SILC: Simple Interface for Library Collections
（実装手法研究グループ）

担当：梶山民人（JST CREST 研究員）

概要

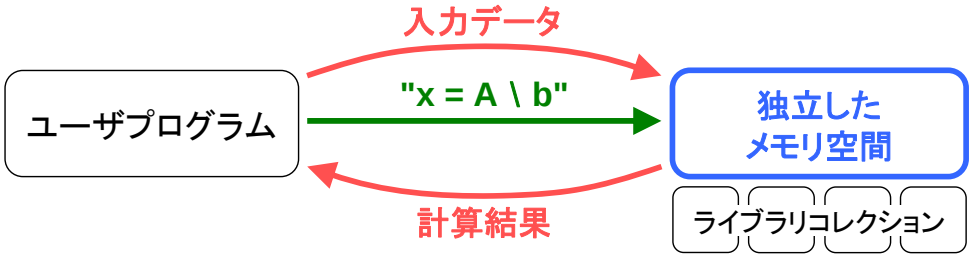
行列計算ライブラリを使いやすくするインタフェースSILC（Simple Interface for Library Collections）を提案し，共有メモリ並列環境向けシステムをクライアント・サーバ方式で実現した．本システムにより，ユーザプログラムを計算環境や行列計算ライブラリに依らない方法で作成することが可能になる．

従来法の問題点

ライブラリ固有のインタフェース（API）に基づいてユーザプログラムを作成すると，ユーザプログラムが特定のライブラリに依存する．そのため，別のライブラリを利用するにはユーザプログラムの大幅な修正が必要である．

SILC の基本アイデア（図1）

1. ライブラリ呼出しを**データ転送**と**計算の指示**に分離
2. 計算を**文字列（数式）**で指示
3. ユーザプログラムから**独立したメモリ空間**で計算を実行



SILC の利点

- ユーザプログラムを特定の計算環境や行列計算ライブラリに依存しない方法で作成できる
  - ユーザプログラムを修正せずに別の計算環境を利用できる
  - さまざまな行列計算ライブラリを同じ枠組みで利用できる
  - 解法や行列の格納形式を容易に変更できる
- 最小限の入力データと記憶領域を用意するだけでよい
  - 計算途中で必要な記憶領域は自動的に確保・解放される
- さまざまなプログラミング言語で利用できる
  - C言語，Fortran，Python（オブジェクト指向スクリプト言語）

クライアント・サーバ方式による実現（図4）

- ネットワークを介してサーバに接続し，PUT/EXEC/GET リクエストを通じてサーバの管理する行列計算ライブラリの機能を利用
- クライアントは逐次のユーザプログラム
- サーバは通信担当のインタフェーススレッドと計算担当の実行スレッドから成る
  - 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動作し，PUT/GET リクエストを処理
  - 実行スレッドはリクエストキューを介して EXEC リクエストを受け取り，非同期に計算を実行
- OpenMP によるサーバの並列化（cc-NUMA 対応）

図1　SILC の基本アイデア

◆ 従来のユーザプログラム（図2）

ライブラリ固有のデータ構造に従って行列やベクトルなどの入力データを用意する. 次に, 所定の関数名と引数の順序で関数呼出しを行なう. この方法は直感的で手軽な反面, ユーザプログラムが特定のライブラリに依存する.

```
double *A, *b;
int kl, ku, lda, ldb, nrhs, info, *ipiv;

/* 行列 A およびベクトル b の作成 */

dgbtrf (N, N, kl, ku, A, lda, ipiv, &info);  /* LU 分解 */
if (info == 0)
    dgbtrs ('N', N, kl, ku, nrhs, A, lda, ipiv, b, ldb, &info);  /* 求解 */
```

図2　従来のユーザプログラム
（LAPACK による連立一次方程式 ***Ax=b*** の求解例）

◆ SILC のユーザプログラム（図3）

SILC のサポートするデータ構造で入力データを用意する. 次に, データに名前を付けて SILC_PUT ルーチンによりデータを別メモリ空間に預ける. 必要な入力データを預けたあと, SILC_EXEC により計算を文字列（数式）で指示する. 計算終了後, SILC_GET で結果を受取る.

```
silc_envelope_t A, b, x;

/* 行列 A およびベクトル b の作成 */

SILC_PUT ("A", &A);
SILC_PUT ("b", &b);
SILC_EXEC ("x = A \ b");  /* 適切な求解ルーチンが呼び出される */
SILC_GET (&x, "x");
```

