FX10 4800ノードを用いた
通信削減型QR分解アルゴリズムの性能評価

深谷 猛

理化学研究所 計算科学研究機構, JST CREST

１．はじめに

京コンピュータの一般利用開始から二年弱が経過し，数百から数千個のCPUを用いた並列計算が日常的に行われる時代となった。そのため，このような規模の並列計算を支援するための基盤ソフトウェアが強く求められており，その開発が進められている。科学技術計算の様々な場面で幅広く使用されている行列計算ライブラリに関する例を挙げると，これまで標準的に利用されてきたScaLAPACK[1]は，現在の計算機アーキテクチャへの対応が大幅に遅れており，上述（あるいはそれ以上）の規模の並列計算において高い性能が期待できないことが指摘されている[2,3]。そのため，最新の計算機の特性を考慮して，アルゴリズムレベルから再開発を行う必要が生じている。

本稿では，基本的な密行列計算の一つである行列のQR分解の並列計算アルゴリズムを扱う。QR分解は，疎行列に関する線形方程式や固有値問題に対する部分空間射影型解法のブロック版における直交基底の計算や長方行列の特異値分解計算の前処理など，他の行列計算の内部で幅広く必要となる計算である。さらに，RSDFTといったシミュレーションの内部で直接使用される応用例[4,5]もある。いずれのQR分解の応用例も，冒頭で述べたような規模での並列計算が行われているため，同様の規模でQR分解の並列計算を効率的に行うアルゴリズム（やプログラム）が強く求められている。

これまでQR分解の数値計算では，計算精度面の理由から，ハウスホルダー変換に基づくQR分解のアルゴリズム（ハウスホルダーQR）がよく用いられてきた。しかし，このアルゴリズムは，大規模な並列計算において通信時間が大きなボトルネックとなる，という問題点を抱えている。それに対して，近年，通信回数削減（Communication-Avoiding）という観点から，CAQR（やTSQR）といった新しいアルゴリズムが提案され[6,7]，その性能が注目されている。そこで，今回，FX10の4800ノードを用いて，ハウスホルダーQRとCAQRの並列性能を測定し，両者の比較を行った。その結果，並列計算におけるCAQRの性能の優位性とアルゴリズム中のパラメータの最適化の重要性を具体的に確認することができた。

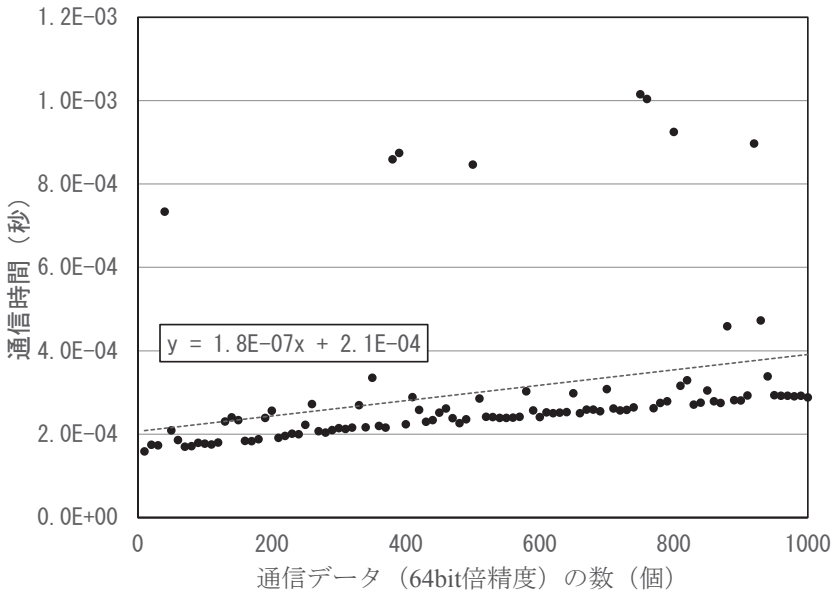
２．通信回数削減（Communication-Avoiding）の重要性

分散メモリ型の並列計算機における典型的な行列計算プログラムの実行時間の内訳は，浮動小数点演算のための時間（演算時間）と分散しているCPU間のネットワークを介した通信（同期もこれに含まれる）のための時間（通信時間）の二種類である。なお，実際のプログラムでは演算と通信をオーバーラップすることが多々あるが，本稿ではオーバーラップはないものとして話を進める。一般的に，CPU 数を増やすと演算時間は減少し，通信時間は増加するため，並列化の効果（高速化率）を向上させるためには通信時間を削減することが重要である。

