自動チューニング機能を備えた固有値計算ライブラリと疎行列反復解法ライブラリの性能評価

黒田久泰　田中献大

愛媛大学大学院理工学研究科

片桐孝洋

東京大学情報基盤センター

櫻井隆雄

(株)日立製作所 中央研究所

1．はじめに

固有値を求めるための計算や大規模疎行列を係数行列とする線形方程式を解くための計算は様々な分野の数値シミュレーションにおいて頻繁に出てくる計算である。我々は広く汎用的に利用できる固有値計算ライブリと疎行列反復解法ライブリの開発を行っている。我々の開発しているライブラリは，解こうとしている問題や実行している計算機に応じて，演算カーネルやMPIの通信方式などを静的（ライブラリインストール時）あるいは動的（ライブラリ実行時）に最適なプログラムの実装コードを選択するといった自動チューニング機能を有している。今回，東京大学情報基盤センターFX10スーパーコンピューターシステムにおいて，4800ノード(76800コア)という大規模なノード数（コア数）で性能評価を行ったのでそれについて報告する。

大規模疎行列を係数行列とする連立一次方程式の解法では，メモリ容量の制約からガウス消去法のような直接解法ではなく，反復解法を用いて解くことが一般的である。係数行列が対称行列の場合には共役勾配法がよく利用されるが，大規模並列計算機環境においてはノード数が多くなると通信時間が増大するため，疎行列ベクトル積に多くの時間が費やされる。係数行列が非対称行列の場合には，GMRES(m)法のようなクリロフ部分空間法を利用した反復解法が利用されるが，こちらは疎行列ベクトル積だけではなくベクトルの直交化にも多くの時間が費やされる。

自動チューニング機能付きライブラリにおいては，実行時間が最も小さくなる実装方式を選択できるかどうかが重要な鍵である。そこで，自動チューニングにかかる時間の短縮及び正確な見積もりのためにMPIライブラリの中のMPI_Barrier，MPI_Alltoall，MPI_Allreduce，MPI_Reduceの性能について調べた。

また，疎行列ベクトル積においては，係数行列の非零要素の位置の分布に合わせてベクトルの要素の値をノード間でやりとりする必要があるが，そこには多くの実装方法が考えられる。今回，非零要素の個数や分布が異なる66パターンの疎行列を作成した。また，実装としては大きく分類して6種類を用意し，これらの組み合わせで疎行列ベクトル積の性能を評価した。行列の一辺のサイズとしては最大で2の30乗となる1073741824，1行あたりの非零要素数は最大で1369，行列全体の非零要素数では最大で約1兆4700億といった大規模な疎行列を対象とした疎行列ベクトル積の性能評価を行うことができた。

