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

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

現在、超並列型の並列計算機が普及してきている。ハイエンドなスーパーコンピュータでは、10 万プロセッサ以上を有する。また、1 チップの上に複数のプロセッサを配置する**マルチコア型**のアーキテクチャが主流となり、各プロセッサはローカルなキャッシュと共有されたキャッシュを有する。このような、マルチコア型かつ階層化されたメモリを持つ計算機環境では、キャッシュレベルの局所性と、マルチコアレベルの並列性を同時に満たす計算アルゴリズムが性能の観点で要求される。この観点からアルゴリズムや実装方式を開発するという、新しい技術が求められている。

一方、数値計算ライブラリにおいては専門知識を必要とするアルゴリズムや実装上の性能パラメタが多数存在し、チューニング作業の人的・時間的コストが高いことが問題となっている。そこで、性能パラメタをユーザが利用する計算機環境に自動的に調整する（以降、**自動チューニング、Auto-Tuning(AT)**とよぶ）機能が研究され、一部の数値計算ライブラリの新機能として研究開発がなされてきた[1]。

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 AT 機能付き数値計算ライブラリを最新のマルチコア型かつ階層化されたメモリを持つ並列計算機で性能評価し問題点を吟味する。対象の並列計算機（マルチコア型）として、2008 年 6 月から東京大学情報基盤センターで稼働している T2K オープンスパコン（HITACHI HA8000 クラスタシステム）を利用する。

本原稿の構成は以下のとおりである。2 節で、AT 機能付き固有値ソルバ ABCLib_DRSSED[1]の概略を説明する。3 節では、T2K オープンスパコン（東大）を利用した ABCLib_DRSSED の性能評価を行う。最後に、本稿で得られた知見を述べる。

２．AT 機能付き固有値ソルバ ABCLib_DRSSED

（１）概要

ABCLib_DRSSED[1]は、対称実数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 1.04 は、MPI-1 と Fortran90 で実装されている。ライブラリが提供する最適化方針と性能パラメタの定義範囲（実装方式）内で性能が自動的に最適化される機能を有し、これを自動チューニング機能とよぶ。性能のモデリングとして、任意次数の多項式近似による実行時間予測機能、および、

サンプリング点の指定機能が実装されている。

ライブラリ上のアルゴリズムとして、対称固有値ソルバに利用される密対称行列用 Householder 三重対角化、三重対角対称行列用の固有値計算のための二分法、三重対角対称行列用の固有ベクトル計算のための逆反復法、および密対称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][1]。すなわち、１プロセッサ当たりのメモリ量を固定しプロセッサ台数が増えるごとに全体の問題サイズが大きくなる状況での台数効果である **weak scaling** の効率を求めるのではなく、全体の問題サイズを固定しプロセッサ台数を増加させる状況での台数効果である **strong scaling** において、従来方式よりも効率が良いアルゴリズムを採用している。したがって、現在主流となっている階層キャッシュを有するプロセッサからなる超並列計算機環境においては、行列サイズが大きくなるとキャッシュミスを生じて性能低下が起こる。単体性能の観点では、効率的なアルゴリズムが採用されていない。ただし台数効果では、十分な性能向上が期待できる。

（２）ベンチマークとしての特徴

ベンチマークの観点からの ABCLib_DRSSED の各処理について、以下にまとめる。

- 対称実数固有値ソルバ
  - Householder 三重対角化部（以降、**TRD**）
    - ✧ BLAS 2 演算性能評価
      - 行列-ベクトル積演算性能
      - 行列更新演算性能
    - ✧ マルチキャスト通信性能評価
      - プロセッサのグリッド（p×q）における、同時放送処理 p 個（従事プロセッサ数 q 個）の性能評価
  - Householder 逆変換部（以降、**HIT**）
    - ✧ BLAS 1 演算性能評価
      - 必要な固有ベクトル数(k)に応じて BLAS 1-k 更新となる行列更新演算性能。全固有ベクトルが必要なときは、BLAS 2 演算となる。
    - ✧ コレクティブ通信性能評価
      - 集団通信性能評価(Gather 演算)。
- QR 分解
  - 修正 Gram-Schmidt 演算部（以降、**MGSAO**）

[1] HITACHI SR2201 において、512PE 実行で 20,000 次元の三重対角化が、従来方式の ScaLAPACK の三重対角化ルーチンに比べ 2.7 倍高速である[2]。

- ✧ BLAS3 演算性能評価
  - 行列更新演算性能評価。ただし、最外ループの刻み幅はブロック幅となる。
- ✧ １対１通信性能評価
  - 通信粒度がブロック幅の逆数で変化する。つまり通信回数が、上記 BLAS3 のブロック幅の逆数となる実装における評価。

対称実数固有値ソルバではブロック化が実装されていないので、階層メモリを有するプロセッサで、かつキャッシュミスヒット時の性能劣化が大きなプロセッサでは、大規模問題実行時に性能劣化が引き起こることが予想される。一方、QR 分解ではブロック化アルゴリズムの採用により、大規模問題実行時の性能劣化を食い止めることが期待できる。ブロッキングある／なしでの性能差も、プロセッサ性能の評価指標の１つである。

（３）性能パラメタの説明

AT 性能パラメタの観点では、ABCLib_DRSSED の性能パラメタは以下に分類される[1]。

- 対称実数固有値ソルバ　性能パラメタ
  - Householder 三重対角化用
    - ✧ 通信実装方式切り替え *ictr*
    - ✧ 行列-ベクトル積アンローリング段数制御 *imv*
    - ✧ 行列更新演算アンローリング段数制御 *iud*
  - Householder 逆変換用
    - ✧ 通信実装方式切り替え *ichit*
    - ✧ 行列更新演算アンローリング段数制御 *ihit*
- QR 分解　性能パラメタ
  - ブロック幅調整 *ibl*
  - 枢軸ブロック演算用
    - ✧ 最外ループアンローリング段数制御 *iop*
    - ✧ 第２ループアンローリング段数制御 *isp*
  - それ以外の演算用
    - ✧ 最外ループアンローリング段数制御 *ioo*
    - ✧ 第２ループアンローリング段数制御 *iso*