通信時間の内訳をもう少し詳しく述べると，通信の立ち上がりの時間（通信するデータ量に関わらず一定）と通信するデータ量に比例して増加する時間に区別することができる。また，このことを踏まえて，通信時間を$T_{\mathrm{comm}}(w) = \alpha + \beta w$とモデル化することがよくある。ここで，$w$は通信するデータサイズ，$\alpha$は通信のセットアップコスト（あるいはレイテンシ），βはネットワークバンド幅の逆数である。図１は FX10 でプロセス数を 4800 とした場合（1 ノードに 1 プロセス）における MPI の集団通信（MPI_Allreduce）の通信時間と通信データサイズの関係を示したグラフであるが，上記の一次関数モデルに比較的よく一致していることが確認できる。

図１では，最小二乗近似で得られる直線（MS Excel の機能を利用）も併せて示している。この結果より，この場合（プロセス数が 4800）では，モデル式中のパラメータは$\alpha \simeq 10^{-4}$(秒)，$\beta \simeq 10^{-7}$(秒/個)であることが確認できる。これは，通信一回当たりのセットアップコストが，通信するデータ一個当たりのコストよりもはるかに大きいことを示している。今回の例では，通信を一回増やす際のセットアップコストと通信データ量を千個増やすコストが同程度ということになる。

従来の行列計算プログラムでは，一度に多くのデータ（例えば，部分行列全体）を通信するよりも，スカラーやベクトルといった少ないデータを逐次的に通信することが一般的である。そのため，上述の事実を踏まえると，従来の行列計算プログラムの並列実行時の通信時間の大部分を通信のセットアップコストが占めていることが容易に推測できる。つまり，通信回数を削減することで，通信時間の大幅な削減が期待できる。加えて，今後の技術の進歩において，レイテンシの短縮率よりもネットワークバンド幅の拡大率の方が相対的に高いことが指摘されており，さらに両者の差が拡大することが予想されている。このような状況から，近年，通信回数を削減した（Communication-Avoiding）アルゴリズムに関する研究・開発が活発に行われている[8]。



第１図：FX10 における MPI_Allreduce の実行時間（4800 プロセス，1 プロセス/1 ノード)。

３．ハウスホルダーQR アルゴリズム

本稿では，$m \times n\ (m \geq n)$の実密行列Aの QR 分解を考える。代表的な QR 分解の数値計算アルゴリズムとして，グラム・シュミットの直交化とハウスホルダーQR 分解があるが，本稿では，対象とする行列の条件数に関わらず計算が安定であるために，数値計算でよく用いられる後者のアルゴリズムを扱う。

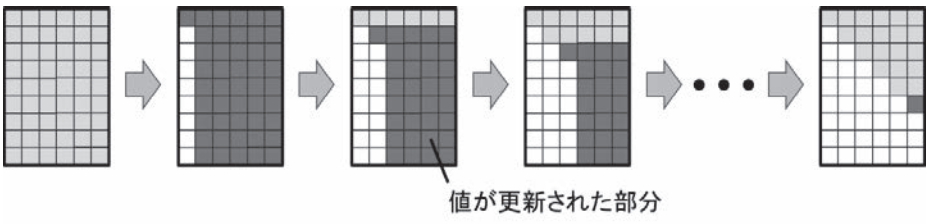
ハウスホルダーQR[9]は，ハウスホルダー変換と呼ばれるm次元の直交変換H_iを用いて，

$$H_n \cdots H_2 H_1 A = \begin{pmatrix} R \\ O \end{pmatrix}$$

と，与えられた行列を一列ごとに上三角化することで QR 分解を計算するアルゴリズムである（図２）。ここで，Rは$n \times n$の上三角行列である。また，ハウスホルダー変換の具体的な形は

$$H_i = I_m - t_i \boldsymbol{u}_i \boldsymbol{u}_i^{\mathsf{T}}, \qquad t_i = \frac{2}{\boldsymbol{u}_i^{\mathsf{T}} \boldsymbol{u}_i}$$