図3　SILC のユーザプログラム
（図2のユーザプログラムと同じ計算を実現する例）

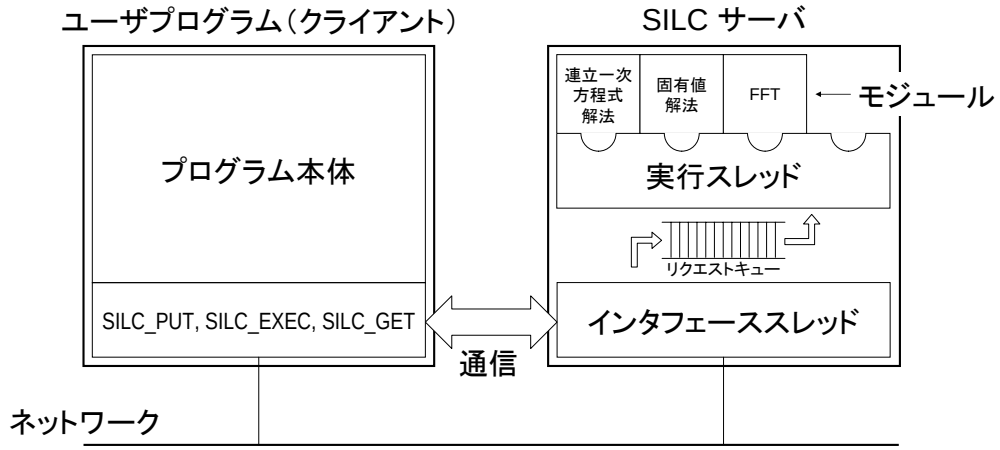


図4　SILC のシステム構成

行列計算ライブラリインタフェースの機能

◆ データ構造
- データ型：スカラー, ベクトル, 行列, 3次元配列
- 精度：整数, 実数, 複素数（単精度および倍精度）
- 行列の格納形式：密, 帯, CRS（行圧縮）形式

◆ 命令記述言語
- 文（statement）
  - 右辺が式である代入文（単純代入文および累算代入文）
  - 引数が式である手続き呼出し文
- 式（expression）の構成要素
  - 変数名, 定数（i, e, pi など）
  - リテラル（すべてのデータ型のリテラルを記述可能）
  - 四則演算（+, −, *, /, %）
  - 連立一次方程式の求解（A \ b）
  - 要素ごと乗算（*@）および要素ごと除算（/@）
  - 共役転置（A‘）, 複素共役（A~）, 転置（A’~）
  - 関数（sqrt, diagvec, zeros など）
- 添字（subscript）による部分参照および制約代入
  - 例："A[1:5,1:5]" は行列Aの部分行列（5行5列）を表す

SILC の利用例

◆ さまざまな計算環境の利用

表1の4つの計算環境で SILC サーバを実行して連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ を CG 法で解いた. ユーザプログラム（図3）は表1（a）のノート PC で実行した. 行列 A には三重対角行列を CRS 形式で与えた. 次数を N とすると通信量は $O(N)$, 計算量は $O(N^2)$ であるため, N が十分に大きければ通信コストを加味しても遠隔の高速な計算機を利用した方が速い. 図5の実験結果はこの予想を支持している.

表1　計算環境（100 Mbps の Ethernet で接続）

	計算環境	仕様	OpenMP
(a)	ノート PC	Intel Pentium M 733 1.1 GHz, メモリ 768 MB, Fedora Core 3	N/A
(b)	SGI Altix3700	Intel Itanium2 1.3 GHz × 32基, メモリ 32 GB, Red Hat Linux Advanced Server 2.1	1スレッド
(c)	IBM eServer OpenPower 710	IBM Power5 1.65 GHz × 2基 (4 論理 CPU), メモリ 1GB, SuSE Linux Enterprise Server 9	4スレッド
(d)	SGI Altix3700	(b) と同じ	16スレッド

