社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－ARC－157　(33)
2004－HPC－ 97　(33)
2004／3／3

# OmniRPCによるグリッド環境での大規模固有値問題の並列解法

櫻井 鉄也*　早川 賢太郎†　佐藤 三久*　高橋 大介*

本論文では大規模な一般化固有値問題の指定した領域内の固有値と固有ベクトルを求める並列解法について述べる．この方法では複数の大規模な連立一次方程式を解くことで，大規模問題を求めようとする固有値のみを持つような小規模な固有値問題に帰着させる．この連立一次方程式は独立に解くことができ，これらをリモートホスト上で並列に解く．グリッドRPCシステムであるOmniRPC上でこの方法のプログラムを作成した．広域ネットワークを介してPCクラスタを利用する環境において実験を行った．

# A parallel method for large scale eigenvalue problems by OmniRPC in grid environment

Tetsuya Sakurai*, Kentaro Hayakawa†, Mitsuhisa Sato* and Daisuke Takahashi*

In this paper we present a parallel method for finding several eigenvalues and eigenvectors of a large sparse generalized eigenvalue problem. In this method, a small matrix pencil that has only the desired eigenvalues is derived by solving large sparse linear equations. We solve them on remote nodes in parallel. We have implemented and tested the method in a grid environment by using a grid RPC system called OmniRPC. The performance of the presented method on PC clusters that were used over wide-area network was evaluated.

## 1　はじめに

一般化固有値問題 $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ において，行列 $A$, $B$ が大規模で疎な場合に，その固有値の一部のみを求める方法について考える．このような問題は微分方程式の解法や構造解析，量子化学などの分野で現れる．

大規模な一般化固有値問題では行列 $A$, $B$ がその要素に0を多く含む疎行列の場合も多い．また，すべての固有値を必要とするわけではなく，限られた範囲の固有値のみを必要とすることがある．$A$，$B$ が疎行列のときに相似変換を行うと0であった要素に値が入ってしまい，大規模問題では計算量や必要とするメモリーの大きさで問題を生ずる．そのため，疎行列に対しては行列の構造を変えないような解法が有効である．また，並列化による高速化も考慮する必要がある．

*筑波大学電子・情報工学系, Institute of Information Sciences and Electronics, University of Tsukuba
†筑波大学理工学研究科, Master's Program in Science and Engineering, University of Tsukuba

指定した領域内にある固有値と対応する固有ベクトルを求める方法として，複素平面上の周回積分を用いた方法 [5] がある．この方法では，複数の連立一次方程式を解くことで指定した領域内の固有値のみの小規模な問題に帰着させる．行列が大規模なときには，この連立一次方程式の解法が計算の大部分を占め，これを並列化することで高速化を図ることができる．このような解法では粗粒度の並列性を持つため，グリッドによる並列化が可能である．

本論文では，グリッド環境での並列プログラミングのためのGrid RPCシステムOmniRPC [6] 上でこの固有値解法を並列化した．OmniRPCは，グリッド環境において遠隔計算機への遠隔手続き呼び出し（RPC: Remote Procedure Call）を用いて，遠隔計算機資源で並列プログラムの実行を可能にするミドルウェアである．master-worker型の並列プログラミングをサポートし，特にグリッド環境において典型的なアプリケーションであるパラメータ検索などのアプリケーションを効率的にサポートす

る機能がある．初期化のための大量のデータ転送や計算が必要な場合に，これを再利用することで効率化する自動初期化実行モジュール機能を提供している．認証を行うグリッド環境として，Globus の他，ssh による認証も可能で，ファイヤウォールのある遠隔の計算機についても計算資源として利用できるなどの特徴を持つ．

本論文で示す方法は，はじめに行列のデータを各リモートホストに転送すると，その後の計算においてリモートホスト間の通信を必要としない．そのため，グリッド環境において通信による効率の低下を起こしにくい．分散した計算機環境において数値実験を行い，その効率について検証した．