で与えられ，$H_i = H_i^{\mathsf{T}} = H_i^{-1}$を満たす。

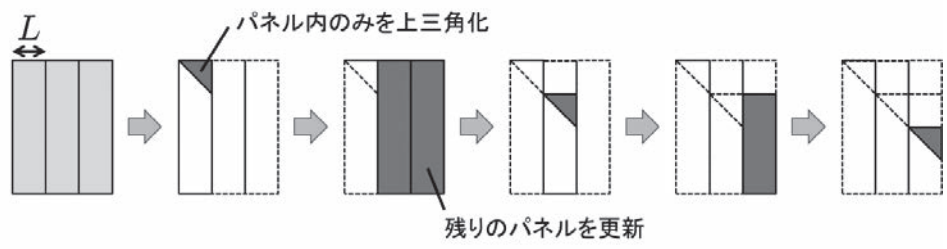


第２：ハウスホルダーQR 分解のアルゴリズムの概要。

実際の計算では，ハウスホルダーQR に対して，パネルブロッキングと呼ばれる手法[10]を用いることが一般的となっている。これは，図３のように行列を列方向にパネルに分割し，パネルごとに計算を進める手法である。この手法では，先にパネル内のみで計算を行い，そこで用いたパネルサイズ（以下L）分のハウスホルダー変換を

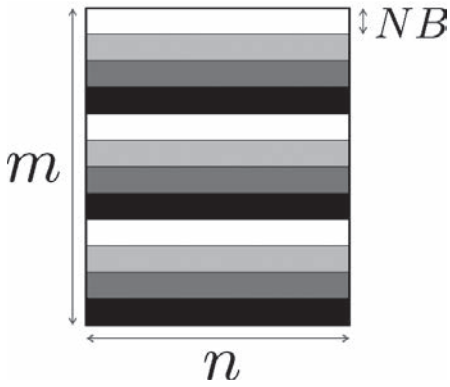
$$H_L \cdots H_2 H_1 = I_m - Y T Y^{\mathsf{T}}$$

と行列の形にまとめてから[11]，残りのパネルを実行効率の高い行列積（Level-3 BLAS）を使用して更新する[10]。なお，Yはハウスホルダー変換のベクトル$\boldsymbol{u}_i$を並べた$m \times L$の行列で，Tは$\boldsymbol{u}_i$から計算される上三角行列である。そのため，パネルブロッキングを使用した場合，使用しない場合と比べて，Tの計算のための演算と通信が余分に必要となる。適切にパネルサイズLを選択して，後者の影響を抑えながら，前者の効果を大きくすることで，計算全体の高速化が期待できる。



第３図：パネルブロッキングを用いたハウスホルダーQR の様子。

最後に，分散並列環境向けのハウスホルダーQR の実装に関して簡単に述べる。なお，本稿では，行列データを各プロセッサが，行方向のブロックサイクリック分散で保持している場合を想定する（図４）。詳細は[9]などに委ねるが，ハウスホルダーQR において，ハウスホルダー変換を生成するために，列ベクトルのノルムの計算が必要となる。また，ハウスホルダー変換を残りの行列（もしくはパネル内の残り部分）に作用させて更新を行う際に行列ベクトル積が生じる。これらの計算において，前述の分散のため，全てのプロセス間での集団通信（具体的には MPI_Allreduce）が必要となる。パネルブロッキングの有無に関わらず，ハウスホルダーQR では左側の列ベクトルから順番に上三角化が行われるので，そのため，アルゴリズム全体では $O(n)$回の集団通信が必要となる。これが，並列計算において大きなボトルネックとなる。



