Vol.2013-HPC-139 No.14
2013/5/30

並列固有値計算ライブラリの統一的インターフェースの開発とベンチマーク

坂下 達哉[1,a)]　五十嵐 亮[1,b)]　大久保 毅[1,c)]　藤堂 眞治[1,d)]

概要：ScaLAPACK の次世代の固有値計算ライブラリとして，EigenK, Elemental 等のライブラリが開発されている．これらのライブラリの仕様はばらばらであり，利用者の立場から非常にわかりづらく十分に性能が調査されていないのが現状である．そこで，我々はこれらのライブラリのインストール方法，分散行列を統一的に扱うためのインターフェース，ベンチマーク採取方法を整備している．発表では，これらのツールを用いて得られた Intel クラスタ/FX10 等でのベンチマーク結果を紹介する．

キーワード：固有値解法，インターフェース，ScaLAPACK，EigenK，Elemental

1. はじめに

固有値計算は物性物理分野において不可欠である．固有値計算ルーチンは各種の数値計算ライブラリに含まれているが，これらのライブラリの使用方法は十分に周知されているとはいえず，十分な性能の調査もなされていない．その原因としてまず考えられるのは，ライブラリの仕様が統一されていない事である．また，MPI の実装，固有値計算ライブラリの下部で用いられる数値計算ライブラリの依存関係，C++/C から Fortran 間の呼び出し方法の違いによりコンパイル・リンク方法が複雑になり，ユーザはそもそもインストールの段階で躓くことがあることもあげられる．このような問題が，計算物質科学の研究者が計算機シミュレーションを遂行するうえで障害になっている．

そこで，我々は固有値計算ライブラリに対して C++による統一的なインターフェースの開発を行っている．これには固有値計算ライブラリのインストールおよびリンク，ベンチマーク測定の作業を容易にするためのスクリプトも含める．これは各ユーザが手間を掛けて独立に行ってきた作業の統一的な方法を提供する一つの試みである．作成したインターフェース，スクリプトは，Boost ライセンスで公開している [1].

[1] 東京大学物性研究所
千葉県柏市柏の葉 5 丁目 1 番 5 号
[a)] t-sakashita@issp.u-tokyo.ac.jp
[b)] rigarash@issp.u-tokyo.ac.jp
[c)] t-okubo@issp.u-tokyo.ac.jp
[d)] wistaria@issp.u-tokyo.ac.jp

2. 密行列の並列固有値計算ライブラリの概観

インターフェース開発の対象とする密行列用の固有値計算ライブラリは，現在も開発が継続されている ScaLAPACK[2]，EigenK[3]，Elemental[4] である．これらのライブラリの概要について以下にまとめる．本稿では，実対称の密行列に対する固有値計算ルーチンを扱う．

2.1 ScaLAPACK

ScaLAPACK は，線型計算ライブラリ LAPACK の MPI 並列版である．ScaLAPACK は，コンパイラごとにチューニングが行われた実装がある．代表的な実装は Intel 製の MKL(Math Kernel Library)，富士通製の SSLII である．

ScaLAPACK には，以下の固有値計算ルーチンがある．

- pdsyev 関数（QR 法）
- pdsyevd 関数（分割統治法）
- pdsyevr 関数（MR3 法）

ScaLAPACK の分散行列は，Fortran の 2 次元配列で実現されている．これは C++/C では 1 次元配列に相当するため，分散行列はリーディング次元を考慮してアクセスする必要がある．

ScaLAPACK では，プロセス間の通信に独自の BLACS(Basic Linear Algebra Communication Subprograms) というライブラリが使用されている．BLACS で用いられる 2 次元グリッドのプロセス配置順は, column-major か row-major かを選択できる．

2.2 EigenK

EigenK は，京コンピュータをはじめとする次世代スーパーコンピュータ向けに開発されたライブラリである．これは，ScaLAPACK, BLACS，LAPACK，BLAS に依存しており，Fortran で書かれている．EigenK には，eigen_s（実対称行列用，3重対角型を経由する），eigen_sx（実対称行列用，5重対角型を経由する），eigen_h（複素エルミート行列用）というルーチン群が含まれている．3重対角行列，5重対角行列の固有値・固有ベクトルは分割統治法で求めている．

2.3 Elemental

Elemental は C++により実装された線型計算ライブラリで，LAPACK/BLAS の上に構築されている．固有値計算には，MR3 法が採用されている．計算に用いられる分散行列型は C++/C の 1 次元配列の上で独自に C++で実装されている．