## 2 指定した領域内の固有値を求める方法

行列 $A, B \in \mathbb{C}^{n\times n}$ とし，$\lambda_1, \ldots, \lambda_d$ $(d \le n)$ は $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ の有界な固有値とする．$B$ が特異のときにはいくつかの固有値は無限大となるため，$d < n$ となる．

ここで，零でないベクトル $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{C}^n$ に対して関数 $f(z)$ を

$$f(z) := \boldsymbol{u}^H (zB - A)^{-1}\boldsymbol{v} \tag{1}$$

と定義する．$z$ が固有値のとき $(zB-A)^{-1}$ は特異となり，これは関数 $f(z)$ の特異点となる．文献 [5] では，式 (1) のように定義した $f(z)$ が $\lambda_1, \ldots, \lambda_d$ を極に持つ有理式で表されることが示されている．この $f(z)$ に対して指定した領域内の極を求める方法 [2] を適用する．固有値の分布と方法の性質の解析には文献 [3, 4] に示されている方法が利用できる．

$\Gamma$ を原点の周りを正の方向に 1 周する単一閉曲線とし，$\Gamma$ 内に $m$ 個の相異なる固有値 $\lambda_1, \ldots, \lambda_m$ があるとする．$\mu_j$ を

$$\mu_j = \frac{1}{2\pi \mathrm{i}} \int_\Gamma z^j f(z) dz, \quad j = 0, 1, \ldots$$

とし，

$$H_k = \begin{pmatrix} \mu_0 & \mu_1 & \cdots & \mu_{k-1} \\ \mu_1 & \mu_2 & \cdots & \mu_k \\ \vdots & \vdots & & \vdots \\ \mu_{k-1} & \mu_k & \cdots & \mu_{2k-2} \end{pmatrix},$$

および

$$H_k^< = \begin{pmatrix} \mu_1 & \mu_2 & \cdots & \mu_k \\ \mu_2 & \mu_3 & \cdots & \mu_{k+1} \\ \vdots & \vdots & & \vdots \\ \mu_k & \mu_{k+1} & \cdots & \mu_{2k-1} \end{pmatrix}$$

とおく．$k = m$ のとき次の定理を得る．

**Theorem 1** $\lambda_1, \ldots, \lambda_m$ は行列束 $A - \lambda B$ の $\Gamma$ 内にある相異なる固有値とする．このとき，行列束 $H_m^< - \lambda H_m$ の固有値は $\lambda_1, \ldots, \lambda_m$ である．

したがって，大規模な一般化固有値問題 $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ の指定した領域 $\Gamma$ 内にある固有値 $\lambda_1, \ldots, \lambda_m$ を求める問題は，一般化固有値問題 $H_m^<\boldsymbol{x}' = \lambda H_m \boldsymbol{x}'$ に帰着する．$\Gamma$ を適当に選んで $m$ がそれほど大きくない場合には，この問題はもとの問題と比べてはるかに小規模になる．

## 3 円内の固有値を求める方法

領域が中心 $\gamma$，半径 $\rho$ の円の場合には，円周上に $N$ 個の等間隔点

$$\omega_j := \gamma + \rho e^{\frac{2\pi \mathrm{i}}{N}(j+1/2)}, \quad j = 0, 1, \ldots, N-1$$

をとり，この点での関数値

$$f(\omega_j) = \boldsymbol{u}^H (\omega_j B - A)^{-1}\boldsymbol{v}, \quad j = 0, 1, \ldots, N-1$$

を用いて，円内の固有値のみを持つ小規模な一般化固有値問題に帰着させる．

$\mu_k$ を求める周回積分を d 台形則で近似することで，

$$\mu_k \approx \hat{\mu}_k := \frac{1}{N} \sum_{j=0}^{N-1} (\omega_j - \gamma)^{k+1} f(\omega_j), \quad k = 0, 1, \ldots,$$