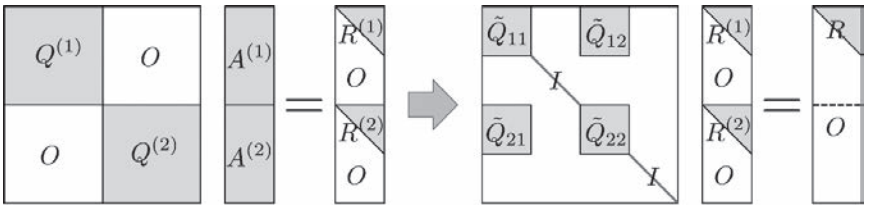
第４図：行方向のブロックサイクリック分散の様子（4 並列の場合）。

４．通信回数削減型アルゴリズム：TSQR と CAQR

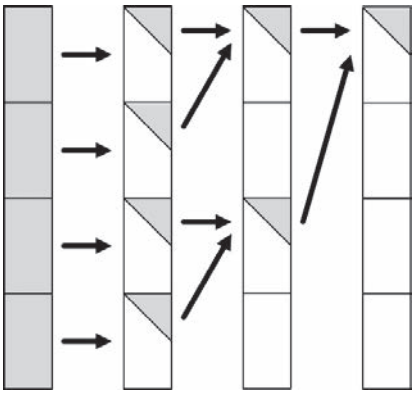
４．１．TSQR アルゴリズム

最初に，縦長（$m \geq nP$）の行列の QR 分解を計算する通信回数削減型アルゴリズムである TSQR アルゴリズム[6,7]を紹介する。このアルゴリズムも，ハウスホルダーQR と同様に，直交変換を用いて対象の行列を上三角化することで QR 分解を計算するものであるが，その特徴は，図５に示すような構造を持った直交変換を用いて，行列を部分ごとに上三角化し，最終的に一つの上三角行列に変換する点である（図６）。なお，部分的な上三角化には，サイズの小さいハウスホルダー変換を利用する。また，このような直交変換を用いても，数値計算における精度はハウスホルダーQR と同等で非常に良いことも示されている[12]。

このアルゴリズムを用いることで，並列計算における並列粒度が大きくなり，通信は複数の三角行列を一つにまとめる（三角行列のリダクション）ときのみに生じる。具体例としては，二分木にしたがって，一対一通信で三角行列を通信して，二つの三角行列を一つにする（図５の右側）ことを再帰的に繰り返す方法（図６）が挙げられる。この場合，一対一通信が$\log_2 P$回のみで済むため，ハウスホルダーQR（一対一通信に換算すると$O(n\log_2 P)$回必要）に比べてはるかに通信回数が削減できていることになる。なお，図５と６では簡単のために単純な（サイクリックではない）ブロック分散の場合を示したが，ブロックサイクリック分散の場合にも，直交変換の形（ブロック構造）をうまく対応させることで，同様の手法が適用できる。



第５図：TSQR で用いる直交変換の構造（左：最初のステップ，右：二番目以降のステップ）

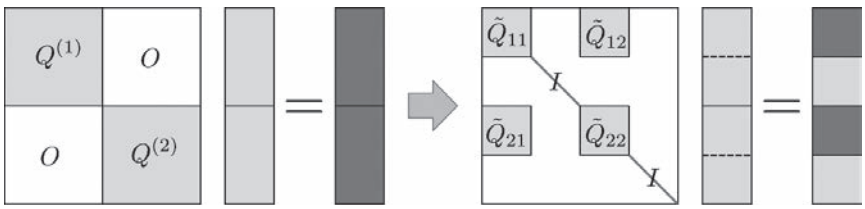


第６図：TSQR における上三角化の様子（4 並列で二分木に従う場合）。

４．２．CAQR アルゴリズム

次に，縦長ではない一般の行列の QR を計算する通信回数削減型アルゴリズムである CAQR[6,7]について説明する。CAQR を一言で述べると，パネルブロッキングを用いたハウスホルダーQR において，パネル内の計算を TSQR で行い，それに伴って更新部分を変更したアルゴリズムである。パネルサイズをうまく調整すれば，パネルが縦長行列となり TSQR を適用できることは自明である。一方，TSQR を適用することで，パネルの上三角化に使用した直交変換が多段化されるので，それに伴い，行列の残りの部分の更新も多段化される（図７）。

