2006年並列／分散／協調処理に関する『高知』サマー・ワークショップ
（SWoPP 高知 2006）

行列計算ライブラリインタフェース SILC の分散並列環境への実装

梶山民人（JST） 額田彰（JST） 須田礼仁（東大）
長谷川秀彦（筑波大） 西田晃（中央大）

発表の概要

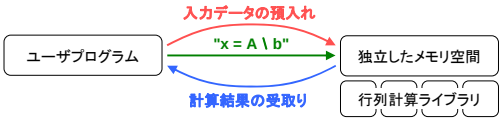
- 研究の背景
  - 行列計算ライブラリの呼出し法
- 分散型 SILC
  - SILC: 行列計算ライブラリを計算環境や言語に依らない方法で利用するためのインタフェース
- 実験
- まとめと今後の課題

研究の背景

- 多種多様な行列計算ライブラリ
  - 互換性のないインタフェース（API）
  - 種々の計算環境（共有メモリ並列, 分散並列）
    - 特定環境でのみ利用できる専用ライブラリ
- 別のライブラリや計算環境を利用するために ユーザプログラムの修正が必要
  - 解法、行列の格納形式、演算精度の変更
  - 逐次プログラムの並列環境への移植

行列計算ライブラリインタフェースSILC

- **S**imple **I**nterface for **L**ibrary **C**ollections
  - ライブラリ、計算環境、プログラミング言語に依存しない インタフェース
- 次の3ステップで行列計算ライブラリを利用
  1. 入力データの預入れ
  2. 文字列（数式）による計算の指示
  3. 計算結果の受取り



例：連立一次方程式 $Ax = b$ の求解

- ScaLAPACK を利用する従来法のプログラム

```
double *A, *B;
int desc_A[9], desc_B[9], *ipiv, info;
/* 行列 A とベクトル b の作成 */
pdgesv(N, NRHS, A, IA, JA, desc_A, ipiv, B, IB,
  JB, desc_B, &info);
```

- ScaLAPACK を利用する SILC のユーザプログラム

```
silc_envelope_t A, b, x;
/* 行列 A とベクトル b の作成 */
SILC_PUT("A", &A);
SILC_PUT("b", &b);
SILC_EXEC("x = A \\ b"); /* pdgesv() の呼出し */
SILC_GET(&x, "x");
```


SILC の特徴

- 用いるライブラリに依存しない
  - 異なるライブラリの解法、前処理、行列の格納形式、演算精度を同じインタフェースで利用できる
- 計算環境に依存しない
  - 逐次、共有メモリ並列、分散並列の計算環境をサポート
  - ユーザプログラムの修正なしに他の環境を利用できる
- プログラミング言語に依存しない
  - どの言語でも同じ数式でライブラリを呼出せる
  - ユーザプログラム: C, Fortran, Java, Python で作成可

関連研究

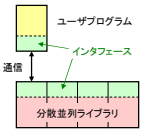
- Trilinos
  - 行列計算ライブラリを C++ クラスライブラリとして統合する枠組み
  - (a)データ構造、(b)抽象クラスの継承、(c)共通のディレクトリ構造
  - API の詳細はライブラリ毎に異なる
- Amesos
  - 種々の直接解法ライブラリに対する共通インタフェース
  - インタフェース統合の対象を連立一次方程式の直接解法に限定
- Ninf-G
  - グリッド環境下で遠隔手続き呼出し(RPC)を実現するミドルウェア
  - RPC を実行するプロセスは1つ(逐次プログラム、または分散並列プログラムの1プロセス)
  - ライブラリ側のプロセス間のデータ分散処理をユーザが記述

分散型 SILC

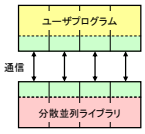
- MPI ベースの行列計算ライブラリに対するインタフェースを提供
- 3種のシステム構成
  - (A)逐次クライアント ＋ 分散並列サーバ
  - (B)分散並列クライアント ＋ 分散並列サーバ
    - クライアント・サーバ方式に基づくシステム構成
  - (C)SILC のインタフェースをライブラリと共にリンク
    - クライアント・サーバ方式を採用できない環境向け

