8GHz、メモリ256MB）
  - サーバ2（ssifs）: Sun SunFire 3800
    （UltraSPARC-III 900MHz×4、OpenMP対応）
  - サーバ3（ssixc0）: IBM eServer xSeries 335
    （Dual Intel Xeon 2.8GHz、OpenMP対応）



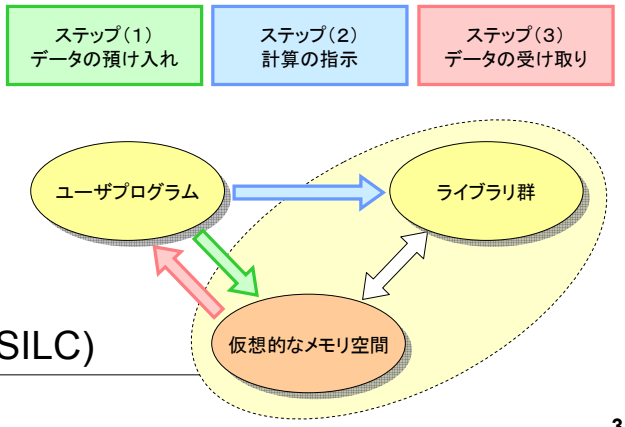
目標

計算環境に依存せずにユーザプログラムを構築できること

- 様々な行列計算ライブラリを共通のインタフェースで利用できる
- ライブラリを変更してもユーザプログラムを変更する必要がない（ライブラリの最適化は計算環境に依存）

基本アイデア

- データの授受と計算指示を分離する
- 計算を文字列（数式）で指示する
- 計算にはユーザプログラムのメモリ空間を利用しない



Simple Interface for Library Collections (SILC)

従来法

- データの授受と計算指示を同時に行なう
  - 関数名、引数、データ型が特定ライブラリに依存
- 計算用メモリはユーザプログラムが確保

SILC

- データの授受と計算指示を個別に行なう
- 計算を文字列（数式）で指示
  - ライブラリを特定せずにプログラムを作成可能
- 計算用メモリは SILC が自動的に確保

例：連立一次方程式 $Ax = b$ の求解

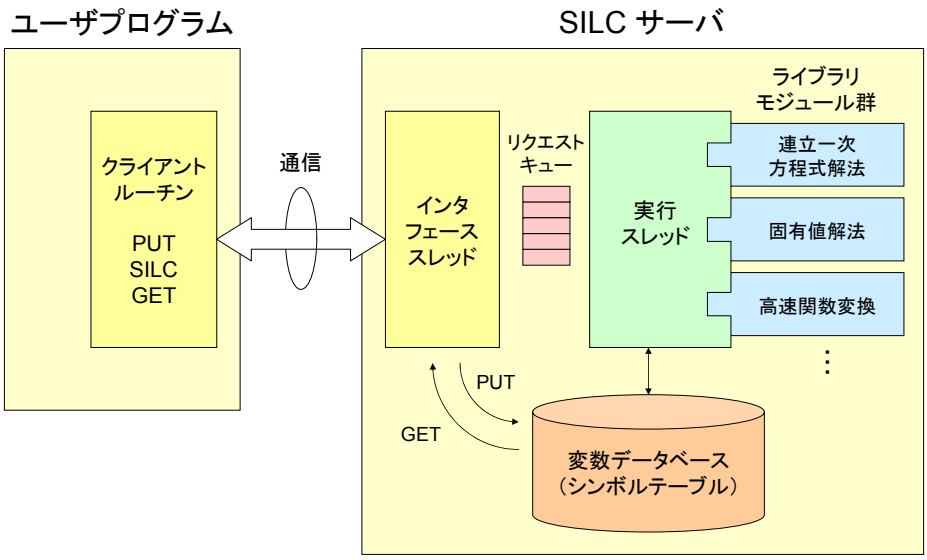
```
REAL*8 A(LDA, N), B(N)
INTEGER IP(N)
  :  データの生成
CALL LU (LDA, N, A, IP, STATUS)      関数呼出し
IF (STATUS.EQ.0) THEN
   CALL SOLVE (LDA, N, A, IP, B)     関数呼出し
END IF
  :  結果の利用
END
```

```
REAL*8 A(LDA, N), B(N), X(N)
  :  データの生成
CALL PUT (“A”, LDA, N, A, STATUS)
CALL PUT (“b”, N, B, STATUS)          (1)データの預け入れ
CALL SILC (“x = A \ b”)               (2)計算の指示
CALL GET (“x”, N, X, STATUS)          (3)データの受け取り
  :  結果の利用
END
```