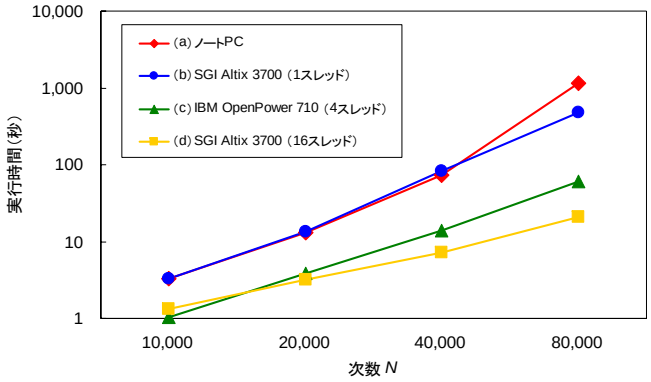


図5　実験結果

◆ 複数の解法の利用

行列計算ライブラリはモジュールとしてサーバに組み込まれる. prefer 文を用いてモジュールの優先順位を変えることにより, 図6に示すようにひとつの演算子に対応する複数のライブラリ関数（解法）を容易に切り替えて利用できる.

まとめ

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 を提案し, 共有メモリ並列環境向けシステムをクライアント・サーバ方式で実現して提案手法の実現可能性を示した. また, 多様な計算環境, 解法, 行列の格納形式を柔軟かつ容易に利用できること, 高性能並列計算機の遠隔利用により本手法の主要な利用コストである通信時間を十分に補える速度向上が得られることを示した.

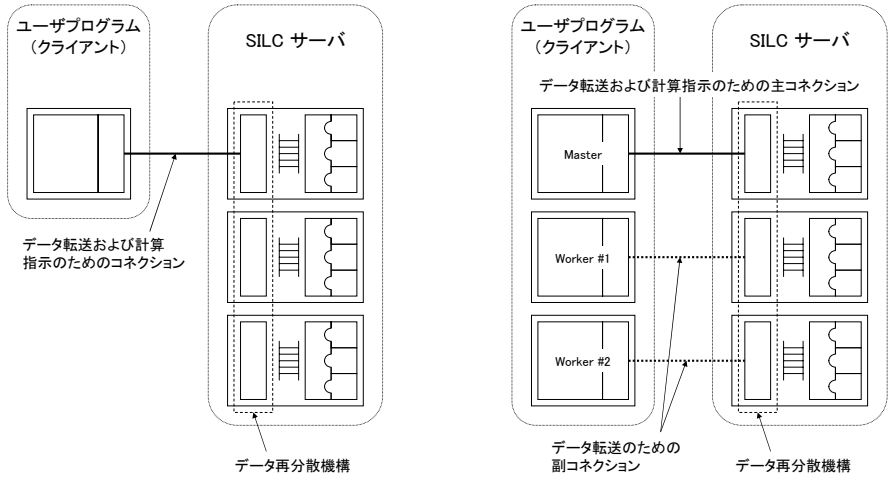
今後の課題

◆ 共有メモリ計算環境向け SILC

- 主要な行列計算ライブラリに対応するモジュール（ラッパー）
- 命令記述言語の単独利用可能なスクリプト言語への拡張
- 数式に基づく実行時最適化

◆ 分散版 SILC の実現（図8, 図9）

- プロセス数の異なるクライアントとサーバの接続方法
- 行列の分散方式を変えるための再分散機構
- サーバプロセスを部分集合に分けて並列動作させる仕組み
- SILC のインタフェース機能を内蔵する MPI 並列ライブラリ



```
SILC_EXEC ("prefer leq_lu");
SILC_EXEC ("x1 = A \ b");  /* LU 分解法で求解 */
SILC_EXEC ("prefer leq_cg");
SILC_EXEC ("x2 = A \ b");  /* CG 法で求解 */
SILC_EXEC ("d = b − A * x1; norm1 = sqrt(d' * d)");  /* ||b − Ax1|| */
SILC_EXEC ("d = b − A * x2; norm2 = sqrt(d' * d)");  /* ||b − Ax2|| */
```

