ペタスケール計算機環境に向けた固有値ソルバの開発

片桐　孝洋

東京大学情報基盤センター　特任准教授

１．はじめに

2008 年 6 月、人類は始めてペタフロップスマシンを持った。米国ロスアラモス国立研究所に設置された Roadrunner である。この計算機は 6,948 個のデュアルコア Opteron と 1 万 2,960 個の Cell BE を組み合わせたものである。2008 年 11 月には、米国オークリッジ国立研究所の Jaguar が 1 ペタフロップス超えの性能を出し、二番目のペタフロップスマシンとなった。それぞれの CPU（コア）は、129,600 コア、150,152 コアである。このことは、ペタフロップスの演算性能を達成する計算機環境（ペタスケール計算機環境）では、10 万プロセッサもの超並列性を、1 つのアプリケーション内で達成しなくてはならないことを意味している。

一方、ペタスケール計算機環境において、重点開発されるソフトウエアは科学技術計算のための数値シミュレーションである。この数値シミュレーションにおいて重要となる処理は、大きく分けて以下の 3 通りになるケースが多い。すなわち、（1）数学的に定式化する前処理部分、たとえば、メッシュ生成や行列生成部分。（2）定式化された問題を求解する部分。（3）計算結果を可視化する部分。ここで、ペタスケール計算機環境の性能を十分に達成しないといけない処理は、（1）と（2）になる。特に（2）はソルバと呼ばれる部分であり、扱う問題によっては、全体の実行性能の 90%を占める処理になることが知られている。

このソルバで解くべき問題は、多くの場合、連立一次方程式か固有値問題の求解となる。連立一次方程式の求解の場合、多くは行列要素の多数が 0 となる疎行列が対象となる。一方で、固有値問題を解く場合には、行列要素に 0 が少ない密行列で、かつ対称行列となる場合が少なくない。対称かつ密行列の固有値問題において、固有値のみ必要／固有値・固有ベクトル双方が必要という場合がある。さらに、必要な固有値・固有ベクトルが少数／ほとんどすべてが必要という場合ごとに、適切な数値アルゴリズムが異なる。

本稿で取り扱う固有値問題ソルバは、対称密行列の全固有値・全固有ベクトルを計算する＜最も＞演算量が必要とされる場合である。この処理が必要とされる分野は、量子化学計算分野などである。

本原稿の目的は、以下の通りである。従来から開発してきた、並列固有値ソルバ ABCLib_DRSSED を当センターの最新スパコン T2K オープンスパコン（HITACHI HA8000 クラスタシステム）で性能評価することで、次世代スパコン（ペタコン、京速計算機）に代表される 10 万並列が必要と予想されるペタスケール計算機環境での問題を概観することにある。

２．並列固有値ソルバ ABCLib_DRSSED

（１）概要

ABCLib_DRSSED[1]は、実数対称密行列の固有値問題の任意の個数の固有値・固有ベクトルを計算できる機能をもつ並列数値計算ライブラリである。現在公開されている ABCLib_DRSSED ver. 1.04 は、MPI-1 と Fortran90 で実装されている。

数値アルゴリズムとして、対称固有値ソルバに利用される対称密行列用 Householder 三重対角化、対称三重対角行列用の固有値計算のための二分法、対称三重対角行列用の固有ベクトル計算のための逆反復法、および対称密行列の固有ベクトル計算のための Householder 逆変換の各ルーチンが利用できる。さらに、密行列用 QR 分解ルーチンも提供している。

ABCLib_DRSSED は、ベクトル化（もしくは、疑似ベクトル化）機能を有するプロセッサをもつノードにおいて、超並列環境（2000 年頃において 1000 並列以上）で高速に動作する方式が採用されている[2]。当時、当センターに導入されていたスーパーコンピュータ HITACHI SR2201 において、日立製作所が最適化した BLAS(Basic Linear Algebra Subprograms)を利用した ScaLAPACK の同種ルーチンに対して、超並列環境で 5.7 倍ほど高速であった（図 1)。