2.4 ブロックサイクリック行列

密行列の並列固有値計算ライブラリで採用されている行列の分散方法は，いずれも2次元ブロックサイクリック法である．ScaLAPACK は任意のブロックサイズに対応しているが，Elemental および EigenK で用いられる分散行列のブロックサイズは 1×1 である．このように密行列の並列固有値計算ライブラリはいずれもブロックサイクリック行列を用いているが，それぞれの仕様には違いがある．その違いを吸収するためのインターフェースを作成し，ユーザの便宜を図るのが本研究の目的である．

3. インターフェースの構成

インターフェースの構成は，Elemental のものを参考に以下のように策定した．

(1) solver クラス

文字列でライブラリ名を与える仕様になっている．これはコマンドライン引数，入力ファイル等で実行時に固有値計算ライブラリを動的に選択できるようにするためである．この仕様は，C++の Factory を用いて実現している．Factory には，EigenK のようにパディングを行った行列を扱えるように，行列サイズを決定する関数を用意し，ユーザから見えないところで呼び出されるようにしている．

(2) initialize 関数， finalize 関数

MPI_init と MPI_Finalize は，ユーザプログラムで行う．これらとは別に固有値計算ライブラリごとの初期化関数, 終了関数がある場合は, initialize 関数，finalize 関数でラップする．

(3) grid クラス

2次元グリッドに関する情報を持つクラスで，以下の特徴がある

- グリッドサイズ，自ランクのグリッド座標を計算し保持する．
- グリッドサイズはできるだけ正方になるように決定する．
- ライブラリごとにグリッド，MPI のコミュニケータの作成は行わない．
- グリッドのプロセス配置順が column-major か row-major であるかは，テンプレートパラメータで指定する．

(4) distributed_matrix クラス

C++/C の 1 次元配列からなる分散行列を持つ．行列が row-major か column-major かはテンプレートパラメータで与える．distributed_matrix には，以下のメンバ関数を用意している．

- grid クラスの情報，グローバル行列のサイズよりブロックサイズの計算を行う関数．ただし，Elemental/EigenK の場合には，ブロックサイズは 1×1 となる．
- グローバル行列と分散行列の添字の変換関数
- 分散行列の読み書きを行う関数
- グローバル行列の添字が与えられたときに，自プロセスが分散行列において対応する行列成分を持っているかを判定する関数
- 各プロセスごとに分散行列の表示を行う関数

(5) diagonalize 関数

- 使用する固有値計算ライブラリの固有値計算ルーチンを呼び出す．
- grid クラス，distributed_matrix クラスの情報を用いて，固有値計算ライブラリで用意されているグリッド型の生成，分散行列型の確保もこの中で行う．

(6) gather/scatter 関数

- 分散行列をルートプロセス（ユーザが決める）に gather/scatter を行う．
- どの固有値計算ライブラリからでも使用できるように，通信には BLACS 等は使わずに独自に MPI で実装した．

以上のインターフェースの構成について，補足する．

- 後に複数の固有値計算ライブラリを組み合わせて使用する場合に備えて，gird クラスおよび distributed_matrix クラスは用いる固有値計算ライブラリによらないようにした．
- 固有値を収めるベクトル，gather/scatter に用いるグローバル行列については，C++線型計算テンプレートライブラリの Eigen3[5] を用いる．これにより，gather 後の固有ベクトルに関する計算を容易に記述できる．
- 使用する固有値計算ライブラリに対応しないグリッド・行列の major が選択された場合には，コンパイル

時アサートによりエラーとなるようにしてある．

4. インターフェースの使用例

図 **1** に開発したインターフェースを用いたプログラム例を掲載する．

5. 各種のスクリプト

作業をなるべく自動化するために，固有値計算ライブラリのインストール，インターフェースのサンプルプログラムのビルド，ベンチマーク測定のスクリプトファイルを用意している．

5.1 固有値計算ライブラリのインストールスクリプト

主要な GNU/Intel/Fujitsu コンパイラに対して，インストールを行うスクリプトを同梱している．ライブラリのダウンロード・解凍なども自動で行うようになっている．

5.2 インターフェースのサンプルプログラムのビルドスクリプト

作成したインターフェースのサンプルプログラムのビルドには cmake[6] を用いる．cmake は与えられたオプションによって自動で Makfile の生成を行うので，人手による makefile の書き換えを行わなくて済む．