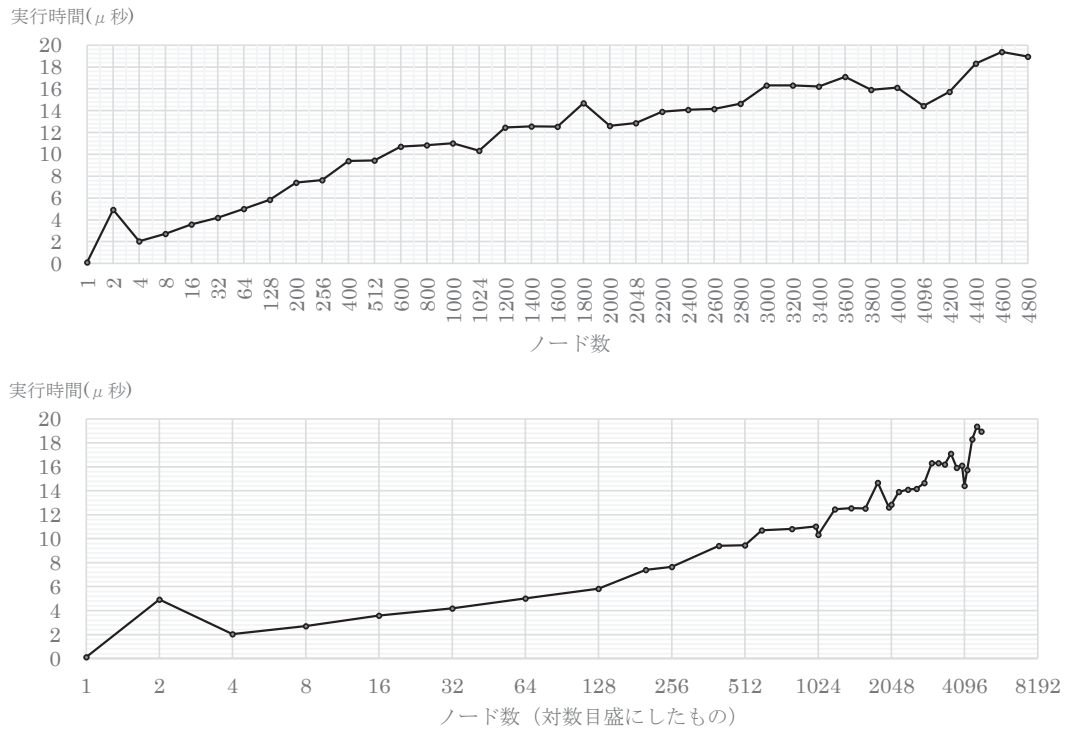
２．MPI 関数の基本性能

MPI 関数の中でもよく使われる MPI_Barrier，MPI_Alltoall，MPI_Allreduce，MPI_Reduce の性能について報告する。なお，MPI 関数の性能については，ノード数が 200 から 4800 までの 200 刻みになるときと 1，2，4，8，…，2048，4096 といった 2 の冪乗になるときを測定した。また，MPI_Barrier を除いては送信サイズあるいは演算対象となるデータの個数を変えて測定を行った。

２．１．MPI_Barrier の性能

自動チューニング機能付き数値計算ライブラリにおいては，適用可能な実装方式の計算時間を正確に測定するため，全ノード間の同期を取らなくてはならない場合がある。その同期のオーバーヘッドを把握しておくことは重要である。MPI_Barrier を 100 万回実行して 1 回あたりの平均実行時間を測定したところ，第 1 図のようになった。600 ノードのときに 10 μ 秒を超え，3000 ノードのときに 15 μ 秒を超えることがわかる（ただし 4096 ノードでは 14 μ 秒）。また，4800 ノードの場合でも 20 μ 秒以下の実行時間だった。さらに，ノード数が 2 の冪乗ではないときも高い性能が得られていることがわかる。

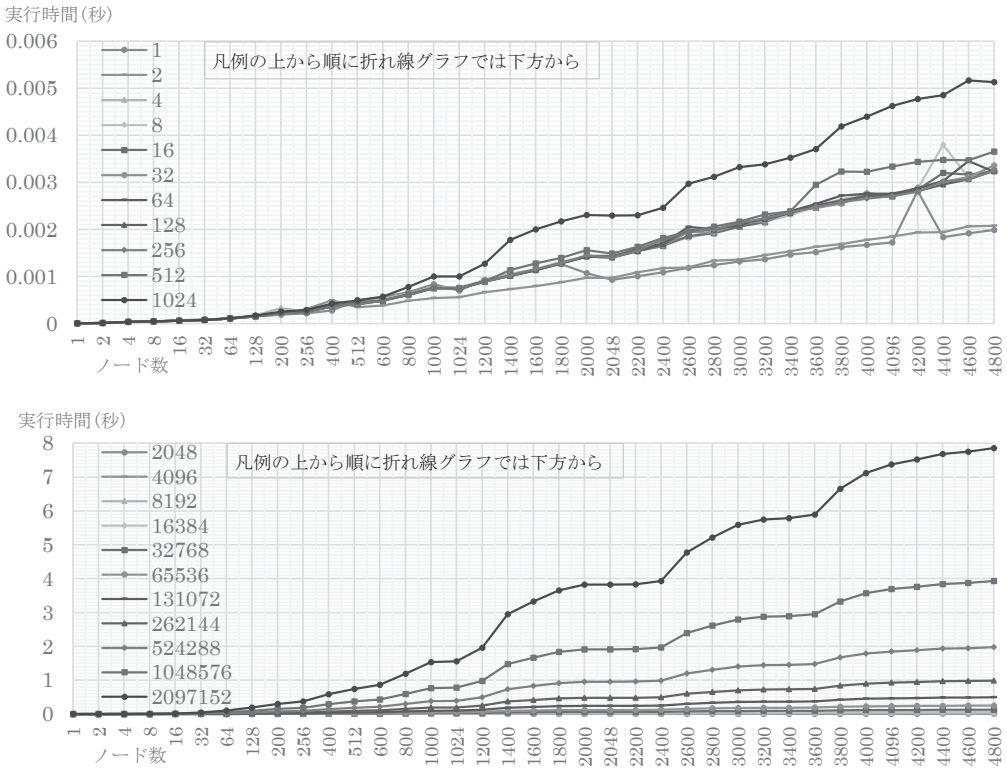
第 1 図の下の図は横軸のノード数を対数表示にしたものである。MPI_Barrier の実行時間はアルゴリズムのステップ数を考えれば，ノード数を p としたとき $O(\log p)$ となり，グラフ上では直線になるのが理想的である。しかしながら，ノード数が 128 を超えたあたりから傾きが大きくなっており通信のオーバーヘッドが生じていることがわかる。



第 1 図：MPI_Barrier の性能

２．２．MPI_Alltoall の性能

MPI_Alltoall は各ノードが異なるデータを相手先ノードに送信するための関数であり行列の転置などに使われる。MPI_Alltoall の相手先 1 ノード毎の送信サイズを 1 バイトから 2^{21}(=2097152)バイトまで 2 の冪乗になるようにし,1 回毎に MPI_Barrier を挟んで MPI_Alltoall を 5 回実行して 1 回あたりの平均実行時間を測定したところ，第 2 図のようになった。上の図は送信サイズが 1 バイトから 1024 バイトのとき，下の図は送信サイズが 2048 バイトから 2^{21}(=2097152)バイトのときである。