分散型 SILC のシステム構成

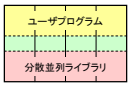
構成(A)
逐次クライアント＋分散並列サーバ



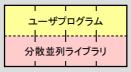
構成(B)
分散並列クライアント＋分散並列サーバ



構成(C)
SILC のインタフェースと共にライブラリをリンク

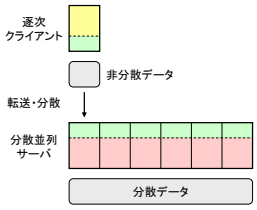


従来法
ユーザプログラムとライブラリが直結

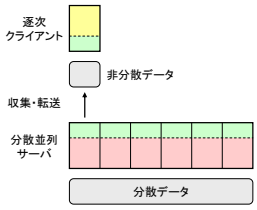


データ転送:逐次の場合

- SILC_PUT
  - 非分散データを転送
  - サーバ側でデータ分散

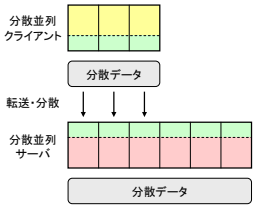


- SILC_GET
  - サーバ側でデータ収集
  - 非分散データを転送

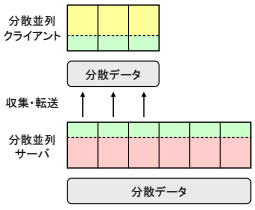


データ転送:分散並列の場合

- SILC_PUT
  - データを並列に転送
  - サーバ側で再分散



- SILC_GET
  - サーバ側で再分散
  - データを並列に転送



データ分散方式

- 種々のデータ分散方式をサポート
  - 2次元ブロックサイクリック分割の密行列
  - 行ブロック分割の密行列、CRS 形式の疎行列
  - 列ブロック分割の帯行列
- 格納形式モジュール
  - データ分散方式(行列の格納形式)に依存する処理を個別のモジュールとして実装
  - サーバの起動時にモジュールを動的にリンク

数式による計算指示

- 数式
  - 四則演算、連立一次方程式の求解、転置(A')、添字(A[1:k,1:k])などで構成
- 演算モジュール
  - 個々のライブラリ関数にラッパーを付与
  - ライブラリ別に演算モジュールとしてまとめる
    - モジュール毎に演算子とラッパーの対応関係を記述
  - サーバ起動時に動的にリンク
  - ScaLAPACK、反復解法ライブラリ Lis などをサポート

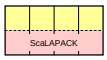
実験

- 従来法と SILC のユーザプログラムの比較
  - 両者で同じ例題を解いて SILC の性能を調べる
- 例題
  1. 連立一次方程式 $Ax = b$ の求解
     - 分散並列ユーザプログラム
  2. 偏微分方程式の初期値問題の求解
     - 逐次ユーザプログラム

1. 連立一次方程式 $Ax = b$ の求解

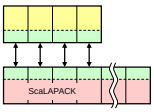
- 従来法

```
pdgesv(N, NRHS, A, IA, JA,
  desc_A, ipiv, B, IB, JB,
  desc_B, &info);
```



- SILC

```
SILC_PUT("A", &A);
SILC_PUT("b", &b);
SILC_EXEC("x = A \\ b");
SILC_GET(&x, "x");
```



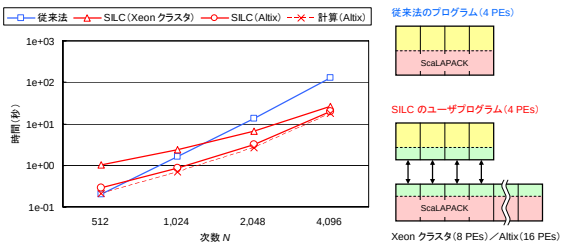
  - 従来法、SILC 共に ScaLAPACK の密行列ソルバを利用
  - 行列 A は2次元ブロックサイクリック分割の密行列

テスト環境

- ユーザプログラム(従来法、SILC)
  - IBM OpenPower 710(Power5 1.65 GHz × 4)
- SILC サーバ
  - Xeon クラスタ(Intel Xeon 2.8 GHz × 8ノード)
  - SGI Altix 3700(Intel Itanium2 1.3 GHz × 16)
- Gigabit Ethernet(1 Gbps)
- 倍精度実数でデータ入出力および計算

連立一次方程式 $Ax = b$ の求解(結果)

- 従来法:ScaLAPACK の求解ルーチン pdgesv の実行時間
- SILC:サーバ接続から SILC_GET までの経過時間
- 速度向上(N = 4,096): 4.88(Xeon クラスタ), 6.46(Altix)