図6　連立一次方程式の求解精度を比較するプログラム

◆ SILC の命令記述言語による CG 法の実現

SILC の命令記述言語は行列計算を計算環境や行列の格納形式に依らない方法で記述できる. SILC の命令記述言語で実現した CG 法を図7に示す. 計算環境や格納形式が変わってもプログラム中の数式部分の修正は不要である.

```
int silc_cg(int n, int nnz, double *value, int *index,
    int *row, double *b, double *x)
{
    silc_envelope_t object;
    int i;
    double nrm2;

    /* SILC サーバに接続する */
    SILC_INIT();

    /* 行列 A を PUT する */
    object.type = SILC_MATRIX_TYPE;
    object.format = SILC_FORMAT_CRS;
    object.precision = SILC_DOUBLE;
    object.m = object.n = n;
    object.nnz = nnz;
    object.v = value;
    object.row = row;
    object.index = index;
    SILC_PUT("A", &object);

    /* ベクトル b を PUT する */
    object.type = SILC_COLUMN_VECTOR_TYPE;
    object.precision = SILC_DOUBLE;
    object.length = n;
    object.v = b;
    SILC_PUT("b", &object);

    /* 連立一次方程式 Ax = b を CG 法で解く */
    SILC_EXEC("rho_old = 1.0");
    SILC_EXEC("n = length(b)");
    SILC_EXEC("p = zeros(n, 1)");
    SILC_EXEC("x = zeros(n, 1)");
    SILC_EXEC("r = b");
    SILC_EXEC("bnrm2 = 1.0 / norm2(b)");
    for (i = 0; i < n; i++) {
        SILC_EXEC("rho = r' * r");
        SILC_EXEC("beta = rho / rho_old");
        SILC_EXEC("p = r + beta * p");
        SILC_EXEC("q = A * p");
        SILC_EXEC("alpha = rho / (p' * q)");
        SILC_EXEC("r = r - alpha * q");
        SILC_EXEC("nrm2 = norm2(r) * bnrm2");
        SILC_EXEC("x = x + alpha * p");

        /* 収束をチェックする */
        object.v = &nrm2;
        SILC_GET(&object, "nrm2");
        printf("iter: %d, res: %e¥n", i+1, nrm2);
        if (nrm2 <= EPSILON)
            break;
        SILC_EXEC("rho_old = rho");
    }

    /* ベクトル x を GET する */
    object.v = x;
    SILC_GET(&object, "x");

    /* 接続を切る */
    SILC_END();

    return i; /* 反復回数を返す */
}
```

図7　SILC の命令記述言語で実現した CG 法

図8　逐次クライアント＋MPI 並列サーバ

図9　MPI 並列クライアント＋MPI 並列サーバ

ソフトウェアの公開

- ◆ SILC のソースコード, ドキュメント, サンプルプログラムをフリーソフトウェアとして一般公開している
- ◆ ホームページ：**http://ssi.is.s.u-tokyo.ac.jp/silc/**
- ◆ 動作環境：Unix, GNU/Linux, Mac OS X
- ◆ 公開履歴
  - □ SILC バージョン1.0（2005年9月20日）
  - □ SILC バージョン1.1（2005年11月25日）

研究成果

[1] T. Kajiyama, A. Nukada, H. Hasegawa, R. Suda, A. Nishida. LAPACK in SILC: Use of a Flexible Application Framework for Matrix Computation Libraries. In Proc. 8th International Conference on High Performance Computing in Asia Pacific Region (HPC Asia 2005), 2005.

[2] T. Kajiyama, A. Nukada, H. Hasegawa, R. Suda, A. Nishida. SILC: a Flexible and Environment Independent Interface to Matrix Computation Libraries. In Proc. 6th International Conference on Parallel Processing and Applied Mathematics (PPAM 2005), LNCS, to appear.

[3] 梶山, 額田, 須田, 長谷川, 西田. 共有メモリ並列環境における SILC の実現と利用, 第34回数値解析シンポジウム講演予稿集, pp.49-52, 2005.

[4] 梶山, 額田, 須田, 長谷川, 西田. SILC: 行列計算ライブラリの利用を簡単化するフレームワーク, 第10回日本計算工学会講演会論文集, pp.239-242, 2005.

[5] 梶山, 額田, 須田, 長谷川, 西田. 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, 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, 2005.