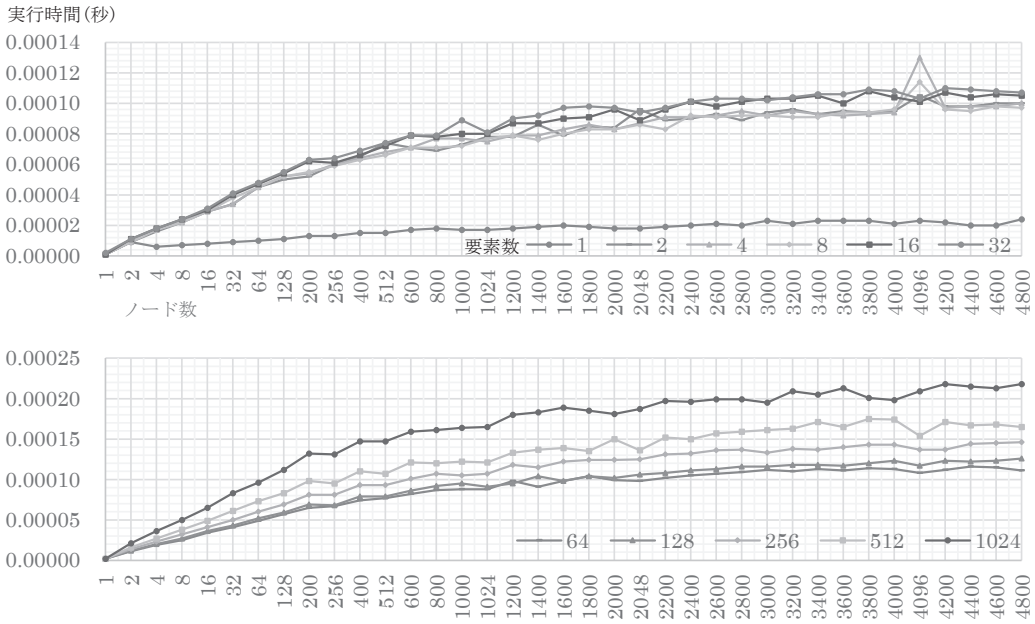
第 2 図: MPI_Alltoall の性能

上の図からは，FX10 に特有な結果として，送信サイズが 1 バイトと 2 バイトのときにほぼ同じような性能値を示し,送信サイズが 4 バイトから 512 バイトの間でもほぼ同じような性能値を示していることがわかる。

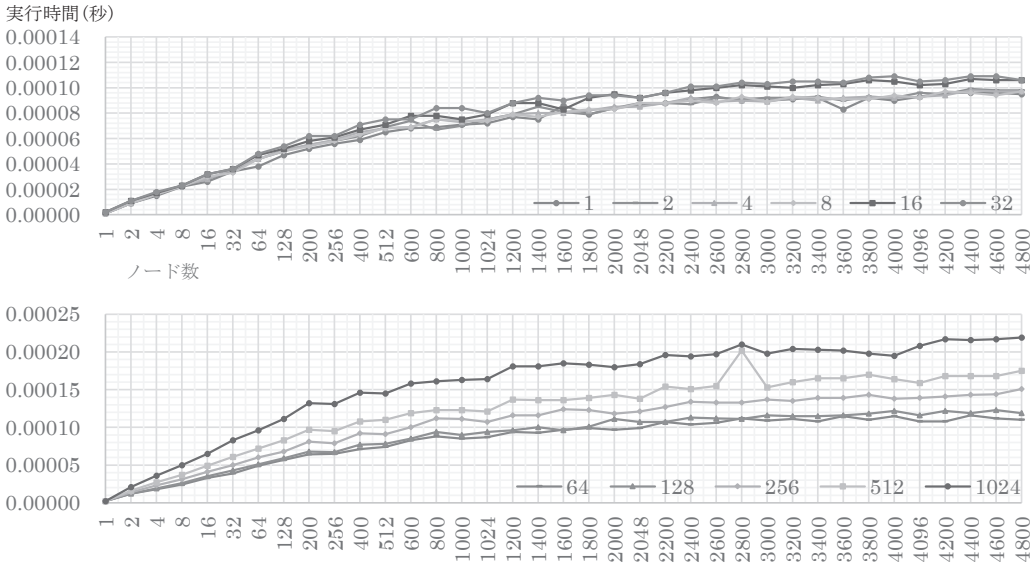
また，ノード数を p，送信サイズを s としたとき MPI_Alltoall の理想的な実行時間は $O(ps)$ である。実際，下の図において送信サイズが大きいとき，1200 ノード，2400 ノード，3600 ノード，4800 ノードのところを線で結ぶと実行時間がノード数に比例していることがわかる。しかしながら,その間のノード数のところではその直線よりも上側にずれており効率が悪くなっていることがわかる。