（４）性能パラメタの定義域：現在の実装における制約

ABCLib_DRSSED 1.04 では、自動チューニングパラメタについて、以下の実装がなされている。

- 対称実数固有値ソルバ　性能パラメタ定義域
  - Householder 三重対角化用
    - ✧ 通信実装方式切り替え

      ictr = {1 対 1 通信関数を用いた実装, 集団通信関数を用いた実装}
    - ✧ 行列-ベクトル積アンローリング段数制御

imv = {1段,2段,…,16段}：最外ループのみ

- ✧ 行列更新演算アンローリング段数制御

  iud = {1段,2段,…,16段}：最外ループのみ

- ➢ Householder 逆変換用
  - ✧ 通信実装方式切り替え

    ichit = {放送関数を用いた実装、1対1ブロッキング通信関数を用いた実装、1対1ノンブロッキング関数を用いた実装}
  - ✧ 行列更新演算アンローリング段数制御

    ihit = {1段,2段,…,16段}：最外ループのみ

- ● QR 分解　性能パラメタ定義域
  - ➢ ブロック幅調整

    ibl = {1,2,3,4,8,16}
  - ➢ 枢軸ブロック演算用
    - ✧ 最外ループアンローリング段数制御

      iop = {1段,2段,3段,4段}
    - ✧ 第２ループアンローリング段数制御

      isp = {1段,2段,3段,4段,8段,16段}
  - ➢ それ以外の演算用
    - ✧ 最外ループアンローリング段数制御

      ioo = {1段,2段,3段,4段}
    - ✧ 第２ループアンローリング段数制御

      iso = {1段,2段,3段,4段,8段,16段}

３．性能評価

マルチコア型の計算機では、MPI とスレッドによるハイブリッド MPI（ハイブリッドプログラミング）の有効性が指摘されている[3]。そこで、本性能評価では、ハイブリッド MPI の性能評価をすることも目的とする。

（１）評価環境

T2K オープンスパコン（東大）(HITACHI HA8000 クラスタシステム）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、メモリは 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 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。

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）では、-Oss -noparallel、**ハイブリッド MPI**（MPI と自動並列化によるスレッド並列化の混合実行）では -Oss -parallel である。実験期間は、2008 年 7 月 1 日～7 月 9 日である。

以降の性能評価では、各問題サイズにおいて性能パラメタを全て自動チューニングして最高速となるパラメタが自動設定されている。なお、ABCLib_DRSSED 1.04 におけるデフォルトパラメタは、以下のとおりである。

- *ictr = {1 対 1 通信関数を用いた実装}*
- *imv = {8 段}*
- *iud = {6 段}*
- *ichit = {放送関数を用いた実装}*
- *ihit = {1 段}*
- *ibl = {4}*
- *iop = {4 段}; isp = {8 段};*
- *ioo = {4 段}; iso = {8 段};*

以上のデフォルトパラメタに対する AT による速度向上について評価する。アンローリング段数を 1 段に固定し、コンパイラ最適化による効果のみと比べたものではない[2]。このようなコンパイラ最適化のみより、本デフォルトパラメタの方が経験的に高速となる。

（2）numactl の効果

Linux では、プロセスを任意の CPU とメモリに割り付ける numactl というコマンドが提供されている。T2K オープンスパコン（東大）でも numactl が利用できる。numactl を利用しないと MPI プロセスにおける物理 CPU と物理メモリの割り当ては OS により決定される。この場合、最適なメモリ配置にならないばかりか、場合により 1CPU に 2 つ以上のプロセスやスレッドが割り当てられ、実行毎に実行時間が異なる不安定な状況を生じる。性能低下が急に起こったり、正しくチューニング作業ができないなどの悪影響が生じる。そこで、numactl を利用する/しない場合に、どのように自動チューニングの効果に影響するか評価した結果を図 1 に示す。

図 1 から、TRD では行列サイズが大きい時に、HIT、MGSAO では行列サイズが小さい時に、numactl を利用すると 1.3～1.5 倍の性能向上の効果があることがわかる。なお、この実行時間は性能パラメタ空間すべての実行により最適化されたパラメタを用いた実行時間であり、あるパラメタによる 1 回限りの実行時間を示しているのではない。

[2] これをベースラインとして AT の効果を評価する論文も多い。このベースラインを用いた評価に比べ、本性能評価の基準では経験的に AT の効果が低めに算出される。