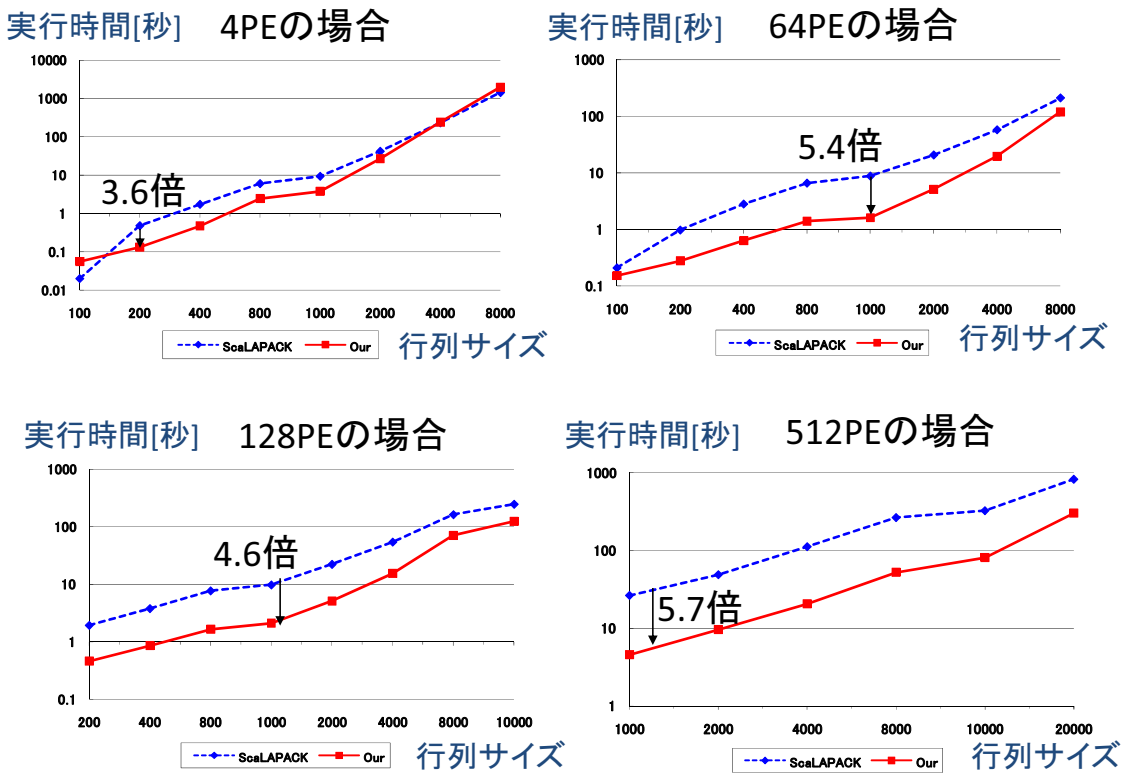

図 1　ScaLAPACK との性能差（Householder 三重対角化、HITACHI SR2201）

（２）解法の説明

問題と解法を定式化する。いま、対称密行列 $A \in R^{n*n}$、実数 $\lambda \in R$、実数ベクトル $x \in R^n$ とすると、以下の標準固有値問題

$$A x = \lambda x \qquad \ldots (1)$$

の解 λ を固有値、ベクトル x を固有ベクトルとよぶ。いま、式 (1) の固有値 n 個を対角要素に並べた行列 Λ を、$\Lambda=diag(\lambda_1, \lambda_2, \ldots, \lambda_n)$、固有値 λ_i に対応する固有ベクトル x_i を並べて構成された行列 X を、$X=(x_1, x_2, \ldots, x_n)$ とすると、式(1)は以下のように書ける。

$$A X = X \Lambda \qquad \ldots(2)$$

いま、式(2)の固有値行列 Λ、固有ベクトル行列 X を求める。このとき、以下の手順で解く方法が多くの固有値ソルバで採用されている。

(Step1) (三重対角化) 行列 A を相似変換により、三重対角行列 T に変換する ($QAQ = T$)
(Step2) 三重対角行列 T の固有値行列 Λ を求める
(Step3) 三重対角行列 T の固有ベクトル行列 Y を求める
(Step4) (逆変換) Y を行列 A の固有ベクトル行列 X に変換する ($X = QY$)

図 2　全固有値・全固有ベクトル求解手順

計算量に言及すると、(Step1)及び(Step4)は $O(n^3)$ である。(Step3)は解法と問題の性質に依存し変化し、$O(n)$～$O(n^3)$ であることが知られている。したがって、ソルバ全体の演算量は $O(n^3)$ となる。メモリ量については、入出力行列 A と X が密行列であるため $O(n^2)$ となるのは自明であるが、解法に必要なメモリ量もソルバ全体で $O(n^2)$ となることが知られている。

（３）並列解法

図 2 の手順を実現する解法は多数提案されており、特に逐次処理を中心に開発されてきた。それら逐次解法のうち、並列化に向くアルゴリズムとして ScaLAPACK などの主流ライブラリで採用されているのが、Householder 変換を用いた方法である。Householder 変換の逐次アルゴリズムの詳細は割愛するが、並列環境において重要な点は以下である。

(a) どのように行列 A を各コアに分散させるのか
(b) どのように行列 X を分散収納して戻るのか

(a) は、通信量と負荷バランスという並列実行性能に影響する要因である。(b) は解の演算精度と並列実行性能に影響する要因であり、均等に分割して収納すると採用する解法の特質により、固有ベクトルの精度が保てない場合[1]が生じる。

(a) について負荷分散の観点から、行列 A を二次元に循環するように分散する方式が有効であることが知られている。すなわち、$A=(a_{ij}), (i, j=1, 2, \ldots, n)$ とするとき、

$$a_{ij} \rightarrow P(i \ \% \ nprocx, \ j \ \% \ nprocy) \qquad \ldots(3)$$

への写像となる。ここで、$P(x, y)$ は、2 次元で表されたコア番号 $(x=0, \ldots, nprocx-1, y=0, \ldots, nprocy-1, \ nprocx*nprocy=$コア数$)$ であり、%はモジュロ演算子である。このような分散方式を**二次元サイクリック分散方式**と呼ぶ。ScaLAPACK では、二次元サイクリック分散の対象は、ある幅 m をもつ正方行列 $m \times m$ の単位で行うなどバリエーションがあるが、本質的に二次

[1] 特に、近接する固有値に対する固有ベクトルの演算精度が低下する。

元サイクリック分散方式である点を言及しておく。

(b) について、まず解法として、図 2(Step2)で二分法による全固有値の求解、(Step3)で逆反復法による全固有ベクトルを求解するアルゴリズムが古典的な方法として知られている。この場合、固有ベクトルが十分に離れている場合には、固有ベクトル計算ごとに演算が独立して行える上に、固有ベクトルの直交精度が十分である場合が多く、きわめて並列化に向く解法となる。一方で、もし固有値が近接密集する場合、各固有ベクトルの計算が逐次化される。いま密集する固有値群 m 個を λ_k, λ_{k+1}, ..., λ_{k+m-1} とすると、逆反復のたびに直交化処理をするため、固有ベクトルの計算順序が、x_k, x_{k+1}, ..., x_{k+m-1} の順に行わなくてはならない。加えて、直交化のための演算量は $O(n \times m^2)$ と大きい。最悪の場合は全固有値が密集する場合であり $m=n$ なので $O(n^3)$ となり、かつ並列化ができないのでコア数が増加するにつれ、ほとんどの実行時間が(Step3)の時間になることが知られている。