２．３．MPI_Allreduce の性能

MPI_Allreduce は複数ノードに跨ったベクトル同士の内積演算を行う際に利用される。ここでは倍精度実数（8 バイト）の要素数が 1 から 1024 までの間の 2 の冪乗になるときを測定した結果を第 3 図に示す。ここでは 20 回の測定を行いその平均を取っている。また，FX10 においてはリダクション演算の順序を固定することで，ノード数が同じであれば常に同じ計算結果となるようにすることができる[1]。実際に実行時に「mpiexec -mca coll_base_reduce_commute_safe 1」のように指定して，リダクション演算の順序を固定した場合の実行時間を第 4 図に示す。

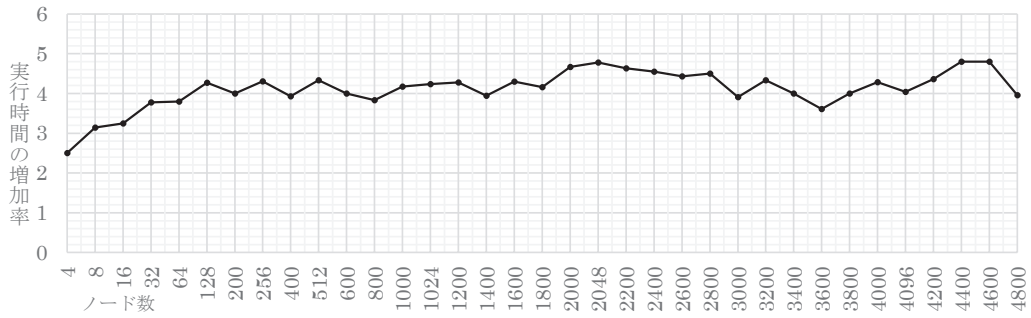


第 3 図: MPI_Allreduce の性能（通常）



第 4 図: MPI_Allreduce の性能（演算順序を固定した場合）

第 3 図と第 4 図からわかるとおり，要素数が 1 以外のときは coll_base_reduce_commute_safe の指定の有無でほとんど実行時間に違いは見られなかった。要素数が 1 のときは大きな違いが見られ，第 5 図に示すようにノード数が 4 から 4800 までの間において実行時間が相乗平均で約 4.1 倍増加するという結果になった。