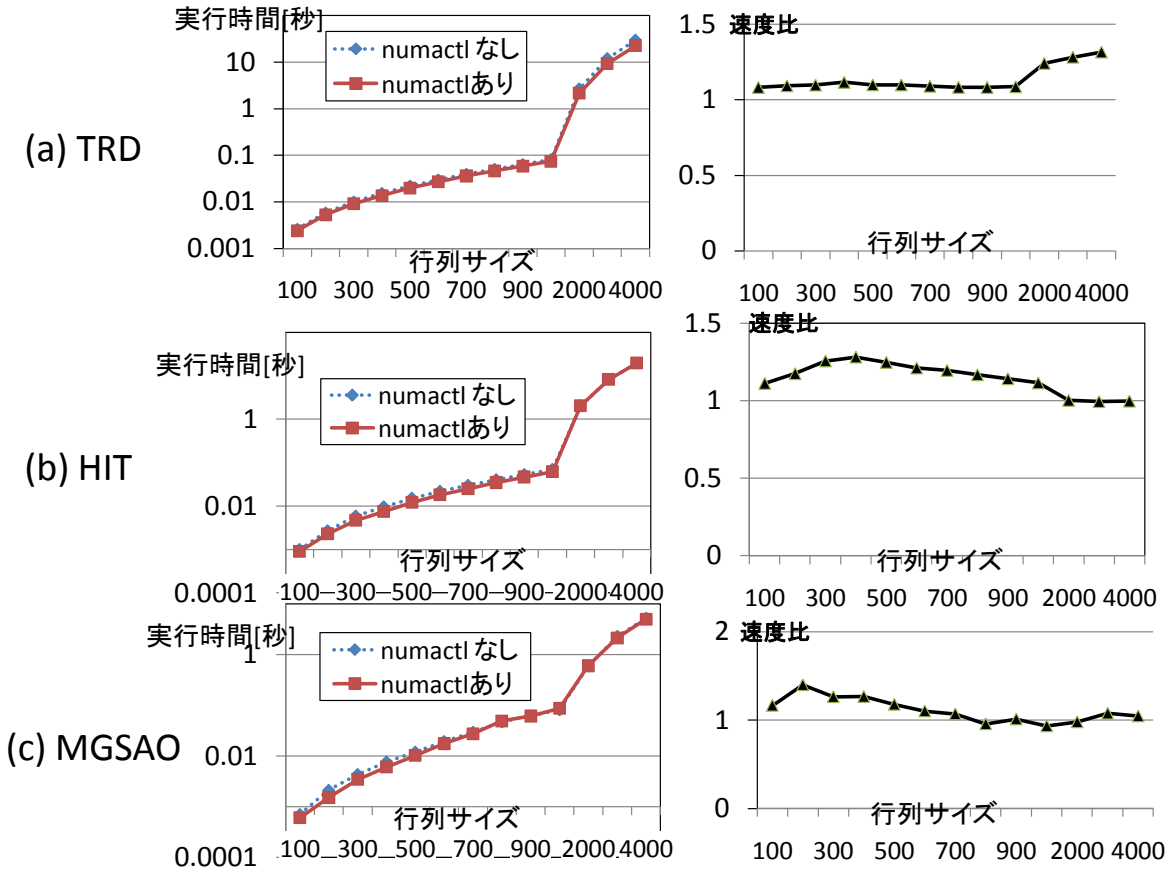

図 1　AT における numactl の効果（1 ノード、ピュアＭＰＩ（16 プロセス））

（3）単体性能

図 1 の numactl ありの時間から、各処理の GFLOPS 値が計算できる。図 2 にそれを示す。

図 2 では、TRD と HIT において行列サイズ 2,000 を超えてから演算性能が 1/4～1/5 に低下する。一方、MGSAO では、このような性能低下がない。MGSAO はブロック化アルゴリズムを採用していることから、ブロック化の効果がきわめて大きいことを意味している。また、行列サイズ 1000 の時のワーキングスペースは、TRD と HIT では概算で 4*N（ここで、N は行列サイズ）となるため 4*1000*8 ≒ 32K である。したがって L1 キャッシュにデータが乗る上限となることが予想されることから、結果は妥当である。

一方、ピーク性能（147.2 GFLOPS）に対する効率は 12%～17%であり、極めて十分でない。この理由は ABCLib_DRSSED 1.04 の実装においては、（1）TRD と HIT はブロック化が採用されていないこと；および、（2）MGSAO ではブロック化が採用されているがループアンローリングやタイリングなどの自動チューニング項目が最内側ループに対して行われていない；という、AT ライブラリ自体の最適化戦略から生じたものと推測される。ABCLib_DRSSED 開発時に想定した計算機では、最内ループ長が長くとも性能劣化が起こりにくいアーキテクチャを想定していたためであり、このことから T2K オープンスパコンでは抜本的な最適化戦略の再構築が必要である。