CAQR では，パネルの上三角化を TSQR で行うことで，1 つのパネルの計算当たりに必要となる通信回数が，$O(L)$回の集団通信（一対一通信に換算すると$O(L\log_2 P)$回）から$O(\log_2 P)$回となり，アルゴリズム全体を通して通信回数がL分の一になる。これにより通信時間が削減されて性能が向上することが期待される。ただし，Lをあまり大きくしすぎると，通信回数はより削減できるが，TSQR における三角行列同士の QR 分解や多段化された行列の更新部分の演算量が大きくなるため，性能低下が生じる可能性がある。そのため，パネルサイズLを適切に設定する必要がある。



第７図：CAQR において，TSQR に併せて多段化した更新ステップの様子。

５．性能評価

５．１．評価条件等

FX10 の 4800 ノードを用いて，パネルブロッキング版のハウスホルダーQR と通信回数削減型アルゴリズム（TSQR/CAQR）の性能を評価し，両者を比較した。なお，ハウスホルダーQR のパネル内の計算はシンプルなもの（再帰的なパネルブロッキングはなし）とした。プログラムは Fortran90 で実装し，必要に応じて BLAS ライブラリの関数（DGEMM など）を使用し，MPI で並列化を行った。

コンパイラは mpifrtpx を使用し，オプションは-Kfast,openmp とした。また，FX10 で提供されている BLAS ライブラリ（スレッド並列版，-SSL2BLAMP）と MPI ライブラリをそれぞれリンクした。実行時は，1 ノード当たり 1 MPI プロセスで，1 プロセス当たり 16 スレッドとした（スレッド並列化はリンクした BLAS ライブラリ内のみ）。

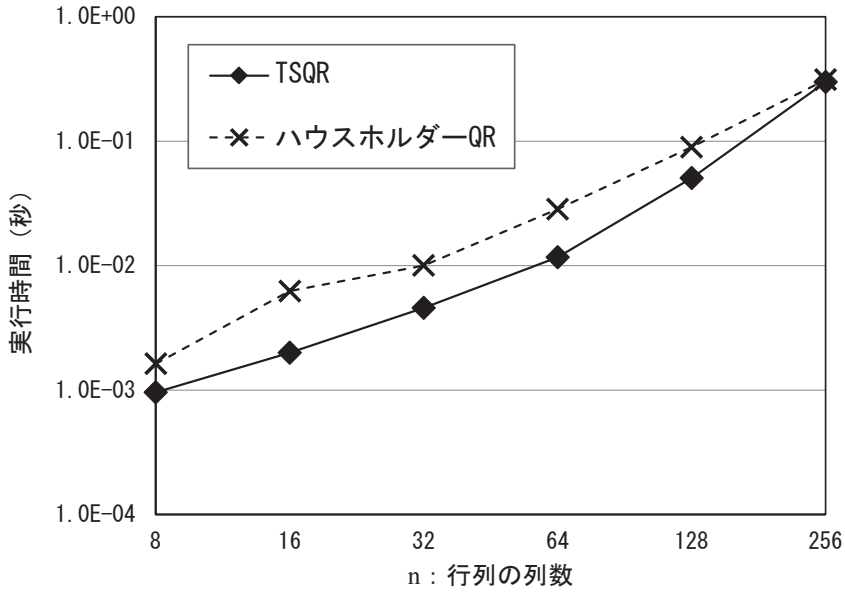
行列データの分散は，すでに述べたように行方向のブロックサイクリック分散とし，実装の都合上，パネルサイズ（ハウスホルダーQR と CAQR 両方とも）と分散におけるブロック幅（図４のNB）は等しいとした。測定したのは，乱数行列の QR 分解の並列計算時間であり，今回はQは直交変換の積として保持する形とした。