以上の並列解法の特徴をまとめると、以下のようになる。

1. Householder 三重対角化を効率よく並列化するため、行列 A は二次元サイクリック分散されて並列固有値ソルバを呼び出す。
2. 固有値が密集しない場合、および、固有ベクトルの直交精度を無視できる場合には、ほとんどの時間が三重対角化と逆変換の時間となる。
3. 固有値が密集する場合は、古典的な手法を用いると、ほとんどの時間が三重対角行列の固有ベクトルを計算する時間となる。

（4）先進的な固有ベクトル解法の導入

問題 3 は古典的な解法である逆反復法を採用して並列化する場合の問題点であった。現在では古典的な解法に加え、先進的な解法がソルバで導入されている。２つ知られている方法があり、１つは分割統治法 (Divide and conquer) 法によるもの、もう一つが MRRR 法(Multiple Relatively Robust Representations)[3][4]によるものである。

両者の特徴を以下にまとめる。

- **分割統治法**：直交精度が多くの場合に良い。並列性はあるが、分散環境では通信が必要である。
- **MRRR 法**：直交精度が 1 桁～2 桁程度、経験的に古典的解法に対して悪い。並列性がきわめて高く、分散環境でも通信を必要としない。

以上から、先進的な解法でも直交精度と並列実行性能のトレードオフの問題があり、ユーザ要求により適する解法が異なる。本稿では、10 万コアからなるペタスケールな超並列計算機環境での実行を目指しているので、並列性が極めて高い MRRR 法を、図 2(Step2)(Step3)で採用する。

３．性能評価

T2K オープンスパコン（東大）(HITACHI HA8000 クラスタシステム、以降、T2K オープンスパコン）を、ペタスケール計算機環境に向けたテストマシンと位置づけ性能評価を行う。

（１）計算機環境

T2K オープンスパコンのノードは、AMD Opteron 8356 (2.3GHz, 4 コア)を 4 台 (4 ソケット) 搭載しており、ノード当たりのメモリ量は 32GB である。理論最大演算性能は、ノードあたり 147.2GFLOPS である。キャッシュサイズは、L1 命令キャッシュ、L1 データキャッシュともに 64Kbytes であり、2 Way Associativity（ライトバック、3 サイクル）である。また、L2 キャッシュは 512Kbytes である。L1 キャッシュと L2 キャッシュはコアごとに独立している。L3 キャッシュ 2048Kbytes であり、ソケット内で共有されている。各ソケットには、ローカルメモリ 8GB（ノード当たり 8GB×4 ソケット＝32GB）がある。ソケット間は HyperTransport という高速通信網で連結されており、ローカルメモリへのデータアクセス時間はソケットからの距離により異なる。だだし、ソケット間のキャッシュデータの一貫性はハードウエア的に保障される。ccNUMA（cache coherent non-uniform memory access）型のアーキテクチャ構成である。

通信性能は運用クラスタ群で異なる。ここでは Miri-10G が 4 本実装されており、最大で 5GB/sec の双方向性能を有する A 群を利用している。通常のユーザ環境では、最大で 64 ノード（64 ノード×16 コア＝1,024 コア）が利用できる。月に一度、256 ノード（256 ノード×16 コア＝4,096 コア）の大規模ジョブ実行が可能である。

コンパイラは、日立最適化 Fortran90 V01-00-/A で、コンパイラオプションは、**ピュア MPI**（MPI のみによる実行）で、-Oss -noparallel である。

（２）評価する固有値ソルバ

今回性能評価を行った固有値ソルバは、ABCLib_DRSSED ver.1.04［5］に、LAPACK 3.1.1 における MRRR 法ルーチン（dstegr）相当のコードを実装し、新規開発したものである。(現状では非公開。公開準備中。)

dstegr は分散並列版ではないため、各コアに均等な担当範囲を決め、その範囲について独立に固有ベクトル計算する実装となっている。ここで均等な担当範囲とは、コア数を p とするとき、n/p づつの範囲で分担することであるが、n/p が割り切れないとき、MPI によるランクが小さいコアから順に n/p の切り捨て数に 1 を加算した数を担当する。

解法上の注意は、もし固有値分布が、MRRR 法が保障する範囲外で極度に密集する場合、ノード間をまたいだ固有値に対応する固有ベクトルの直交性が保障されないことである。この保障を行うためには、MRRR 法を分散並列化するしかないが、並列実装がきわめて複雑となるため現状では実現されていない。

（３）テスト行列と演算精度

テスト行列には､Frank 行列を利用した｡Frank 行列の固有値分布は解析解が知られており、解析解との誤差が算出できる。また、行列サイズが大きくなるにつれ密集固有値の割合が大きくなり、古典解法では 2,000 次元以上ではすべてが密集固有値と判定される。

Frank 行列を用いた式(2)の標準固有値問題の演算精度について、以下の 3 つの指標がある。

1. 各固有値における解析解との誤差の最大値

   $max_i\ (|\lambda_i - \lambda^*_i|),\quad (i=1,2,\ldots,n)$、

   ここで、λ_i：解析値による固有値、λ^*_i：計算による固有値。

2. 固有ベクトルの直交性

   $|\ X^T X - I\ |_F$ 、

   ここで、$|\cdot|_F$はフロベニウスノルム。

3. 元の行列 A に対する固有方程式の各残差ベクトルの最大値

   $max_i\ (|A\ x^*_i - \lambda^*_i\ x^*_i|),\quad (i=1,2,\ldots,n)$、

   ここで、x^*_i：計算による固有ベクトル値。

（4）MRRR 法と古典解法の実行性能と演算精度

図 3～図 5 に、Frank 行列を入力とした場合の、MRRR 法、逆反復法（直交化なし版）、および逆反復法（修正 Gram-Schmidt 法（MGS）による直交化版）の、1 ノード（16 コア実行）の実行時間を載せる。