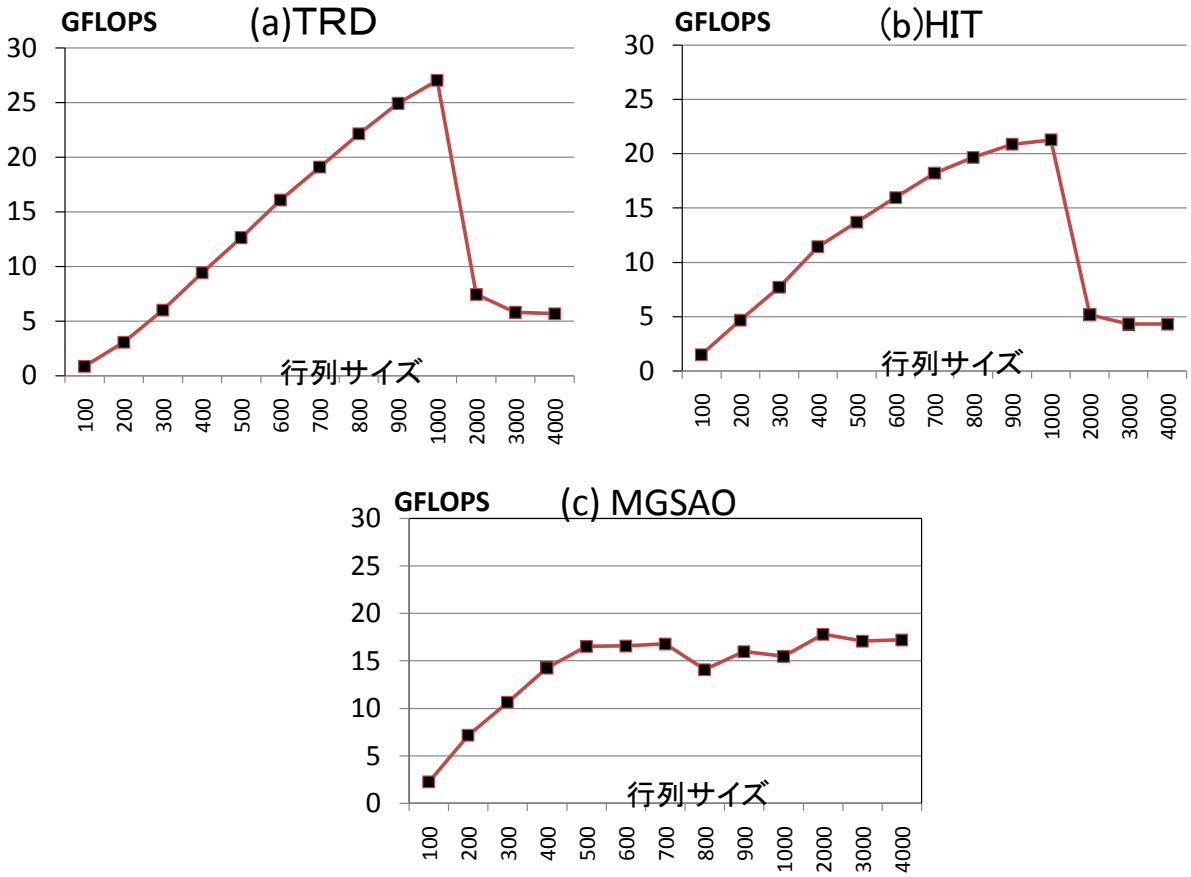

図 2　各処理の GFLOPS 値（１ノード、ピュアＭＰＩ）

（４）ハイブリッド MPI 実行の効果

次に、MPI のみによる実行（ピュア MPI）と、MPI とスレッドの混合環境による実行（ハイブリッド MPI）の性能を比較する。図 3～図 5 に、各処理における行列サイズ 4000 の時の、ピュア MPI による実行とハイブリッド MPI による実行の実行時間を載せる。ここで図中の表記"pure"とはピュア MPI（16 プロセス）、"p4＊th4"はハイブリッド MPI（4 プロセス、4 スレッド）、"p1＊th16"はハイブリッド MPI（1 プロセス、16 スレッド）である。これらはいずれも、ノード内における実行形態であり、ノード間は MPI による実行となっている。

図 3～図 5 では、１ノードのときにはピュア MPI による実行が高速であるが、４ノード、もしくは８ノードを超えたあたりから逆転が生じ、最も高速となるのは４プロセス、４スレッドのハイブリッド MPI であることがわかる。したがって、ノード数が増加するにつれ、ハイブリッド MPI が高速となる場合があることが判明した。