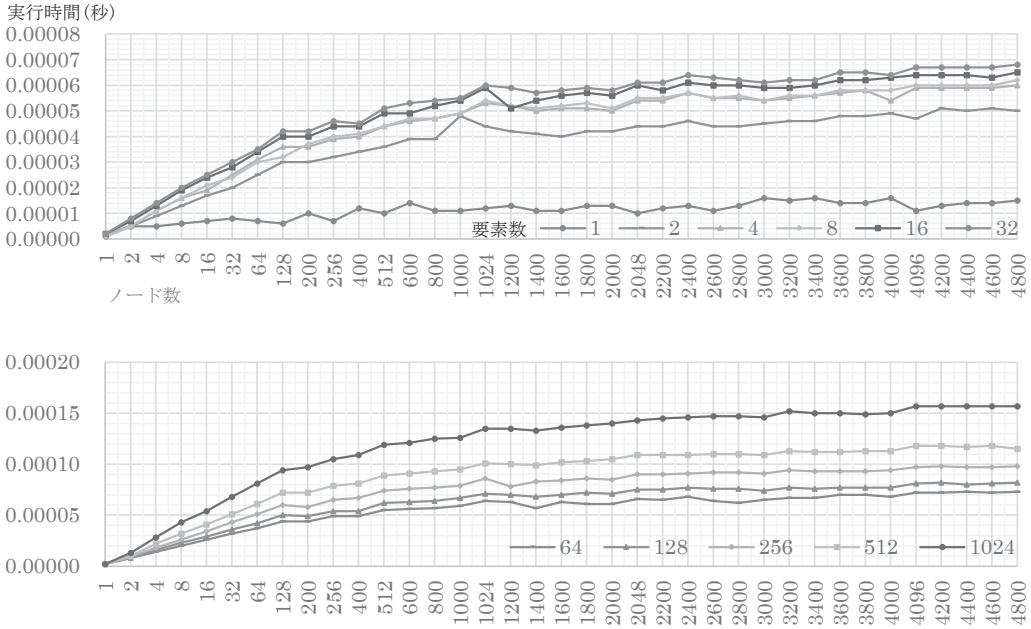


第 5 図：演算順序を固定したときの MPI_Allreduce の実行時間の増加率（要素数 1）

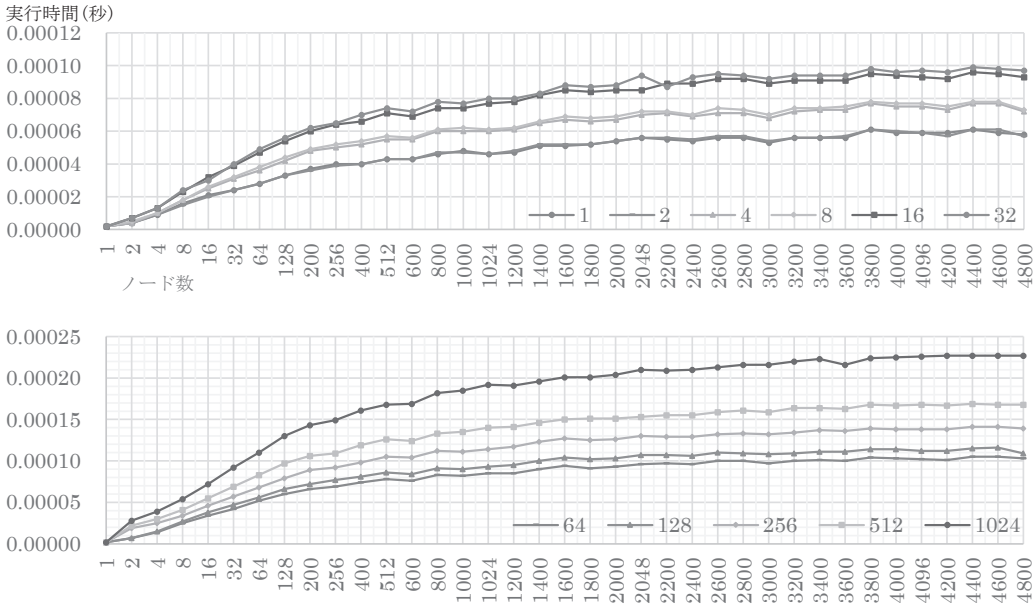
これらの結果から，一組のベクトル同士の内積演算を行う場合に演算順序を固定することは大幅な性能低下が発生するが，二組以上のベクトル同士の内積演算を同時に行う場合には演算順序の固定の有無で性能にほとんど差がでないことがわかる。

２．４．MPI_Reduce の性能

MPI_Reduce は MPI_Allreduce とは異なり 1 つのノードだけが演算結果を知ればよいときに利用される。ここでも倍精度実数（8 バイト）の要素数が 1 から 1024 までの間の 2 の冪乗になるときを測定した結果を第 6 図に示す。また，MPI_Allreduce のときと同様にリダクション演算の順序を固定した場合の性能については第 7 図に示す。

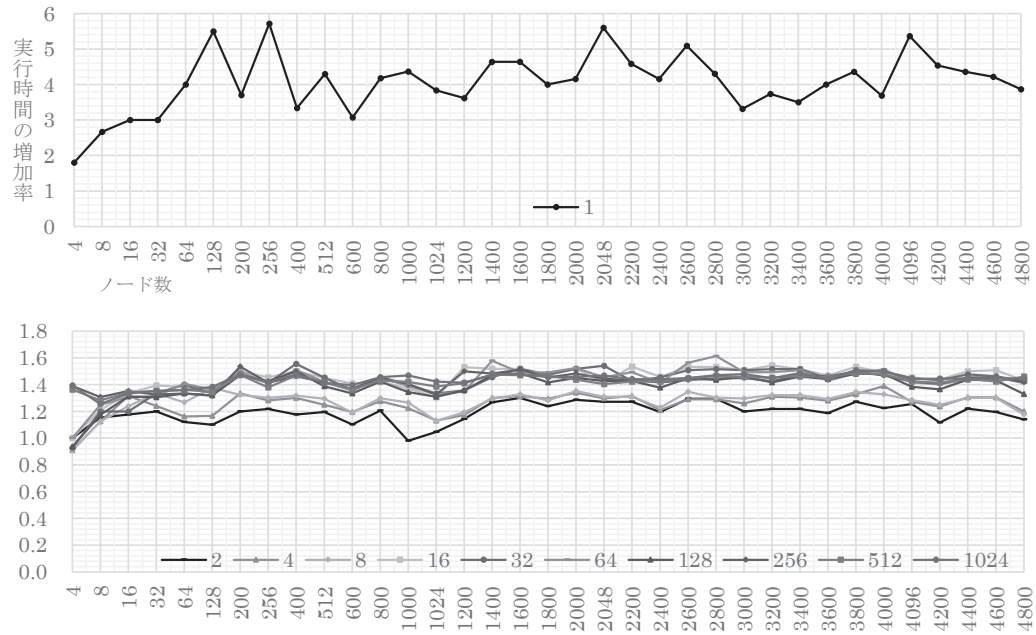


第 6 図：MPI_Reduce の性能（通常）



第 7 図: MPI_Reduce の性能（演算順序を固定した場合）

MPI_Allreduce のときとは異なり，coll_base_reduce_commute_safe の指定の有無で実行時間に違いが見られる。要素数が 1 のときは，第 8 図の上の図に示すようにノード数が 4 から 4800 までの間において実行時間が相乗平均で約 4.0 倍増加するという結果になった。また，下の図に示すように要素数が 2 のときは約 1.2 倍，要素数が 4 と 8 のときは約 1.3 倍，要素数が 16 から 1024 までのときは約 1.4 倍の実行時間の増加が見られた。