図 3～図 5 から、以下の傾向が読み取れる。

- MRRR 法は、直交化なしの逆反復法と実行時間がほぼ同じである。
- MGS 直交化を行った逆反復法の実行は、MRRR 法の実行に対し約 8 倍遅くなる。
- MRRR 法では、ほとんどの実行時間（98%以上）は、三重対角化と逆変換の時間である。

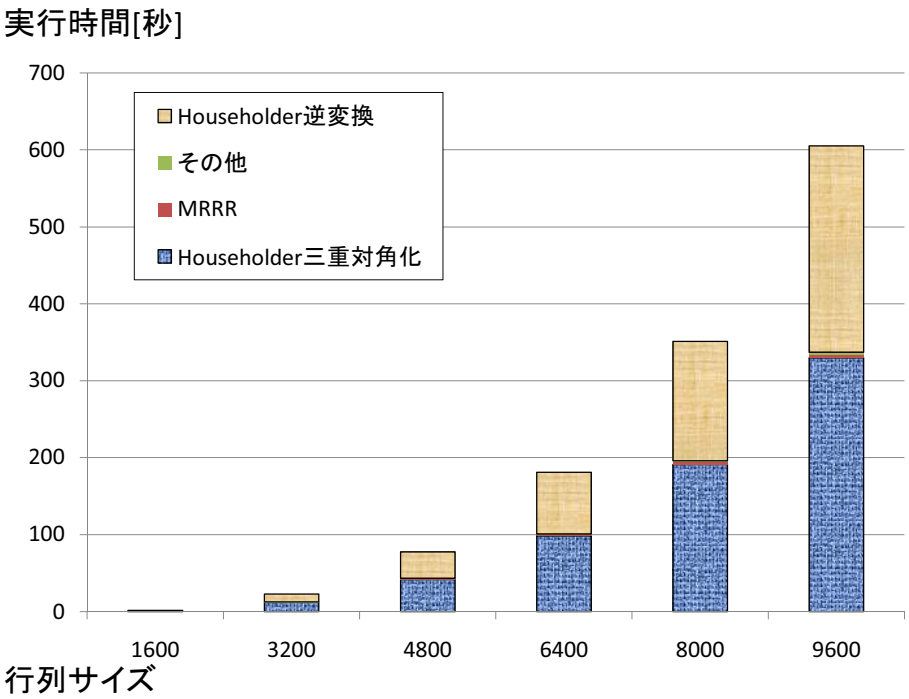

図 3　固有値ソルバ実行時間（MRRR 法、1 ノード(16 コア)、Frank 行列）

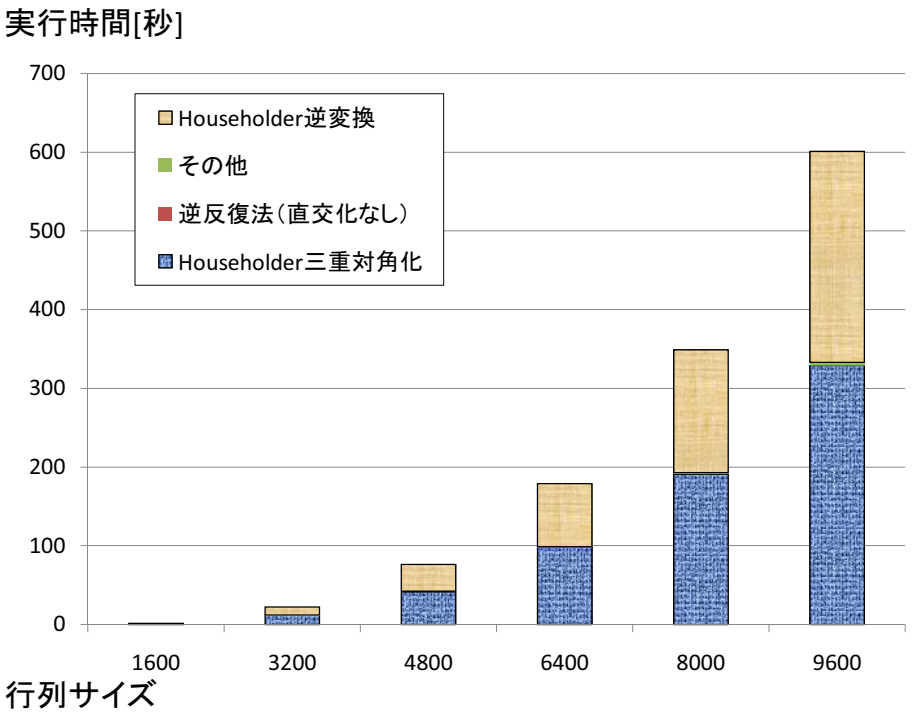

図 4　固有値ソルバ実行時間（逆反復法（直交化なし）、1 ノード(16 コア)、Frank 行列）

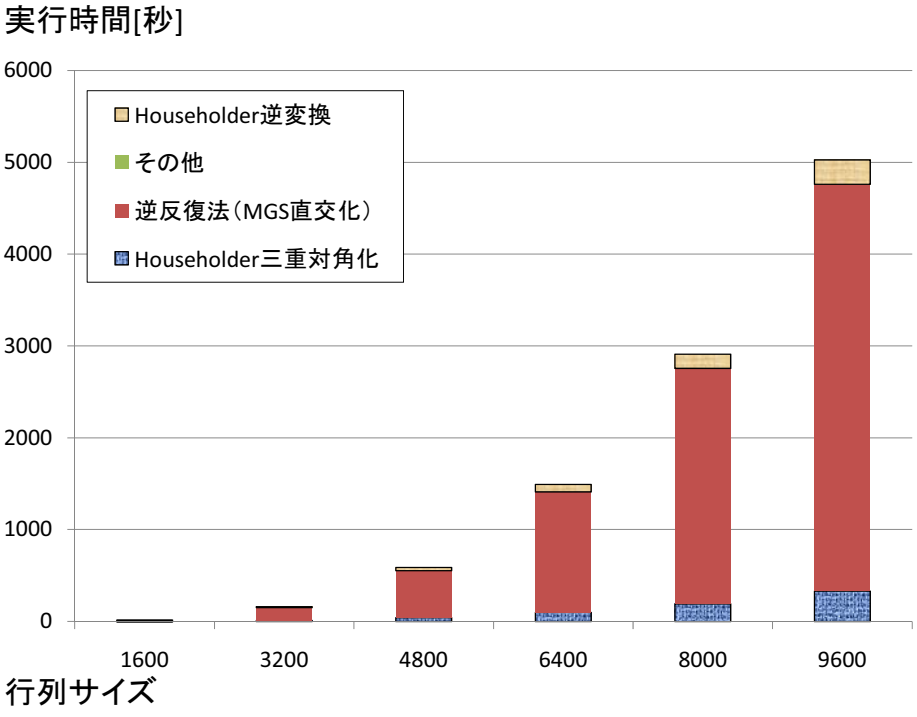

図 5　固有値ソルバ実行時間（逆反復法（ＭＧＳ直交化）、1 ノード(16 コア)、Frank 行列）

次に演算精度について検証する。図 6 は、Frank 行列の固有値の解析解に対する最大誤差である。図 6 から、MRRR 法とその他の解法との固有値計算に対する誤差は生じないと言える。