5.3 ベンチマーク用のスクリプト

ベンチマーク採取用を簡単かつ統一的に行えるように，python のスクリプトを用意している．このスクリプトでは，ベンチマーク測定用プログラムの MPI 実行からグラフの描画までを自動で行えるようにしてある．各種のスーパーコンピュータを用いたベンチマーク結果については，研究会当日に紹介する．

6. おわりに

ScaLAPACK，Elemental，EigenK の実対称の密行列の固有値計算ライブラリに対して，C++を用いた統一的なインターフェースの開発を行った．また，固有値計算ライブラリのインストール，ベンチマーク採取のためのスクリプトを作成した．作成したインターフェース，スクリプトは，Boost ライセンスで公開している [1]．これを期に統一的なインターフェースのあり方，インストール方法，ベンチマーク採取方法に関する情報の共有を行いたいと考えている．

今後の課題としては，以下が挙げられる．

- 固有値計算ライブラリのインストールのさらなる自動化（コンパイラメーカごとに異なるリンクオプションの自動検出など）
- 複素エルミートの密行列，疎行列に対するインターフェースの開発
- gather/scatter 以外の通信ルーチンの実装

謝辞 本研究では，東京大学物性研究所の共同利用スーパーコンピュータのシステム B(SGI Altix ICE 8400EX) およびシステム C(FX10) を使用している．ここに謝意を表す．

参考文献

[1] 坂下達哉，五十嵐亮，大久保毅，藤堂眞治：竝列固有値計算ライブラリの統一的インターフェス rokko, https://github.com/t-sakashita/rokko.git.
[2] Univ. of Tennessee, Univ. of California, Berkeley and Univ. of Colorado Denver: ScaLAPACK(Scalable Linear Algebra PACKage), `http://www.netlib.org/scalapack/`.
[3] 山田 進，今村俊幸，町田昌彦：「京」コンピュータ用固有値計算ライブラリ EigenK，`http://ccse.jaea.go.jp/ja/download/eigenk.html`.
[4] Poulson, J., Petschow, M. and Bientinesi, P.: Distributed-memory dense linear algebra Elemental, http://code.google.com/p/elemental/.
[5] Guennebaud, G. and Jacob, B.: C++ template library for linear algebra Eigen3, `http://eigen.tuxfamily.org/index.php?title=Main_Page`.
[6] Hoffman, B., Martin, K. and King, B.: The cross-platform, open-source build system CMake, `http://www.cmake.org`.

```
int main(int argc, char *argv[]) {
  MPI_Init(&argc, &argv);
  typedef rokko::grid_row_major grid_major;  // グリッドに関して，row-major を用いる．
  typedef rokko::matrix_row_major matrix_major; // 行列に関して，row-major を用いる．

  rokko::solver solver("eigen_s");  // 固有値計算ルーチンとして eigen_s を使用
  solver.initialize(argc, argv); // 固有値計算ライブラリごとの初期化処理

  MPI_Comm comm = MPI_COMM_WORLD;
  rokko::grid<grid_major> g(comm);  // ２次元グリッドの宣言

  const int root = 0;
  const int dim = 10;

  rokko::distributed_matrix<matrix_major>  mat(dim, dim, g, solver);  // 分散行列の宣言
  rokko::generate_frank_matrix(mat);  // 分散行列 mat に Frank 行列を生成
  mat.print();   // 生成された Frank 行列の確認

  Eigen::VectorXd w(dim);  // 固有値を収める Eigen3 のベクトル
  rokko::distributed_matrix<matrix_major>  Z(dim, dim, g, solver);  // 固有ベクトルを収める分散行列の宣言

  solver.diagonalize(mat, w, Z);  // 固有値・固有ベクトルの計算

  Z.print();  // 各プロセスの分散行列の表示

  Eigen::MatrixXd eigvec_global;
  rokko::gather(Z, eigvec_global, root);  // 固有ベクトルの集約
  if (myrank == root) {
    std::cout << "Computed Eigenvalues= " << eigval.transpose() << std::endl;
    std::cout << "eigvec:" << std::endl << eigvec_global << std::endl;
  }

  solver.finalize();  // 固有値計算ライブラリごとに必要な終了処理
  MPI_Finalize();
  return 0;
}
```

図 1　作成したインターフェースを用いたプログラム例