とする．また，

$$\boldsymbol{y}_j := (\omega_j B - A)^{-1}\boldsymbol{v}, \quad j = 0, 1, \ldots, N-1$$

とおく．

$$\hat{H}_m := [\hat{\mu}_{i+j-2}]_{i,j=1}^m,$$

および

$$\hat{H}_m^< := [\hat{\mu}_{i+j-1}]_{i,j=1}^m$$

とし，$\hat{H}_m^<, \hat{H}_m$ から $\lambda_1, \ldots, \lambda_m$ の近似値を求める．

以下に固有値を求めるアルゴリズムを示す [5]．

**Algoritheorem:**

入力: $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{C}^n$, $N$, $m$, $\gamma$, $\rho$
出力: $\hat{\lambda}_1, \ldots, \hat{\lambda}_m$
Step 1. $\omega_j \leftarrow \gamma + \rho \exp(2\pi \mathrm{i}(j+1/2)/N)$, $j = 0, \ldots, N-1$ とする
Step 2. $(\omega_j B - A)\boldsymbol{y}_j = \boldsymbol{v}$, $j = 0, \ldots, N-1$ を解く
Step 3. $f_j \leftarrow \boldsymbol{u}^H \boldsymbol{y}_j, j = 0, \ldots, N-1$ とする
Step 4. $\hat{\mu}_k, k = 0, \ldots, 2m-1$ を求める
Step 5. 行列束 $\hat{H}_m^< - \lambda \hat{H}_m$ の固有値 $\zeta_1, \ldots, \zeta_m$ を求める
Step 6. $\hat{\lambda}_j \leftarrow \gamma + \zeta_j$, $j = 1, \ldots, m$ とする

対応する固有ベクトルも Step 2 で求めたベクトル $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{N-1}$ を用いて求めることができる [5].

# 4 OmniRPC 上での並列化

行列 $G_j$ を $G_j := \omega_j B - A$ とおく．$f(z)$ の $z = \omega_j$ における値は，$N$ 個の連立一次方程式

$$G_j \boldsymbol{y}_j = \boldsymbol{v}, \quad j = 0, 1, \ldots, N-1$$

を解き，$\boldsymbol{u}^H \boldsymbol{y}_j$ とすることで求められる．行列 $A$, $B$ が大規模で疎のときには $G_j$ もまた大規模で疎であり，この連立一次方程式を解く時間が全体の計算のほとんどを占めている．これらの方程式を複数の CPU で解くことで並列化を行う．

OmniRPC は，グリッド環境において遠隔計算機への遠隔手続き呼び出し（RPC: Remote Procedure Call）を用いて，遠隔計算機資源で並列プログラムの実行を可能にするミドルウェアである．master-worker 型の並列プログラミングをサポートし，特にグリッド環境において典型的なアプリケーションであるパラメータ検索などのアプリケーションを効率的にサポートする機能がある．初期化のための大量のデータ転送や計算が必要な場合に，これを再利用することで効率化する自動初期化実行モジュール機能を提供している．認証を行うグリッド環境として，Globus の他，ssh による認証も可能で，ファイヤウォールのある遠隔の計算機についても計算資源として利用できる．

OmniRPC 上での並列化は，以下のように行った．

1) あらかじめ $A$, $B$, $\boldsymbol{u}$, $\boldsymbol{v}$ のデータをクライアントホストから PC クラスタの各ノードに送る．
2) $N$ 個の連立一次方程式を非同期遠隔手続き呼び出しを行うことにより，各ノードで並列に処理する．
3) 各ノードの計算結果をクライアントホストに集めて固有値，および固有ベクトルを求める．

ステップ 1) においていったん行列のデータを各ノードに送った後は，ステップ 2) の実行中にノード間で通信する必要はない．OmniRPC が利用可能なリモートホストに順次計算を割り振るため，ユーザはプログラム中では単にステップ 2) のモジュールをリモートコールするだけでよく，スケジュールの管理などは必要としない．