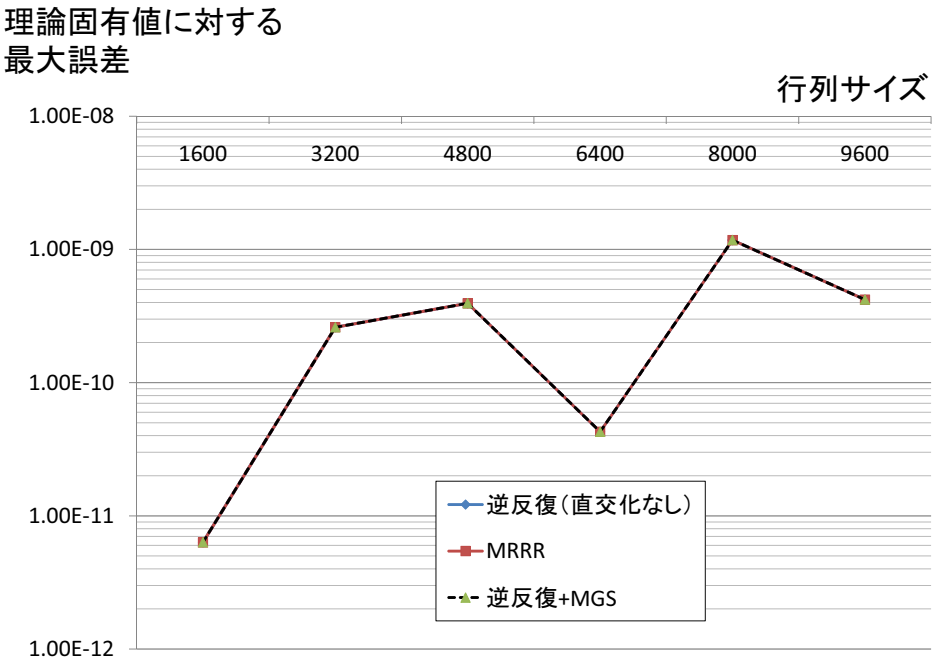

図 6　固有値の最大誤差（1 ノード(16 コア)、Frank 行列）

図 7 に、固有ベクトルの直交性をフロベニウスノルムにより評価した結果を載せる。

図 7 では、3,200 次元より大きくなると、直交化をしない逆反復法では直交性が破たんすることがわかる。一方、直交性が一番良いのは MGS 直交化を行った逆反復法であるが、これは逆反復ごとに陽に直交化することから当然の結果である。興味深いのは MRRR の直交性であり、陽に直交化を行った逆反復法に対し、フロベニウスノルムの観点で 2 桁程度の精度悪化を生じるだけである。また MRRR では、直交性の破たんがないのも特筆すべき結果である。

図 8 は、元の行列 A に対する残差ベクトルについて、2 ノルムの観点で評価したもののうち、最大値について載せたものである。図 8 から、MRRR 法では、逆反復法に対して行列サイズが小さい場合に若干の精度悪化が見受けられるものの、大規模問題になると減少する。したがって、ほぼ同程度の誤差で計算できることがわかる。