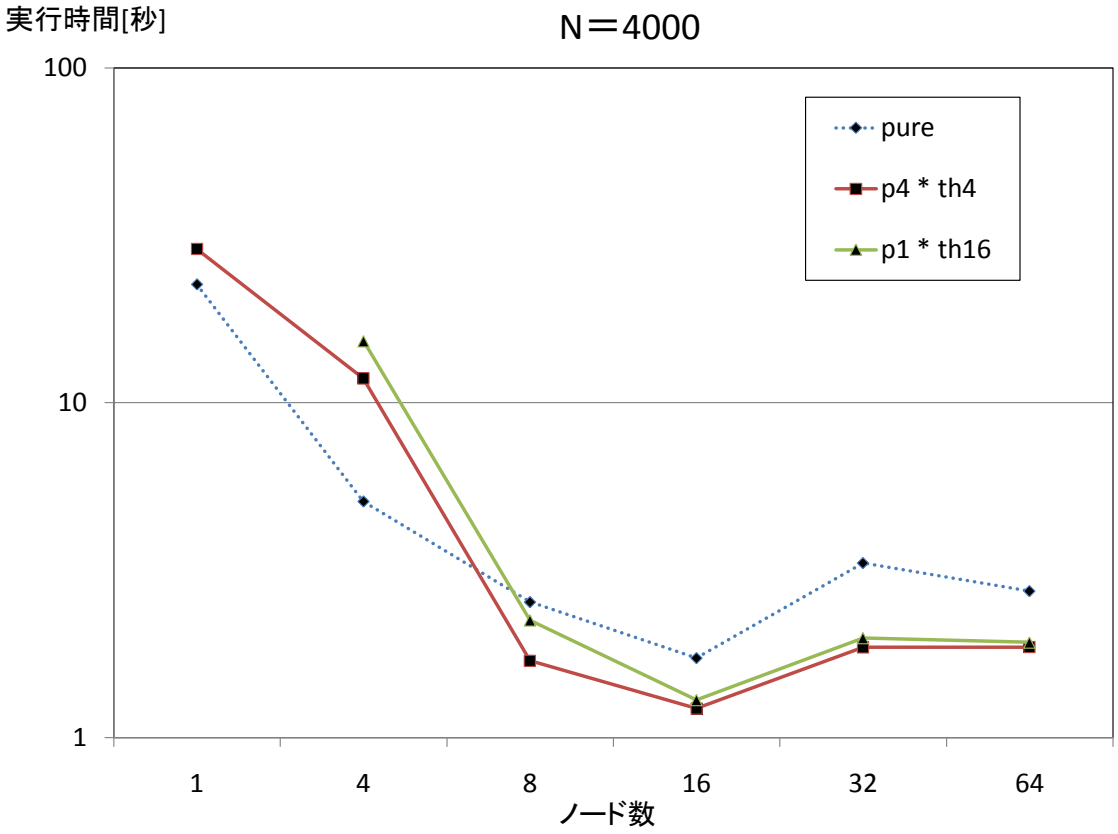

図 3　ハイブリッドMPIの効果（ＴＲＤ）

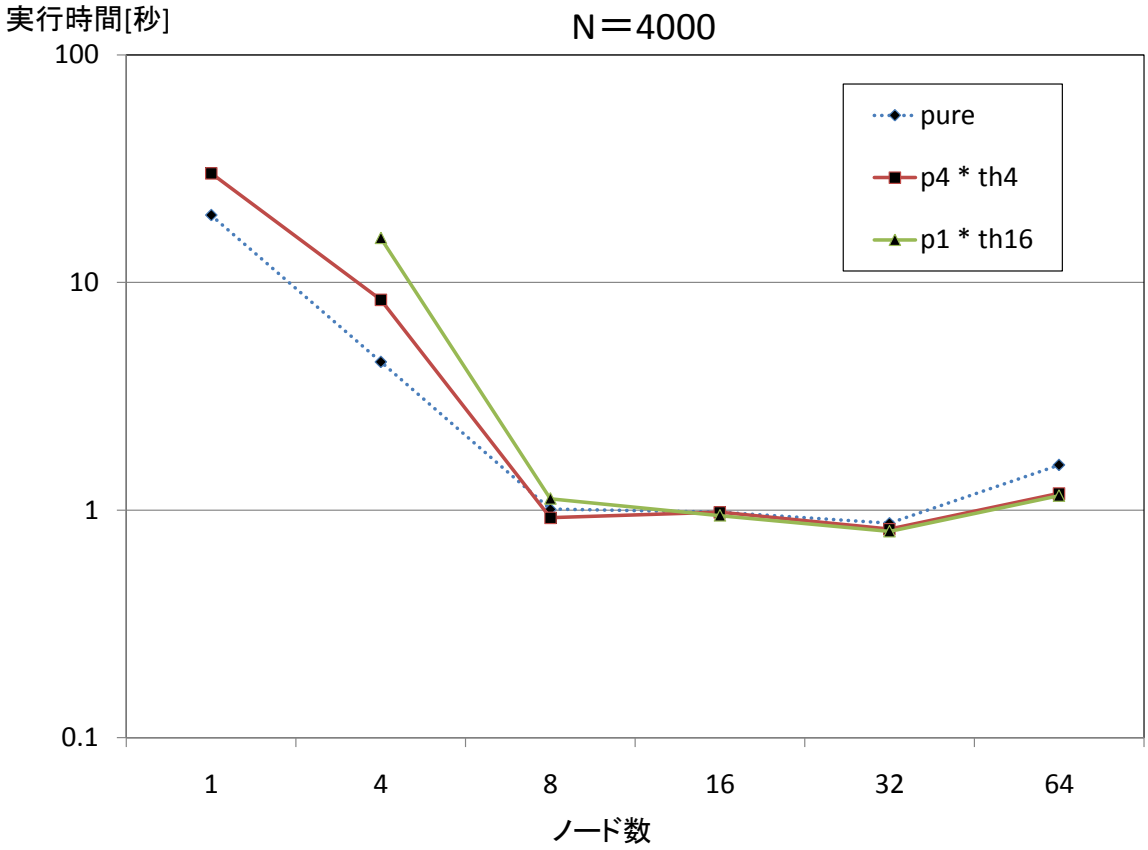

図 4　ハイブリッドMPIの効果（ＨＩＴ）

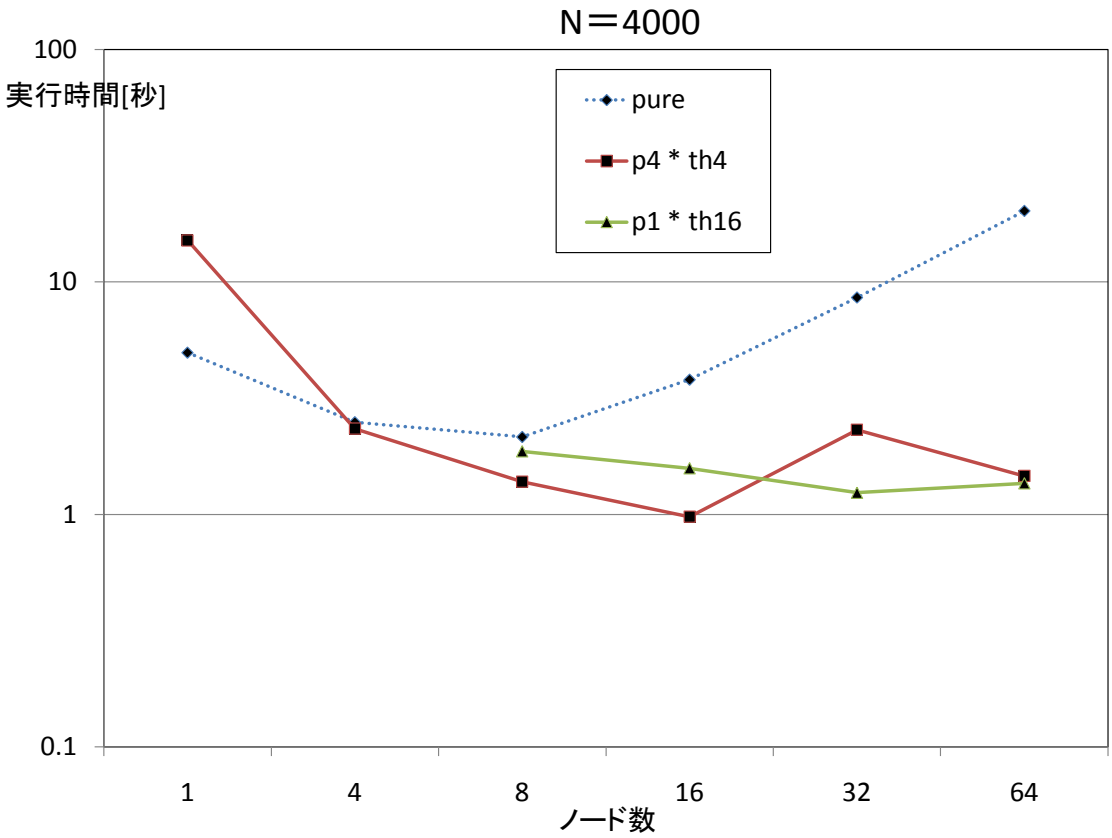

図 5　ハイブリッド MPI の効果（ＭＧＳＡＯ）

一方、台数効果をみるため、図 6 に TRD の場合の台数効果（ピュア MPI（16 プロセス）、ハイブリッド MPI（4 プロセス、4 スレッド））の１ノード実行を基準とする台数効果を乗せる。

図 6 から、４ノード～１６ノードにおいて、理想的な台数効果を超えるスーパーリニアスピードアップが観測された。この理由は、図 2 からわかるように、４ノードを超えると行列サイズ 4,000 では L1 キャッシュに乗るサイズになることから、キャッシュの効果だと推定される。また、ピュア MPI よりハイブリッド MPI の方が台数効果が高い。これは、ccNUMA アーキテクチャを考慮したメモリ配置によりメモリアクセス時間が短縮されることと、MPI プロセス数がハイブリッド MPI によるものはピュア MPI より 1/4 になることから通信性能の劣化が少ないことが原因と推察される。いずれにせよ、ハイブリッド MPI の長所が現れたものと思われる。