５．２．評価結果

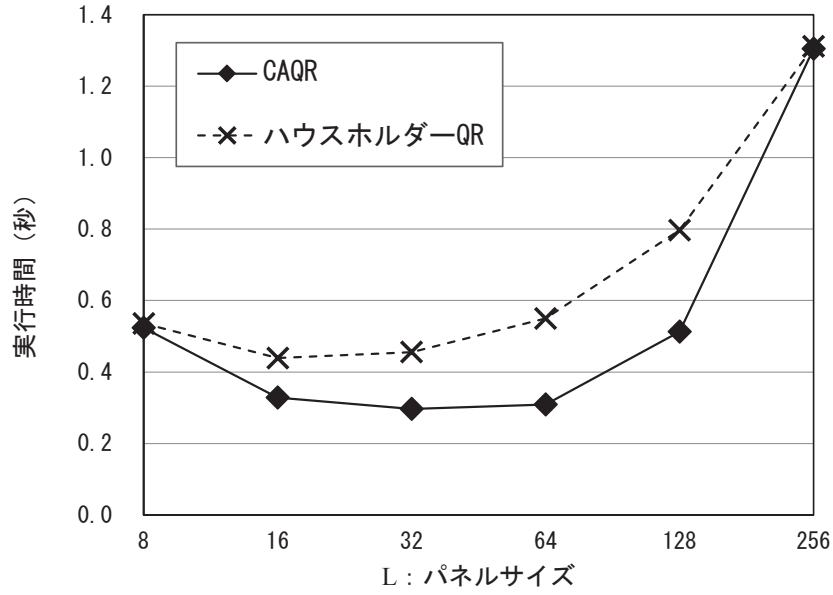
まず，縦長行列の QR 分解に対するハウスホルダーQR と TSQR の並列計算時間を評価した結果を図８に示す。ここでは，$m = 9830400$と固定して，$n = 8,16,\ldots,256$と変化して，両者を比較した。この結果から，TSQR で通信回数を削減した効果が確認できる。ただし，nが大きくなると両者の差がほとんどなくなっており，CAQR においてパネルサイズを適切に設定することの必要性が確認できる。

次に，一般の行列に対するハウスホルダーQR と CAQR の並列計算時間の評価結果を図９から１１に示す。ここでは，$m = 9830400$として，$n = 1024,8192,24576$に対して，パネルサイズLを変化させて計算時間を測定している。まず，全ての結果から，パネルサイズを最適化することの必要性が確認できる。その上で，図９と１０の結果から，CAQR における通信回数削減の効果が読み取れる。一方で，図１１の結果を踏まえると，nが大きくなる場合に CAQR では何か別のボトルネック（恐らく演算時間に関するものだと推測しているが）が生じることで，現状ではハウスホルダーQR よりも性能が低くなってしまう可能性があることが明らかとなった。また，これらの 3 つのケースを比較すると，最適なパネルサイズはnによって異なり，同時に，アルゴリズム（ハウスホルダーQR または CAQR）によっても異なることが確認できる。