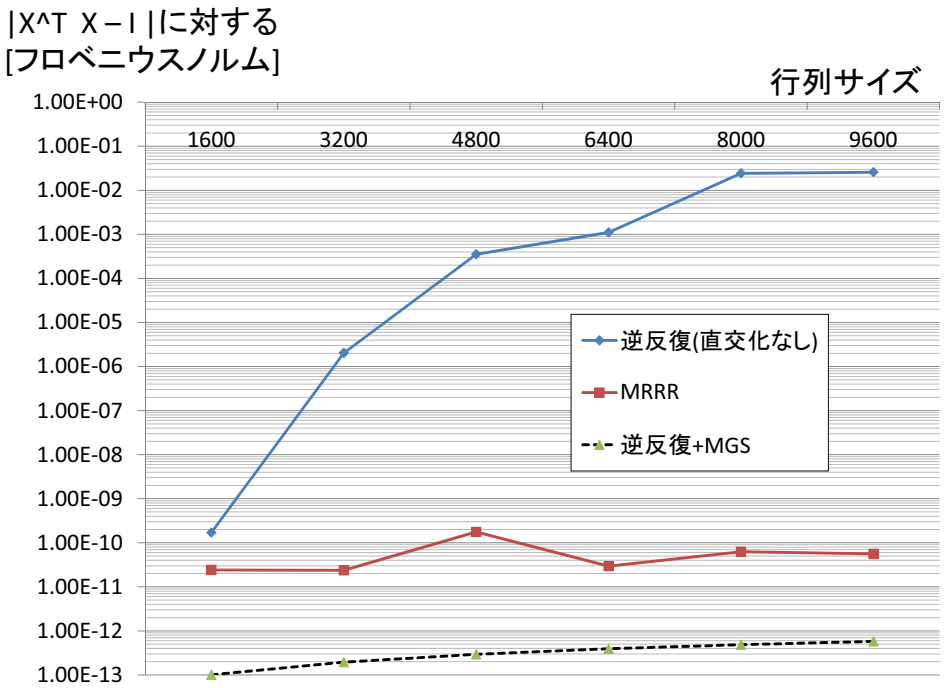

図 7　固有ベクトルの直交性（1 ノード(16 コア)、Frank 行列）

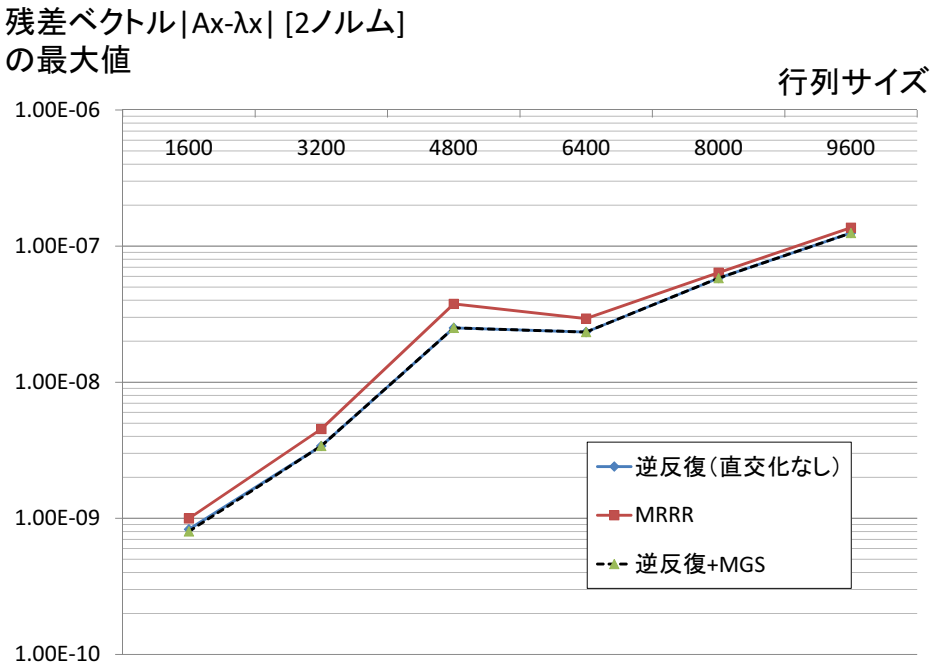

図 8　元の行列 A に対する残差ベクトルの 2 ノルムの最大値（1 ノード(16 コア)、Frank 行

以上の演算精度に関する評価のまとめとして、以下が言える。

- MRRR 法は、直交化しない逆反復法と同等の実行性能をもつ。直交化しない逆反復法は通信がないので、超並列向きの解法である。したがって MRRR 法も、超並列向きの実行性能が実現できる解法である。
- MRRR 法は、陽に直交化した逆反復法に対して、直交性が 2 桁程度悪化するだけであり、直交化なしの逆反復法のように直交性の破たんを起こすことがない。

- MRRR 法を採用する場合、98%以上の時間は三重対角化と逆変換の時間となる。したがって、この 2 処理をいかに高速化するかが高性能化の鍵となる。

以上から、ペタスケール計算機環境向きの固有値ソルバにおいては、2 桁程度の直交化の崩れが許容できるのであれば、MRRR 法はきわめて有効な解法になりうることが結論付けられる。

（5）大規模問題の実行性能

現在公開されている ABCLib_DRSSED ver.1.04 に MRRR 法を実装した場合における 64 ノード（1,024 コア）における大規模問題の実行時間を図 9 載せる。