OmniRPC を用いるときのプログラムソースには以下に示すような記述を行う．

```
. . .
call OmniRPC_Init
call OmniRPC_Module_Init(. . .)
. . .
do j = 0, N-1
    . . .
   call OmniRPC_Call_Async(. . .)
    . . .
enddo
call OmniRPC_Wait_All
. . .
```

ここで `OmniRPC_Module_Init(. . .)` において，リモートホストが最初に呼び出されたときにのみ行う処理を記述する．本方法では行列のデータをここで送る．2 回目以降呼び出されたときは，初回に受け取ったデータを利用し，再度行列のデータを送らない．そのため，データ転送のための時間が節約される．

DO ループ中の `OmniRPC_Call_Async(...)` では，リモートホスト上の連立一次方程式を解くサブルーチンを非同期で呼び出している．アルゴリズム中のステップ 2 に相当する計算を行う．このとき，サブルーチン名や引数の指定を行うのみで，どのリモートホストで実行するかを明示する必要はない．OmniRPC が自動的に空いているリモートホストに処理を割り当てる．

ループ後の `OmniRPC_Wait_All` において，すべてのリモートホストでの処理が終わるのを待つ．その後，アルゴリズム中のステップ 3 以降の処理を行

う．大規模な問題では，ステップ 2 に要する時間がほとんどを占め，ステップ 3 以降の処理に要する時間はわずかである．

## 5　数値例

本方法の数値例を示す．実験環境として，8 ノードの PC クラスタ（Pentium4 Xeon 2.4GHz，dual CPU，Linux，1000BASE-T 接続，メモリ 1GB）を用いて，クライアントホスト（Pentium III 1GHz，dual CPU，Linux，メモリ 512MB）から実行した．なお，クライアントホストから PC クラスタは，広域ネットワークを経由して接続されている．

数値例として，有限要素法を用いた水分子の電子状態計算で現れる固有値問題 [1] から得られる行列を用いた．行列 $A$，$B$ はともに実対称で，行列のサイズは $n = 9274$，非零要素数は 375208 である．$N = 128$ とし，中心 $\gamma = -9.0$，半径 $\rho = 0.3$ の領域にある 4 個の固有値，および固有ベクトルを求めた．計算は倍精度で行い，ベクトル $\boldsymbol{u}$ と $\boldsymbol{v}$ の要素は乱数で与えた．

この例では，行列 $A$，$B$ がともに対称であるため，$\omega_j B - A$ は複素対称行列となる．そのため，連立一次方程式は複素対称行列向きの反復解法である COCG 法 [7] で解いた．また，$f(\bar{z}) = \overline{f(z)}$ となるため，関数値は $f(\omega_0), \ldots, f(\omega_{N/2-1})$ のみを求め，$f(\omega_{N-1-j}) = \overline{f(\omega_j)}$ とした．

プロセス数を変えたときの計算時間を表 1 に示す．連立一次方程式をどのノードで解くかは OmniRPC が自動的に割り当てている．表より，1 ノードのときと比べて 8 ノードのときの計算時間は約 7 分の 1 になっていることがわかる．

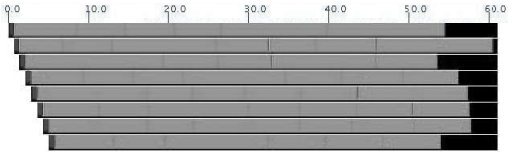


図 1: プロセスの様子（自動割り当てあり）

図 1 に，このときのプロセスの割り当ての様子を示す．各横棒はノード毎の処理の様子を表しており，左端の色の濃い部分は 1 回目の呼び出しで行列のデータを転送していることを表している．1 つめのノードにデータを送り終えた後，すぐに次のノードにデータを送り始めていることがわかる．処理の終わったノードに対して次の処理を割り当てている．