第 8 図: 演算順序を固定したときの MPI_Reduce の実行時間の増加率

３．並列疎行列ベクトル積

計算機で疎行列を扱うとき，通常，非零要素の存在する場所の情報とその非零要素の値のみを保持する。ここでは，疎行列格納方式の一つである CRS(Compressed Row Storage)形式を採用する。CRS 形式は疎行列を行方向に走査して非零要素の位置情報と値を格納する格納形式である。

行列 A を CRS 形式で格納された疎行列，x と y をベクトルとし，疎行列ベクトル積 y = Ax を計算することを考える。疎行列ベクトル積 y = Ax を計算するプログラムでは疎行列 A の非零要素を取り出し，その非零要素の列インデックス値に対応するベクトル x 内の要素との積を計算し，同じ列インデックス値に対応するベクトル y の要素に加えていく，という手続きを踏むことで計算を行う。FX10 は分散メモリ型並列計算機であるため，行列 A は行方向にブロック分割，ベクトル x と y もブロック分割されているとする。そして，疎行列ベクトル積計算自体も行方向ブロック単位で分解し，それらの領域をそれぞれのノードで計算させるという並列化手法を用いる。

３．１．並列疎行列ベクトル積に伴うノード間通信

各ノードのプロセスは行列 A の一辺のサイズと同じサイズのベクトルをテンポラリとして利用できるとし，このベクトルを t とする。通常，各プロセスはブロック分割されている x の値の全てを一旦 t に保持する必要がある。この実装方法として最も単純であるのは集団通信関数である MPI_Allgather を用いることである。しかしながら，MPI_Allgather では各プロセスで実行される疎行列ベクトル積に必要とされないデータも含めて全て転送されてしまうため，通信量と通信時間が増加してしまう。そこで我々は各ノードでの疎行列ベクトル積に必要な非零要素の値に絞ってデータの送受信を行う方法を開発している[2,3]。ここでは，下記の 6 つの実装方式についての性能を報告する。

(1) 全対全通信

MPI_Allgather を用いて各ノードが他の全てのノードに対して自分が持つベクトルデータの全てを送信する。

(2) 範囲限定通信

各ノードが他のノードに対し，必要とする最小の連続した範囲のブロックのみを送信する。ただし，1 つの相手先ノードに対して 1 回のみの通信で行うようにする。また，自分が持つベクトルの要素を一切必要としない相手先ノードに対しては通信を行わない。この処理はさらに MPI_Isend-MPI_Irecv，MPI_Irecv-MPI_Isend，MPI Sendrecv をそれぞれ用いて実装している。ここで MPI_Isend-MPI_Irecv とは，MPI_Isend を呼び出した後に MPI_Irecv を呼び出し，MPI_Irecv-MPI_Isend とは MPI_Irecv を呼び出した後に MPI_Isend を呼び出すことを意味する。

(3) 最小量通信

1 つの相手先ノードに対して最大 1 回しか通信を行わないという点は(2)の範囲限定通信と同じであるが，各ノードは相手先ノードが必要とする要素のみを 1 つの連続したバッファー領域にパッキングして送信する。通信量は最小になるが，通信前と通信後にバッファー領域に要素を格納するパッキングと，元の並びに戻すアンパッキングの処理が必要となる。

(4) 最小限通信

(3)の最小量通信では相手先ノードにおいて計算に使わないベクトルの要素は一切含まないが，この実装方式ではそれらの要素を含んでパッキングを行う。例えば，相手先ノードが必要とする

2つの非零要素の間に必要としない非零要素が何個か含まれていたとしてもひとまとめにして処理を行う。(3)の最小量通信よりも通信量が増えるが，パッキングとアンパッキングの処理の時間とメモリ使用量を抑えることができる。

(5) アンパッキング無し最小量通信

送信する際には(3)の最小量通信と同じようにバッファー領域にパッキングしてから送信を行うが，受信側では一切アンパッキングの処理を行わずに直接バッファー領域にあるベクトルの要素の値を参照して疎行列ベクトル積を行う。

(6) アンパッキング無し最小限通信

(5)のアンパッキング無し最小量通信では相手先ノードにおいて計算に使わないベクトルの要素は一切含まないが，この方式ではそれを含んでパッキングを行う。非零要素の位置情報を保持するメモリが節約できるようになるため(5)のアンパッキング無し最小量通信に比べてメモリ使用量を抑えることができる。