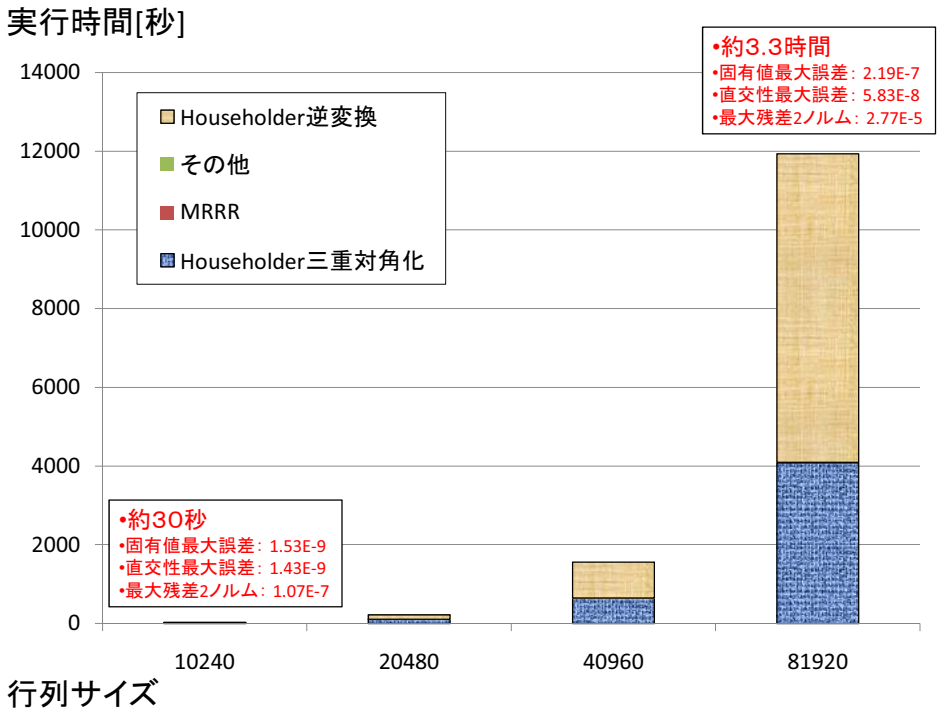

図 9　大規模問題実行時間（64 ノード(1,024 コア)、Frank 行列）

図 9 から、T2K スパコン 64 ノードを用いる場合、10,240 次元の全固有値・全固有ベクトル求解時間が約 30 秒である。経験上、このサイズの問題の求解時間としては、きわめて高速であるといえる。一方、81,920 次元においては、約 3.3 時間である。前述のとおりほとんどの時間は三重対角化と逆変換の時間となるが、現在のバージョンではキャッシュブロック化の実装がされておらず単体の演算効率が悪い[6]。したがって、実装方式の改善とチューニングにより、さらに 2～3 倍の速度向上が望める。

（6）台数効果（weak scaling）

10 万コアものペタスケール計算機環境での並列実行を想定する場合、台数効果が極めて重要となる。台数効果の中でも、大規模計算を前提とした指標である weak scaling が達成できるかどうかが第一段階として重要である。weak scaling とは、コア（もしくはノード）当たりの行列データ量を固定した場合において、FLOPS 値が線形向上していくかを評価する指標である。

図 10 に、三重対角化および逆変換部分の weak scaling の評価結果を載せる。

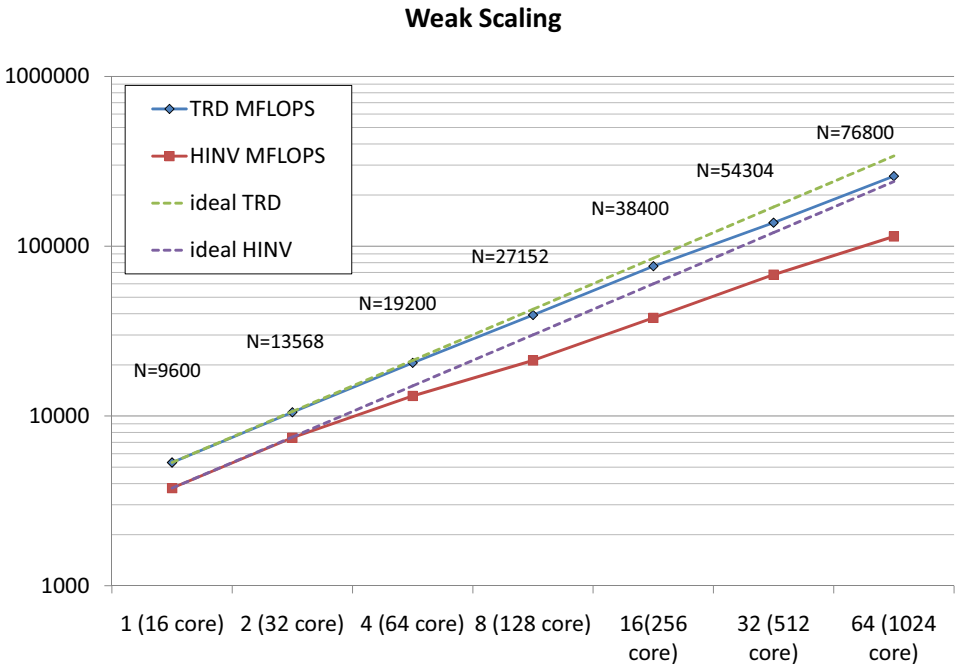

図 10　三重対角化および逆変換部分の weak scaling

図 10 では、64 ノードまでにおいては、三重対角化部分においては若干の性能劣化が観測されるものの線形向上が観測できる。性能劣化を防ぐためには、1 ノードあたりの行列サイズを大きくする以外に、通信時間を隠ぺいできるアルゴリズムの実装が考えられる。

一方、逆変換部分においても線形向上が確認できるが、8 ノードあたりで激しい性能劣化が観測された。この理由は、現在の通信実装方式の悪さによる。三重対角化部分で計算された Householder 変換ベクトルは二次元サイクリック分散で格納されており、逆変換部分で収集しつつ Householder 逆変換をしている。現在の実装では、1 ベクトル収集するごとに 1 回の逆変換を行っており、ノード数が増えると通信時間の割合が増加する実装になっている。

この性能劣化を改善する方法は、比較的簡単である。すなわち、ある数m（ブロックサイズという）を決め、m本のHouseholder変換ベクトルを収集・蓄積し、その後、m本ごとの逆変換を一度に行うように変更することで、演算時間の割合を増加させて通信時間の占める割合を減少させることができる。その結果として、線形向上を維持できるほか、演算カーネルがBLAS2 から BLAS3 相当に変更され、より効率のよい実装も可能となる。これは、単体性能（FLOPS値）の向上にもつながる。そこで、図 11 に、上記の通信ブロック化（$m=16$に固定）を行った結果を載せる[2]。

[2]加えて、Householder 変換のための行列 Q を構成する Householder 変換ベクトル収集のため、必要最低限のメッセージサイズしか収集しない改良も行った。この Householder 変換ベクトルのメッセージサイズは、2〜n まで毎回の収集ステップで変化するが、2 次元サイクリック分散となっているため、各ステップでは *(現在のメッセージサイズ)/nprocx* の切り上げ数のサイズを *nprocx* 数のコアが収集作業を行う。