2. 偏微分方程式の初期値問題

- 1次元拡散方程式を以下の条件で解く

$$\frac{\partial \varphi}{\partial t} = \frac{\partial^2 \varphi}{\partial x^2} \quad (t \geq 0, 0 \leq x \leq \pi)$$

初期条件: $\varphi = \sin x \ (t = 0, 0 \leq x \leq \pi)$

境界条件: $\varphi = 0 \ (t > 0, x = 0), \varphi = 0 \ (t > 0, x = \pi)$

- クランク・ニコルソン法
  - 時間刻み毎に連立一次方程式を解く

クランク・ニコルソン法

- 初期条件と境界条件から行列 A と時刻 t_0 の解 $\boldsymbol{x}_0$ を作成
  - 行列 A は $N \times N$ の三重対角行列
- 時刻 $t_k = t_{k-1} + \Delta t$ $(k = 1, 2, 3, \ldots)$ について、
  - 第 $k-1$ 時刻の解 $\boldsymbol{x}_{k-1}$ から $\boldsymbol{b}$ を作成
    - 別の三重対角行列 C を設けて $\boldsymbol{b} = C\boldsymbol{x}_{k-1}$ を求める
  - 連立一次方程式 $A\boldsymbol{x}_k = \boldsymbol{b}$ を解く
    - 従来法は LAPACK の帯行列ソルバ、SILC サーバは ScaLAPACK の帯行列ソルバを利用

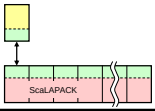
ユーザプログラムの構成

- 従来法

行列 A, C とベクトル x_0 を作成
時刻 t_k $(k = 1, 2, 3, \ldots)$ について {
  $\boldsymbol{b} = C\boldsymbol{x}_{k-1}$ を計算
  連立一次方程式 $A\boldsymbol{x}_k = \boldsymbol{b}$ を解く
}

- SILC

```
行列 A, C とベクトル x0 を作成
SILC_PUT("A", &A);
SILC_PUT("C", &C);
SILC_PUT("x", &x); /* x0 */
時刻 tk (k = 1, 2, 3, ...) について {
  SILC_EXEC("b = C * x");
  SILC_EXEC("x = A \\ b");
}
SILC_GET(&x, "x"); /* xk */
```

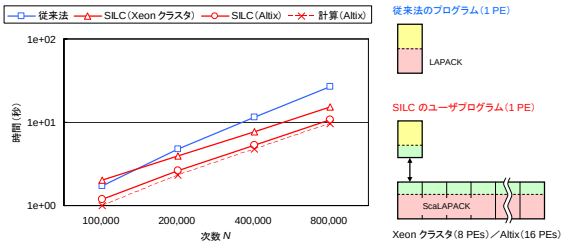


テスト環境

- ユーザプログラム（従来法、SILC）
  - IBM ThinkPad T42（Intel Pentium M 1.7 GHz）
- SILC サーバ
  - Xeon クラスタ（Intel Xeon 2.8 GHz × 8ノード）
  - SGI Altix 3700（Intel Itanium2 1.3 GHz × 16）
- Gigabit Ethernet（1 Gbps）
- 倍精度実数でデータ入出力および計算

偏微分方程式の初期値問題（結果）

- 時間刻み20ステップ分の実行時間
- 速度向上（N = 80,000）: 1.76（Xeon クラスタ）, 2.50（Altix）



考察

- 計算の高並列化により通信コストを上回る速度向上

	計算量	データ量	速度向上	
			Xeon クラスタ（8 PEs）	Altix（16 PEs）
例題1（密行列ソルバ）	$O(N^3)$	$O(N^2)$	4.88（N = 4,096）	6.46（N = 4,096）
例題2（帯行列ソルバ）	$O(kN)$	$O(N+k)$	1.76（k = 20, N = 80,000）	2.50（k = 20, N = 80,000）

- 別のライブラリ・計算環境の利用が容易
  - 逐次プログラムからも MPI ベースのライブラリを利用可
  - サーバの変更にはユーザプログラムの修正は不要

まとめ

- 分散型 SILC
  - MPI ベースの行列計算ライブラリを計算環境やプログラミング言語に依存しない方法で利用できる
  - 別のライブラリ・計算環境をユーザプログラムの修正なしに利用可
  - 性能比較実験により速度向上の可能性を確認
- 今後の課題
  - システム構成（C）の実装
  - 通信コストの分析と速度向上率の予測モデルの提案
  - 既存の主なライブラリに対する演算・格納形式モジュール

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。