３．２．測定対象行列

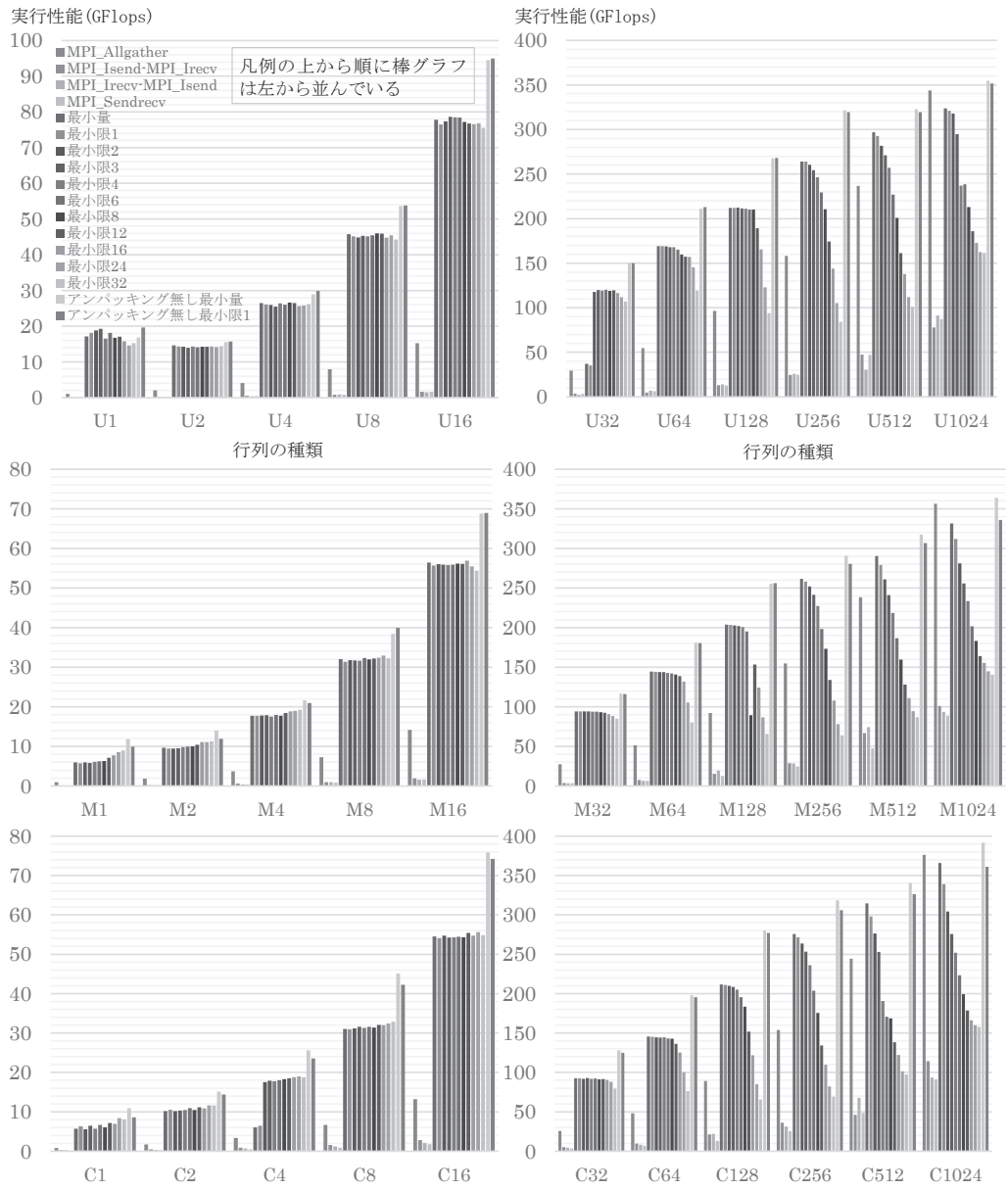
疎行列ベクトル積の計算においては，非零要素の位置の分布のみが重要になってくる。我々は人工的に第1表に示す疎行列データを作成し，評価を行った。

第1表：測定対象行列

行列名	行列の説明
U1, U2, …, U1024 （11パターン）	各行毎に一様分布乱数を用いて非零要素の位置を決める。 Uに続く数値は1行あたりの非零要素数を示す。
M1, M2, …, M1024 （11パターン）	各行毎に対角要素付近に非零要素が多く分布するように非零要素の位置を乱数で決める。Mに続く数値は1行あたりの非零要素数を示す。
C1, C2, …, C1024 （11パターン）	各行毎に対角要素付近に非零要素がさらに多く分布するように非零要素の位置を乱数で決める。Cに続く数値は1行あたりの非零要素数を示す。
B2_1,B2_2,…,B2_1024 （11パターン）	対角要素のインデックス値を d としたとき $d-\mathrm{n}/2 \sim d+n/2$ のインデックスの範囲（ただし $1 \sim n$ に収める）に非零要素が乱数により一様分布する。 B2_に続く数値は1行あたりの非零要素数を示す。
B4_1,B4_2,…,B4_1024 （11パターン）	対角要素のインデックス値を d としたとき $d-\mathrm{n}/4 \sim d+n/4$ のインデックスの範囲（ただし $1 \sim n$ に収める）に非零要素が乱数により一様分布する。 B4_に続く数値は1行あたりの非零要素数を示す。
B8_1,B8_2,…,B8_1024 （11パターン）	対角要素のインデックス値を d としたとき$d-\mathrm{n}/8 \sim d+n/8$ のインデックスの範囲（ただし $1 \sim n$ に収める）に非零要素が乱数により一様分布する。 B8_に続く数値は1行あたりの非零要素数を示す。