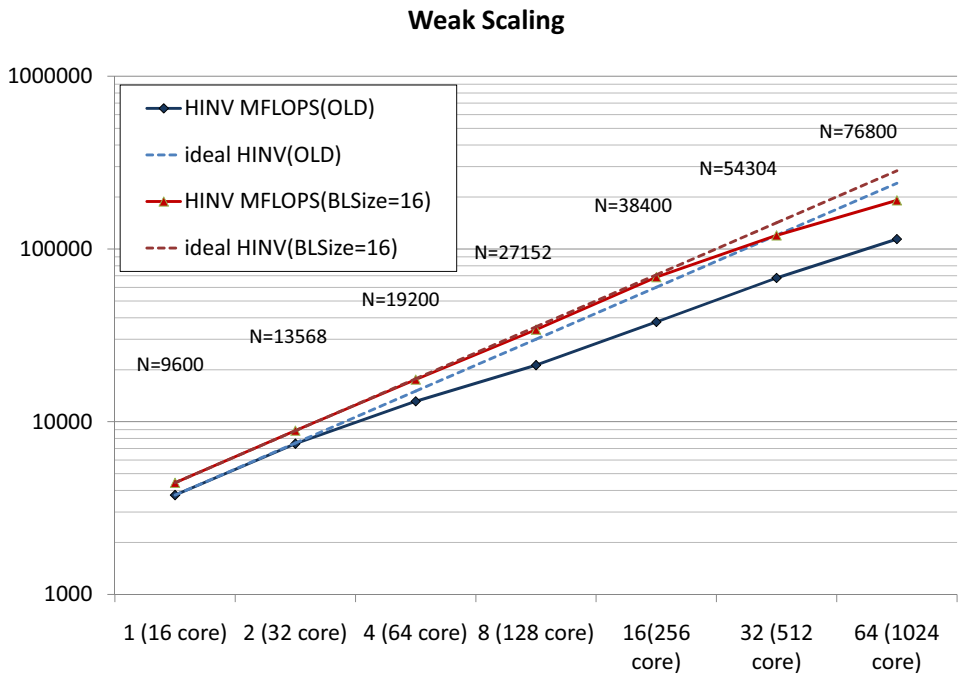

図 11 通信ブロック化された逆変換部分の weak scaling

図 11 から通信ブロックサイズ 16 にすることで、ノード数が増加した場合にも線形向上が維持できることがわかる。ただし、64 ノード利用時に若干の性能劣化が観測されたが、これは通信ブロックサイズ 16 でも不十分であることが 1 つの要因である。通信ブロックサイズをさらに大きくすることで性能劣化を回避できる可能性があるが、予測できない要因による通信部分や演算カーネル部分の性能劣化も否定できない。今後、詳細な性能解析が必要である。また予想通り、ブロック化を行うことで単体性能（FLOPS 値）の向上（1 ノードにおいて、3,757MFLOPS から 4,435MFLOPS）が観測できた。したがって、ブロック化の逆変換部分への適用は weak scaling の観点のみならず、単体性能向上にも有効であることがいえる。

３．おわりに

本稿では、10 万コアに及ぶと予想されるペタスケール計算環境に向けた実数対称密行列用の固有値ソルバを開発するため、きわめて並列性の高い固有ベクトル計算アルゴリズムである MRRR 法を、著者が開発している固有値ソルバ ABCLib_DRSSED に組み込むことで性能評価を行った。

性能評価の結果、固有ベクトルの直交性が従来法よりも 2 桁程度劣化することが許容できるのであれば、直交化を行わない逆反復法と同程度の並列実行時間になることが明らかとなった。直交化を行わない逆反復法は、理論上きわめて高い並列性があり、ペタスケール計算機環境で有効となることが期待される。一方、ペタスケール計算機環境での問題は、三重対角化と逆変換部分の性能であり、並列性能と単体性能の観点からの改良や新アルゴリズムの開発が必要となる。特に、コア数の増加に応じた繊細なチューニングを行わないと、線形向上が維持できないことから、何らかの性能チューニングの自動化がペタスケール計算環境では必須となるだろう。

現在、T2K オープンスパコンの 256 ノード（4,096 コア）実行において、行列サイズ 102,400 次元での全固有値・全固有ベクトルの計算時間は、約 2 時間 30 分である。これは、通信方式と演算カーネルのチューニングや改良を行っていない性能である。これらの改良を行うと 2〜3 倍の高速化がさらに達成できるという感触がある。

今後、さらなる超並列実行向けの実装方式、および、ペタスケール計算機環境向けの性能自動チューニング機能（Auto-tuning Facility, AT 機構）を固有値ソルバに実装していく予定である。また、本稿で言及した機能をもつ ABCLib_DRSSED のソースコードを含めた公開を行っていく予定である。

参 考 文 献

[1] Takahiro Katagiri, Kenji Kise, Hiroki Honda, and Toshitsugu Yuba: ABCLib_DRSSED: A Parallel Eigensolver with an Auto-tuning Facility, Parallel Computing, Vol.32, Issue 3, pp.231-250 (2006)

[2] Takahiro Katagiri and Yasumasa Kanada: An Efficient Implementation of Parallel Eigenvalue Computation for Massively Parallel Processing, Parallel Computing, Vol.27, No.14, pp.1831-1845 (2001)

[3] B.N.Parlett and I.S.Dhillon: Relatively Robust Representations of Symmetric Tridiagonals. Linear Algebra and Appl., 309(1-3):121-151 (2000)

[4] 山本 有作: 密行列固有値解法の最近の発展(I) : Multiple Relatively Robust Representations アルゴリズム(<特集>行列・固有値問題における線形計算アルゴリズムとその応用), 日本応用数理学会論文誌, Vol.15, No.2(20050625), pp. 181-208 (2005)

[5] ABClib_DRSSED: http://www.abc-lib.org/

[6] 片桐孝洋: T2K オープンスパコン（東大）における自動チューニング機能付き固有値ソルバの性能評価, 東京大学情報基盤センター スーパーコンピューティングニュース、Vol.10, No5., pp.52-65 (2008.9)