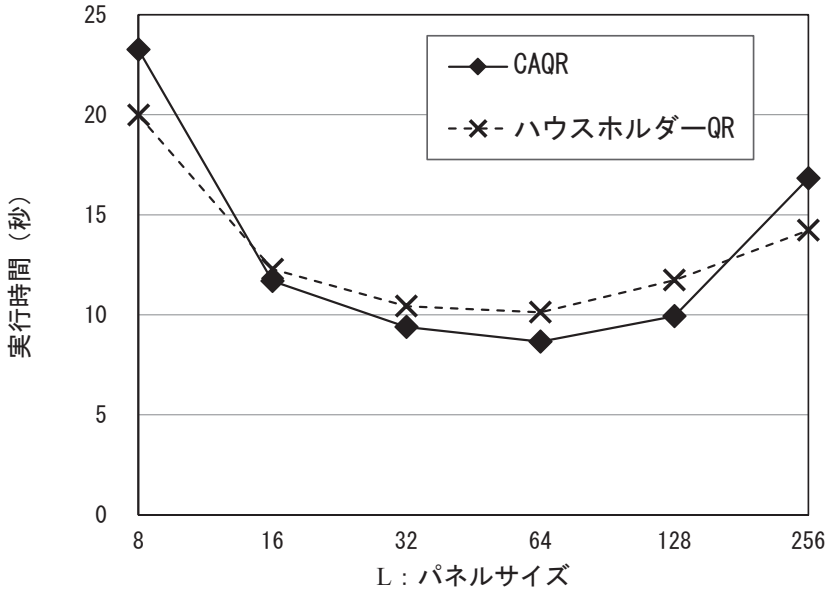
以上の結果から，確かに CAQR(あるいは TSQR)は並列計算時間の面において，従来のハウスホルダーQR よりも優れていることが確認できた。ただし，行列サイズ（と恐らくは計算機環境）によって，優れない場合もあり得ることも確認できたため，詳細な原因の調査が今後の課題である。同時に，パネルサイズの最適化が大きな課題となることも併せて確認できたので，この最適か方法も今後の課題である。



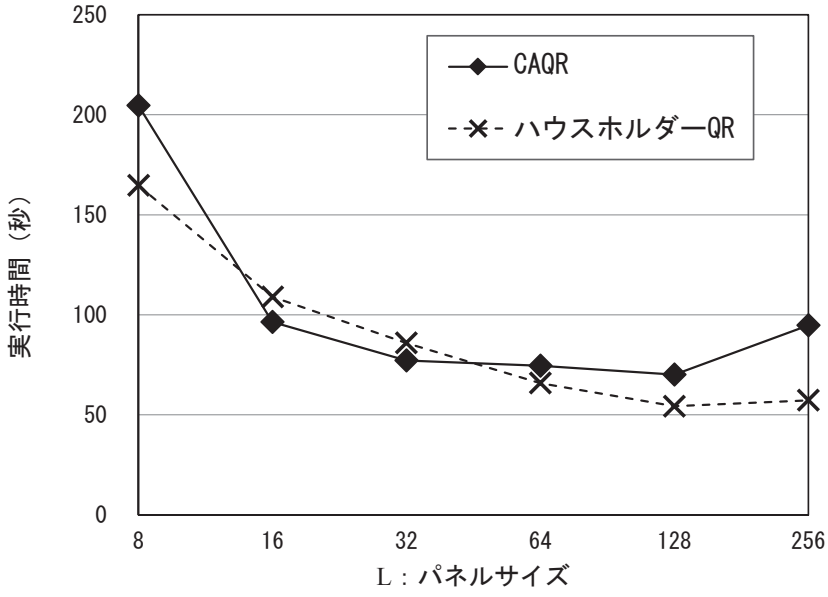
第８図：TSQR とハウスホルダーQR の実行時間。
（行列の行数は m=9830400 で，FX10 4800 ノードを使用。Q は陰的に保持している）



第９図：CAQR とハウスホルダーQR（パネルブロッキング版）の実行時間。
（行列サイズは m=9830400，n=1024 で，FX10 4800 ノードを使用。Q は陰的に保持している）



第１０図：CAQR とハウスホルダーQR（パネルブロッキング版）の実行時間。
（行列サイズは m=9830400, n=8192 で，FX10 4800 ノードを使用。Q は陰的に保持している）



第１１図：CAQR とハウスホルダーQR（パネルブロッキング版）の実行時間。
（行列サイズは m=9830400, n=24576 で，FX10 4800 ノードを使用。Q は陰的に保持している）