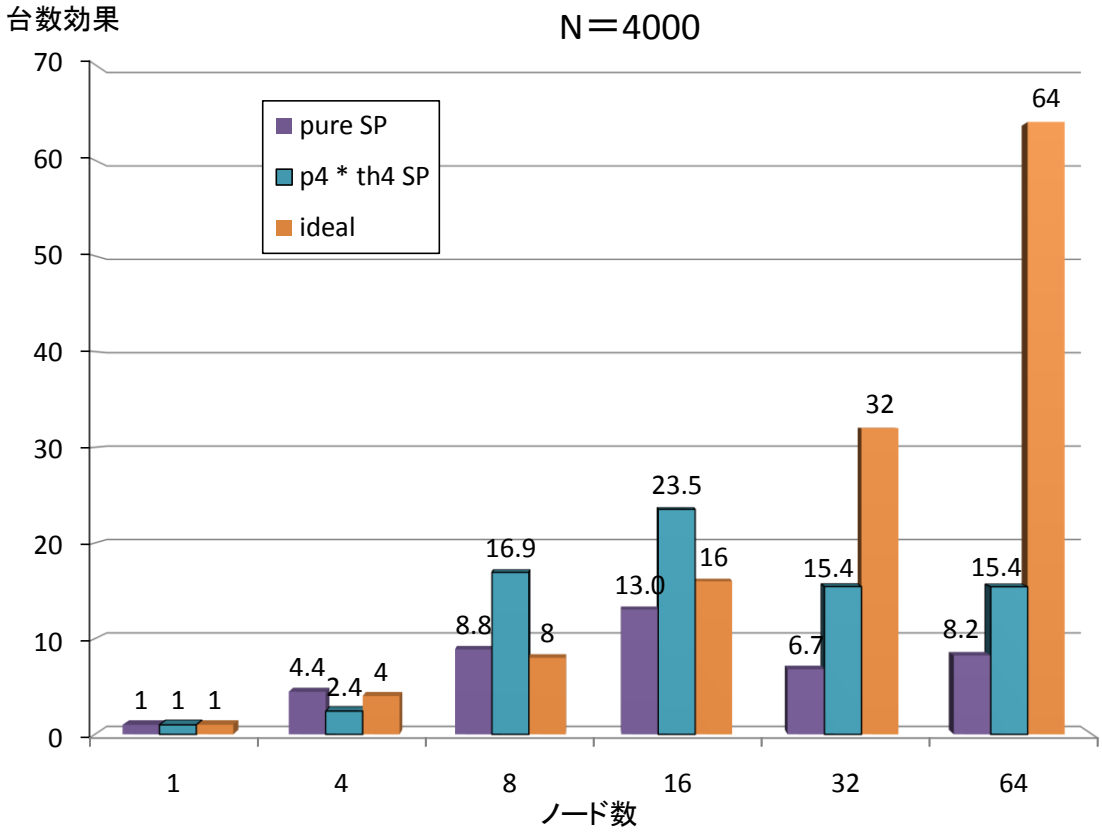

図 6　台数効果（ＴＲＤ）

（５）自動チューニングの効果

図 7、図 8 に AT による速度向上を示す。

図 7、図 8 から、TRD、MGSAO では１ノードより６４ノードのほうが概ね AT の効果が高い。この理由は、それぞれ集団通信演算実装選択と通信粒度調整が AT 機能として入っているので、通信最適化の効果がこれらのアルゴリズムでは出やすいからと推察される。一方 HIT では 64 ノード時に１ノードよりも AT 効果が低い。これは、HIT では頻繁に Gather 処理をする通信が必要であるが、この処理を最適化するだけの実装選択のバリエーションが少なかったことによると思われる。なお、オリジナルの ABCLib_DRSSED 1.04 では HIT においてノンブロッキング通信による実装選択機能が提供されているが、本環境では MPI バッファのオーバーフローとなり実行できなかったことから、この通信実装は OFF にしてある。また、全般的にハイブリッド MPI は AT 効果が高い。この理由は、デフォルトパラメタのアンローリング段数では４スレッド実行で性能を出すだけの並列性がなく、AT によりアンローリング段数を大きくすることで並列性を向上できた（高速化できた）ことが理由と考えられる。