表 1: 数値例 ($N = 128$，自動割り当てあり)

| プロセス数 | 時間 [秒] | 速度向上率 |
|---|---|---|
| 1 | 428 | 1.00 |
| 2 | 219 | 1.95 |
| 3 | 149 | 2.87 |
| 4 | 113 | 3.79 |
| 5 | 90 | 4.76 |
| 6 | 78 | 5.49 |
| 7 | 67 | 6.39 |
| 8 | 60 | 7.13 |

表 2 では，各ノードに 8 分の 1 ずつの数の方程式を始めに割り当てた場合の結果を示す．この場合には，計算時間が増加していることがわかる．図 2 に，このときのプロセスの割り当ての様子を示す．

表 3 に，$N = 64$ の場合の結果を示す．この場合には，方程式を解く時間が少なくなるため，相対的にデータ転送の時間の割合が大きくなり，速度向上率は低下している．

表 2: 数値例 ($N = 128$，自動割り当てなし)

| プロセス数 | 時間 [秒] | 速度向上率 |
|---|---|---|
| 1 | 418 | 1.00 |
| 2 | 222 | 1.88 |
| 3 | 151 | 2.77 |
| 4 | 115 | 3.63 |
| 5 | 92 | 4.54 |
| 6 | 76 | 5.50 |
| 7 | 70 | 5.97 |
| 8 | 63 | 6.63 |

## 6　おわりに

本論文では，グリッド環境での並列プログラミングのための Grid RPC システム OmniRPC 上で，


図 2: プロセスの様子（自動割り当てなし）

表 3: 数値例 ($N = 64$)

| プロセス数 | 時間 [秒] | 速度向上率 |
|---|---|---|
| 1 | 213 | 1.00 |
| 2 | 109 | 1.95 |
| 3 | 75 | 2.84 |
| 4 | 58 | 3.67 |
| 5 | 47 | 4.53 |
| 6 | 42 | 5.07 |
| 7 | 36 | 5.92 |
| 8 | 33 | 6.45 |

与えられた領域内の固有値を求める解法を並列化した．OmniRPC の持つ自動初期化やプロセスの自動割り当てを利用することで，容易に並列プログラムが得られた．

本方法は遠隔ノード間でのデータ通信を必要としないため，グリッド環境での計算に適しており，広域ネットワークを介して PC クラスタを利用した実験でも高い効率を示した．

より大規模な並列環境でのテストや，実用的な問題への適用などが今後の課題である．

## 参考文献

[1] Hyodo, S., Meso-scale fusion: A method for molecular electronic state calculation in inhomogeneous materials, *Proc. 15th Toyota Conference, Special issue of J. Comput. Appl. Math.* **149** (2002), 101–118.

[2] Kravanja, P., Sakurai, T., and Van Barel, M., On locating clusters of zeros of analytic functions, *BIT* **39** (1999), 646–682.

[3] Kravanja, P., Sakurai, T., Sugiura, H., and Van Barel, M., A perturbation result for generalized eigenvalue problems and its application to error estimation in a quadrature method for computing zeros of analytic functions, *J. Comput. Appl. Math.* **161** (2003), 339–347.

[4] Sakurai, T., Kravanja, P., Sugiura, H., and Van Barel, M., An error analysis of two related quadrature methods for computing zeros of analytic functions, *J. Comput. Appl. Math.* **152** (2003), 467–480.

[5] Sakurai, T., and Sugiura, H., A projection method for generalized eigenvalue problems, *Proc. JCJS2002, Special issue of J. Comput. Appl. Math.* **159** (2003), 119–128.

[6] Sato, M., Boku, T., and Takahashi, D., OmniRPC: a Grid RPC System for parallel programming in cluster and grid environment, *Proc. CCGrid 2003* (2003), 206–213. (http://www.omni.hpcc.jp/OmniRPC)

[7] van der Vorst, H. A., and Melissen, J. B. M., A Petrov-Galerkin type method for solving $Ax = b$, where $A$ is a symmetric complex matrix, *IEEE Trans. on Magnetics* **26** (1990), 706–708.