６．おわりに

本稿では，FX10 の 4800 ノードを用いて行った，最近注目を集めている通信回数削減型の QR 分解アルゴリズム（TSQR/CAQR）の性能を，従来のパネルブロッキング版のハウスホルダーQR と比較した結果の一部について報告した。今回報告した結果から，CAQR（や TSQR）が大規模分散並列環境で効果的なアルゴリズムであることが確認できた。一方で，(従来のハウスホルダーQR においてもそうであるが）パネルサイズの最適化が重要なポイントとなるので，性能モデル等を利用した（自動）チューニング手法の研究が課題といえる。また，今回の評価で行列の列数が大きくなった場合に，CAQR になんらかのボトルネックが生じて，ハウスホルダーQR よりも性能が低くなったケースが確認されたので，より詳細な性能解析を行い，原因の解明とその対応も必要である。

その他の今後の課題としては，データ分散に関して，二次元のブロックサイクリック分散もよく利用されるので,そのような分散の場合の CAQR の性能評価を行うことが挙げられる。また，CAQR のアイディアを基にして，通信と演算をオーバーラップさせることで並列性を向上させて高性能化を図った，タイル QR[13]と呼ばれるアルゴリズムが提案されているので，その評価も必要である。

謝辞

今回の「大規模 HPC チャレンジ」を通して大変お世話になりました，東京大学情報基盤センターの関係者の皆様に深く感謝申し上げます。また，本研究に対して日ごろから有益なご議論をさせていただいている理化学研究所 計算科学研究機構の今村俊幸チームリーダ,電気通信大学の山本有作教授,及び下記 CREST プロジェクトのメンバーの皆様に感謝いたします。併せて，RSDFT コードにおける CAQR の適用に関して，有益な議論をさせていただいた東京大学の片桐孝洋准教授をはじめとする関係者の皆様にも感謝いたします。本研究は JST CREST プロジェクト「ポストペタスケールに対応した階層モデルによる超並列固有値解析エンジンの開発」（領域名：ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出）の援助を受けています。

参　考　文　献

[1] ScaLAPACK, http://www.netlib.org/scalapack/

[2] 今村俊幸，T2K スパコンにおける固有値ソルバの開発，東京大学スーパーコンピューティングニュース，Vol.11，No.6，pp.12-32，2009.

[3] 深谷猛，超並列環境向け固有値計算プログラムの性能予測モデルの開発，東京大学スーパーコンピューティングニュース，Vol.15 No.6，pp.33-43，2013.

[4] Y. Hasegawa, et al., First-principles calculations of electron states of a silicon nanowire with 100,000 atoms on the k computer, in Proceedings of the 2011 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC11), pp.1-11, 2011.

[5] 片桐孝洋 他, 通信削減アルゴリズム CAQR の RSDFT の直交化処理への適用と評価, 情報処理学会 研究報告ハイパフォーマンスコンピューティング(HPC),Vol.2014-HPC-144,pp.1-6, 2014.

[6] J. Demmel, et al., Communication-avoiding parallel and sequential QR factorizations, CoRR, vol.abs/0806.2159, 2008.

[7] J. Demmel, et al., Communication-optimal parallel and sequential QR and LU factorizations, SIAM J. Sci. Comp, vol.34, no.1, pp.206-239, 2012.

[8] G. Ballard, et al., Minimizing communication in numerical linear algebra, SIAM J. Matrix Anal. Appl., vol.32, no.3, pp.866-901, 2011.

[9] G. H. Golub and C. F. V. Loan, Matrix Computations, 3rd ed. Johns Hopkins University Press, 1996.

[10]C. Puglisi, Modification of the Householder method based on the compact WY representation, SIAM J. Sci. Stat. Comp., vol.13, pp.723-726, 1992.

[11]R. Schreiber and C. F. van Loan, A storage-efficient WY representation for products of Householder transformations, SIAM J. Sci. Stat. Comp., vol.10, pp.53-57, 1989.

[12]D. MORI, Y. YAMAMOTO, and Z. Shao-Liang, Backward error analysis of the AllReduce algorithm for Householder QR decomposition, JPN J. IND. APPL. MATH., vol.29, no.1, pp.111-130, 2012.

[13]F. Song, et al., Scalable tile communication-avoiding QR factorization on multicore cluster systems, in Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC10) pp.1-11, 2010.