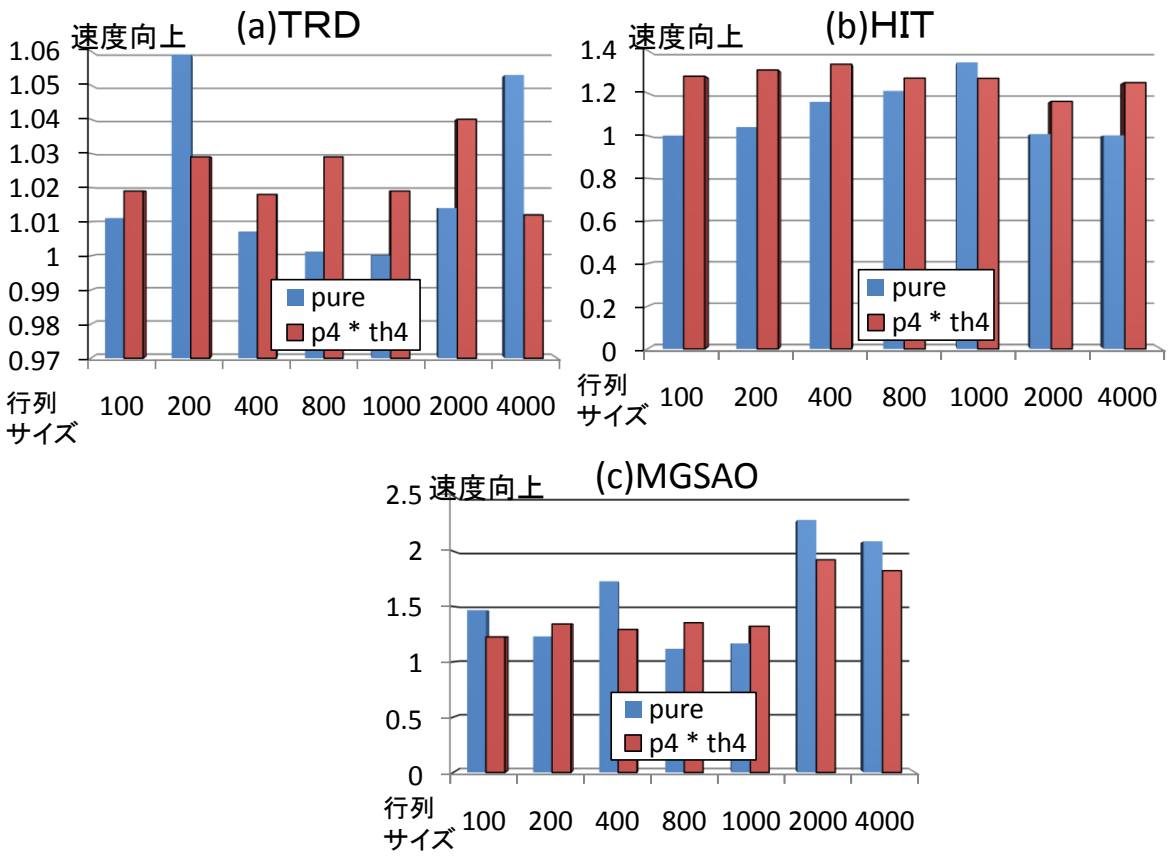

図 7　ＡＴの効果（１ノード）

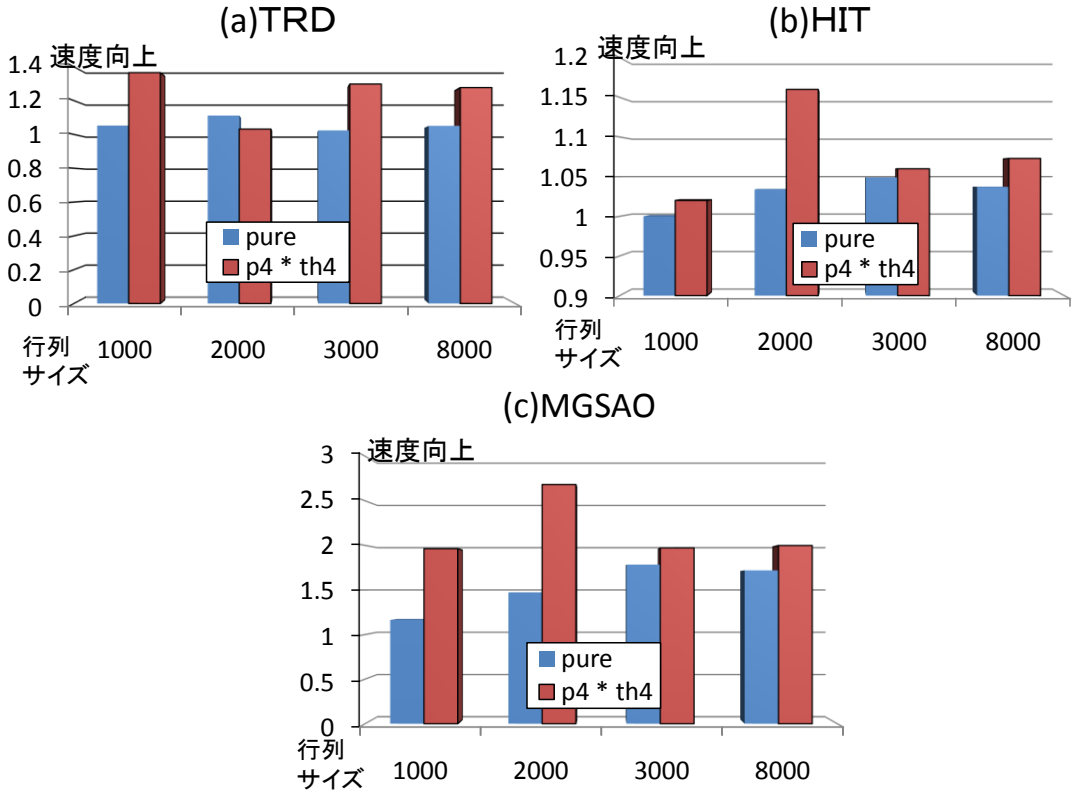

図 8　ＡＴの効果（６４ノード）

一方、ブロック化アルゴリズムの効果をみるため、MGSAO において AT された最適なブロック幅を調べた。これを、図 9 に示す。

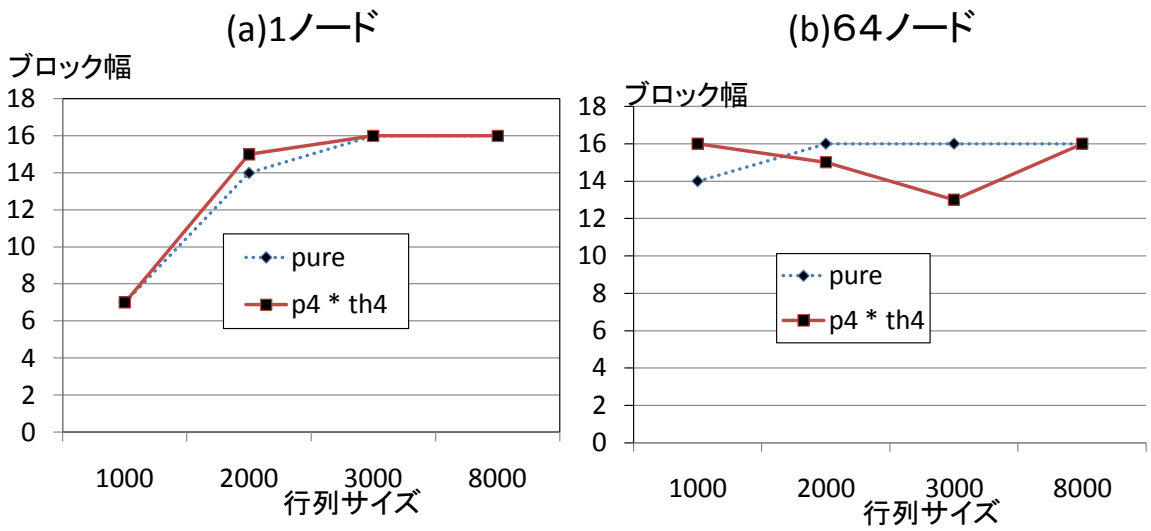

図 9　最適ブロック幅（MGSAO）

図 9 から、1 ノードにおいては行列サイズが 1,000 と小さいときブロック幅 7 と小さい値をとる。行列サイズが 3,000 以上となるとき、ブロック幅 16 が設定される。また１６ノードでは、ブロック幅が 13～16 と大きな値が設定される。この理由は、ノード数が大きくなると通信粒度を粗くしないと通信待ち時間が生じ性能が劣化することによる。一方、１ノードでは通信性能が６４ノードの場合に比べて高性能となるが、問題サイズが小さいときブロック幅が大きいと負荷バランスが悪くなり性能が劣化するので、ブロック幅が小さめの値に設定されたもの推察される。いずれにせよ、ブロック幅調整の AT 機能がうまく働いていると結論づけられる。