３．３．実行結果

行列の一辺のサイズが 2^{29}(=536870912)，ノード数が 4800 において，実装方式と行列を変えて性能を測定した結果を第 9 図に示す。縦軸は 1 秒間あたりの浮動小数点数演算の回数（GFlops 値）を示しており，大きいほど実行性能が高いことを示す。

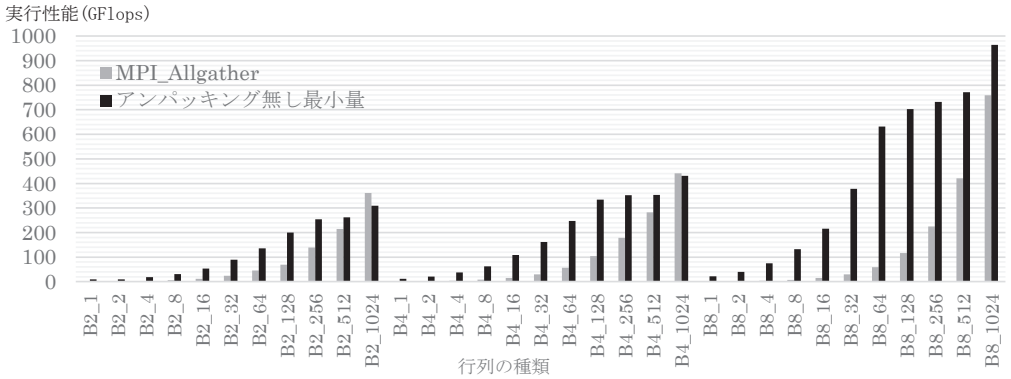


第 9 図：疎行列ベクトル積の計算性能（U，M，C）

「最小限 n」の意味は，相手先ノードに必要な 2 つの非零要素の間に t 個の不必要な非零要素が含まれていた場合に，t≦n のときは前後の 2 つの非零要素と連結してデータの送受信の対象とする。

通信量としては一番多くなる MPI_Allgather を用いた実装方式も 1 行あたりの非零要素数が 1024 になってくると他の実装方式に比べて高い性能を示していることがわかる。

第 10 図に行列 B2，B4，B8 の MPI_Allgather を用いた実装方式とアンパッキング無し最小量通信の 2 つの実行性能を示す。ここでも，非零要素数が 1024 になると MPI_Allgather を用いた実装方式が高い性能を示していることがわかる。



第 10 図：疎行列ベクトル積の計算性能（B2，B4，B8）

第 9 図と第 10 図から，全体を通してアンパッキング無し最小量通信が高い性能を示していることがわかる。なお，(6)のアンパッキング無し最小限通信は計算時間の短縮よりも大規模な行列に対して計算を行えるようにするために開発したものである。

４．まとめ

疎行列ベクトル積は連立一次方程式を反復解法で解く際に頻繁に行う計算であり，大規模並列計算機環境においては多くの通信が発生するため全体の計算時間の中で多くの割合を占める。さらに，疎行列ベクトル積の実行時間は，疎行列部分の非零要素の位置の分布構造に大きく左右される。そのため，一つの実装方式に頼ったのでは高い性能を得ることは難しい。我々は，送信の際に相手先ノードが必要とする非零要素の値をバッファー領域に詰め込むパッキングの処理は行うものの，受信側ではアンパッキングの処理は行わずに直接疎行列ベクトル積を行うというアルゴリズムを開発し実装を行った。4800 ノードという大規模な並列計算機環境において高い性能が出ていることが今回の大規模 HPC チャレンジで確認することができた。

謝辞

本研究の一部は，文部科学省 科学技術研究費補助金，基盤研究（B），課題番号：24300004，「実行時自動チューニング機能付き疎行列反復解法ライブラリのエクサスケール化」による。

参 考 文 献

[1] 片桐孝洋：富士通 PRIMEHPC FX10 チューニング連載講座 5. MPI 性能チューニング，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.15，No.5，pp.10-21，2013

[2] 田中献大，黒田久泰：FX10 における疎行列ベクトル積で必要となる通信部分の高速化について，日本応用数理学会環瀬戸内応用数理研究部会第 16 回シンポジウム講演予稿集，pp.38-41，2013

[3] 田中献大，黒田久泰：FX10 における RDMA を用いた疎行列ベクトル積の通信性能の評価，平成 25 年度電気関係学会四国支部連合大会講演論文集，p.275，2013