逐次環境用 SILC の構成

- サーバ・クライアント方式
- 計算の指示と実行は非同期

– ユーザプログラムとインタフェーススレッドは同期してデータと計算指示を授受
– インタフェーススレッドと実行スレッドはリクエストキューを介して非同期に動作



モジュール関数

SILC からライブラリ関数を呼び出すためのラッパー（3種）

（a）組み込み演算子を実装するモジュール関数
– 数式中の演算子に応じて呼び出される

x = A \ b （連立一次方程式 $Ax = b$ の求解）

演算子とオペランドの組み合わせに基づき適切なモジュール関数を選択

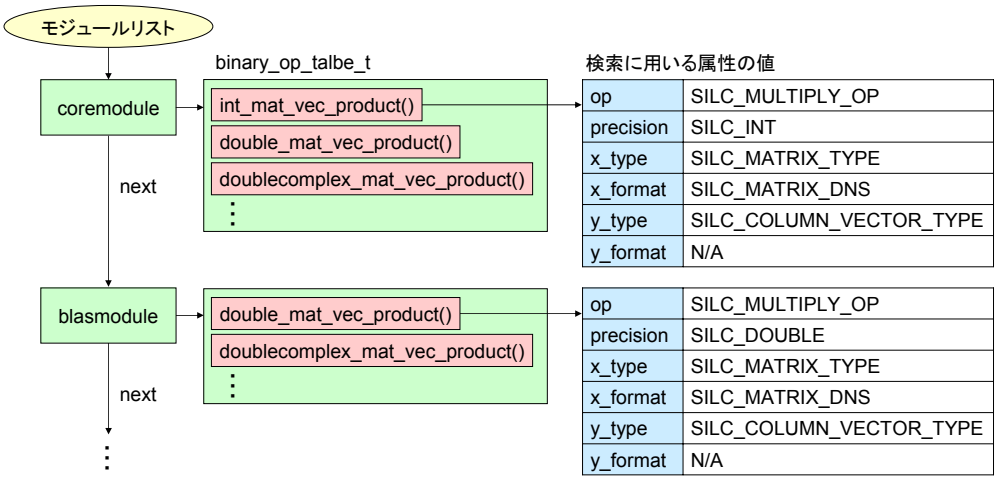
coremodule
LU_SOLVE(A, b)

（b）名前で呼び出すモジュール関数
– 数式中の関数・手続き呼び出しに対応するモジュール関数を名前で検索
– 関数の例
d = **diagvec**(A)
行列 A の対角要素から成るベクトルを返す
– 手続きの例
split(A, L, D, U)
A の下三角部分、対角部分、上三角部分をそれぞれ L, D, U に出力

（c）行列の格納形式を実装するモジュール関数
– 新しい格納形式に対応

組み込み演算子を実装するモジュール関数の検索

- モジュールリストの先頭から順に検索
  - 検索条件：演算子、データ型、精度、修飾子（添字、転置など）
- 検索条件にマッチする最初のモジュール関数を呼び出す
- prefer文：モジュールの検索順序の変更
  - SILC(“prefer blasmodule”)
    - blasmodule をモジュールリストの先頭に移動



op	SILC_MULTIPLY_OP
precision	SILC_INT
x_type	SILC_MATRIX_TYPE
x_format	SILC_MATRIX_DNS
y_type	SILC_COLUMN_VECTOR_TYPE
y_format	N/A

op	SILC_MULTIPLY_OP
precision	SILC_DOUBLE
x_type	SILC_MATRIX_TYPE
x_format	SILC_MATRIX_DNS
y_type	SILC_COLUMN_VECTOR_TYPE
y_format	N/A

関数・手続きのプロトタイプ

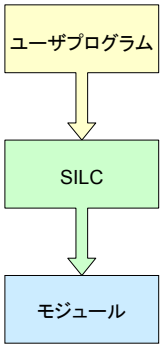
- モジュール関数の検索および引数・戻り値のチェックに利用

function_table_t

プロトタイプ	モジュール関数
vector **diagvec**(matrix) 戻り値の型、関数名、引数のデータ型	**func_diagvec**

procedure_table_t

プロトタイプ	モジュール関数
split(matrix in, matrix out, matrix out, matrix out) 手続き名、引数のデータ型、入出力タイプ	**proc_split**



関数・手続き
（命令記述言語レベルのインタフェース）
vector **diagvec**(matrix)
split(matrix in, matrix out, matrix out, matrix out)

モジュール関数
（実装言語レベルのインタフェース）
int **func_diagvec**(int n, silc_mfunc_param_t *param);
int **proc_split**(int n, silc_mfunc_param_t *param);

関数・手続きとモジュール関数の対応関係

モジュール関数の仕様

- 定義

```
int name (
    int n,
    silc_mfunc_param_t param[]);
```

- 引数
  - n：引数・オペランドの数
  - param[]：引数・オペランド、戻り値を授受するための構造体配列

```
typedef struct {
    silc_object_t *o;       // オブジェクト
    silc_modifier_t *m;     // 修飾子
} silc_mfunc_param_t;
```

- 戻り値
  - 正常終了なら 0、エラーなら −1