ブロック化に伴う演算カーネル実装の変化について、最適な MGSAO のループアンローリング長を見ることで推察してみる。MGSAO では、BLAS3 演算部分に最外ループと第２ループについてのアンローリング段数を AT している。そこで、（最外ループアンローリング段数）×（第２ループアンローリング段数）を実装指標とした場合の評価結果を図 10 に示す。

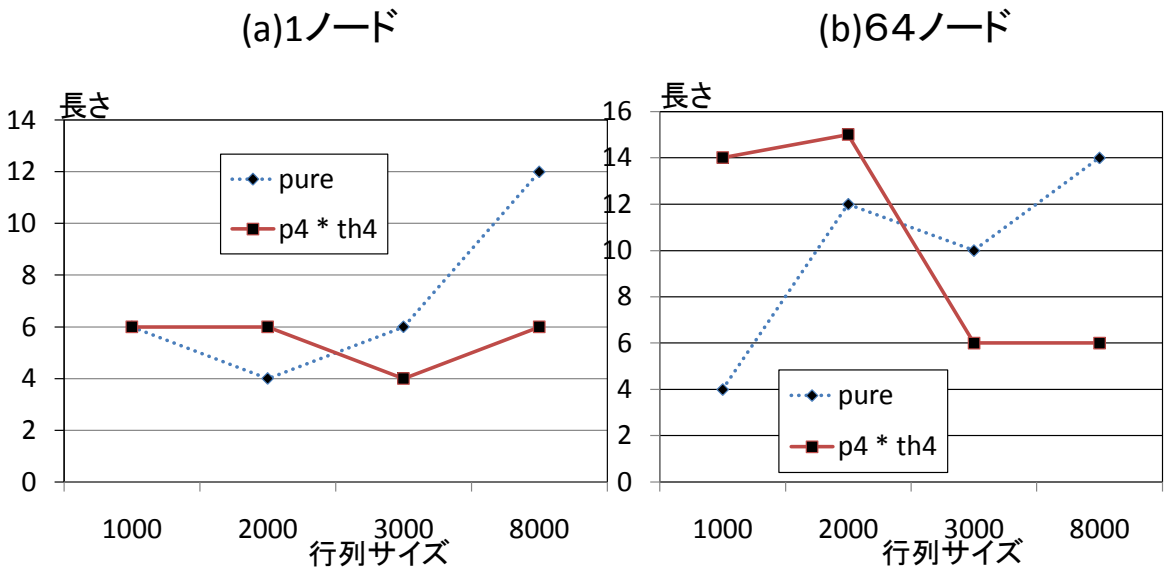

図 10　最適ループ長の指標
（MGSAO、最外ループアンローリング段数×第２ループアンローリング段数）

図 10 から面白いことに、ピュア MPI では行列サイズが小さいとき指標が小さく、行列サイズが大きくなるにつれ指標も大きくなる傾向がある。一方、ハイブリッド MPI ではピュア MPI と逆の傾向が伺われる。この理由は、ピュア MPI では行列サイズを大きくすると各コアの演算器を効率的に動かすために多数の命令レベルの並列性（プログラム上に"陽"に現れるもの）が必要となることから指標が大きくする必要があること、一方で、ハイブリッド MPI では行列サイズが大きくなると各スレッドからのデータ読み書きによるメモリ衝突がオーバーヘッドとなるため命令レベル並列性（プログラム上に"陽"に現れるもの）を抑える必要があることが理由として考えられる。

いずれにせよここでの主張は、

「ピュア MPI とハイブリッド MPI では、高性能を達成するために、
行列サイズに影響される全く異なった実装が必要となる」

ということである。このことは高性能を達成するために、ユーザに従来よりもさらなる負担（ピュア MPI とハイブリッド MPI の 2 種それぞれについて、行列サイズを考慮した最適実装）が必要となることを意味している。

今後、コンパイラ自動最適化やライブラリ上の AT 機能など、何らかの自動性能チューニング技術の強化が、マルチコア型の並列計算機環境では必須となるに違いない。

３．おわりに

T2K オープンスパコン（東大）の 64 ノードにおける三重対角化処理は、行列サイズ 10,000 のとき約 4.7 秒、行列サイズ 80,000 においても約 50 分である。これは、ブロック化など最適化された性能ではないものの、行列サイズ 10,000 での実行に限定すると経験的に十分高性能である。

一方、約 8 年前、HITACHI SR2201 の 512PE 実行（月 1 回サービスにおける利用）において、行列サイズ 10,000 の三重対角化が約 80 秒であったこと[2]を考慮すると、約 16 倍の高速化が達成されている。SR2201 のノード理論最大性能は 300MFLOPS で 512 ノード（512 PE）での理論最大性能 153.6GFLOPS、HA8000 でのノード性能は 147.2GFLOPS で 64 ノード(1024 Core)での理論最大性能は 9420.8GFLOPS を考慮すると、HA8000 では理論性能では約 60 倍になっているものの、実行性能ではそれに及ばない[3]。これは、十分なキャッシュ最適化が施されていないということがあるものの、CPU における演算速度向上に対するメモリアクセス性能の向上が不十分であることを意味している。このハードウエア上の演算速度とメモリアクセス速度の「ギャップ」を解決するには、さらなるアルゴリズム上の工夫が必要となる。

今後の課題として、ブロック化アルゴリズムの実装や、マルチコア型のアーキテクチャを意識した新アルゴリズムの開発が必要である。また SSE 命令など、マルチメディア処理を指向して実装されたベクトル演算命令を用いてカーネルを再構築することも、実用的なソルバの開発では必要であろう。

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] 中島 研吾:階層型領域分割によるマルチステージ並列前処理手法へのハイブリッド並列プログラミングモデルの適用、情報処理学会研究報告 2007-HPC-110、Vol.2007, No.59(20070608) pp. 25-30 (2007)

[3] 当然、行列サイズ 10,000 程度のきわめて小さい問題ではアムダールの法則によりスケールしないことを考慮しなくてはならない。この観点から、先述のとおり HA8000 で約 5 秒での三重対角化は十分に高速であるといえる。