2008－HPC－116 (8)
2008／8／5

Windowsクラスタにおける疎行列反復解法ソルバの自動チューニング

片桐孝洋†, 黒田久泰†

本稿では, Windowsクラスタにおける以下の2点を評価する：(1) MS-MPIの通信性能；(2) 疎行列反復解法ソルバにおける実行時自動チューニング機能の有効性. 性能評価の結果, MS-MPIは特定のメッセージサイズで性能が劣化する部分があるものの全体として高い実行性能を持つことが確認できた. また疎行列反復解法ソルバでは, 標準設定では解けない問題が解けるようになるなど有効性が高いことが確認できた.

An Auto-tuning Method for Sparse Matrix Iterative Solvers on Windows Cluster

Takahiro Katagiri†, Hisayasu Kuroda†,

In this presentation, we evaluate the following two topics on Windows cluster: (1) Communication performance of MS-MPI; (2) Effectiveness of a run-time auto-tuning facility for sparse matrix iterative solvers. As a result of our evaluation, we found that MS-MPI was good performance with some exceptions. In addition, we confirmed such a high efficiency that the sparse matrix iterative solvers can solve the problem which has not solved with a standard parameter setting.

1. はじめに

マルチコアCPUが普及し, 並列計算機が複雑な計算機構成を取るようになってきた. システムの大規模化が容易となり, ハイエンドな並列計算環境では10万コアを有するシステムも実現されている. このような複雑化・大規模化した並列環境では, コンパイラによる自動並列化に留まらず人手によるチューニングさえも困難となる. 性能チューニングの自動化が急務となることは疑う余地がない. この要請に対し, 自動チューニング機能付き数値計算ライブラリの研究開発がなされてきた[1-3].

一方, Windowsを利用したPCが広く普及している. 特にビジネス分野において, Windows PCを利用したPCクラスタを用いて並列処理を行いたいというニーズが出てきている. このような背景から本研究では, 以下の2つを目的とする. 第1に, 身近となったWindows PCをクラスタ化して並列処理をする場合の基本性能を, 実用的な数値計算ライブラリの観点から評価する. 第2に, 数値計算ライブラリの中でも疎行列を対象とする反復解法ソルバでは, 実行時に定まる行列データの数値特性から最適な数値アルゴリズムが定まる. この最適なアルゴリズムは, 従来の商用ライブラリでは実行時に自動性能チューニングする機能がないため, ユーザが自ら難しい内部パラメタを調整する必要があった. そこで本研究では, 実行時におけるアルゴリズム選択機構, および通信実装方式の選択機構を自動チューニング機能として付加した場合の有効性検討を, Windows HPC Server 2008(β2)のクラスタ環境で行う.

†東京大学情報基盤センター スーパコンピューティング部門
Supercomputing Division, Information Technology Center, The University of Tokyo

2. 疎行列反復解法GMRES(m)法

2.1 処理概略とベンチマークの観点での特徴

本稿で用いる疎行列反復解法ソルバは, GMRES(m)法にMPIを用いて分散メモリ型並列環境向けに並列化した数値計算ライブラリである. 数値アルゴリズム, 疎行列カーネル, 並列実装方式に関する自動チューニング機能が付加されている[2]. 本研究は新しい数値アルゴリズムを提案することは目的ではない. GMRES(m)法による疎行列反復解法ソルバを1つの並列ベンチマークとして捉えた時の性能評価と, 実行時自動チューニング機能の有効性を評価することにある. GMRES (m) 法の概略を図1に示す.

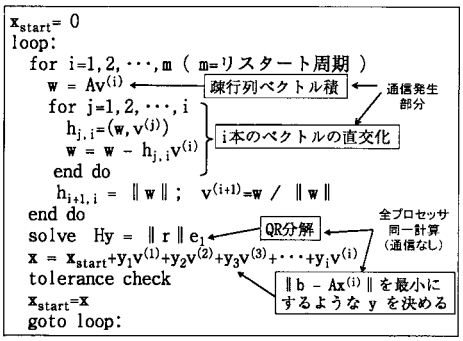

図 1 GMRES (m) 法の概略

ここで，リスタート周期 m の値は疎行列の数値特性により定まる性能パラメタであり，事前に最適値を決定することはできない．

この GMRES(m)法ベンチマークで最も特徴的な点は，前処理方式の評価を含む点である．前処理方式は解法の収束性に影響する数値アルゴリズム上重要な処理であるばかりか，疎行列・ベクトル積演算と通信実装へ影響するので，双方の性能評価への影響が大きい．したがって前処理方式を含めた性能評価は，実用上評価されるべき項目であるといえる．

2.2 並列実装方式

疎行列・ベクトル積演算(以下，SpMxV)は，入力疎行列の非零要素の分布に応じ最適な並列実装方式が異なる．以下にそれを述べる．

SpMxV において，疎行列 A がランダム行列か密行列に近い場合，集団通信関数である MPI_Allgather が使われる．MPI_Allgather は，全プロセスが従事するので，通信コストは大きい．ところが，疎行列 A が対角部分の非零要素が集中しているような場合だと，MPI_Allgather を用いた実装は最適ではない．この場合は，隣接プロセスへの 1 対 1 通信処理だけで実装できる．このように非零要素の分布に応じて，適した実装方法が異なる．これは，(1)計算ノード数；(2)各ノードが処理する疎行列の行数；(3)通信ライブラリ性能；に依存する．疎行列の非零構造の問題に加え，実行するノード数ですら実行時にならないとわからない状況もあるので，最適な実装方式を事前に決定できない．

以上から，SpMxV の最適な並列実装方式選択では，実行時の自動チューニングが必須である．

2.3 SpMxV カーネルの実装方式

SpMxV カーネルはループ長が疎行列データの非零分布に依存するため，最適な実装方式は実行時にならないと定まらない．特に演算カーネルのループアンローリングは SpMxV でも有効な高速化手法となるが，ループ長が実行時にならないと定まらないため実装が困難である．したがって，コンパイラによる静的最適化も効果があまり期待できない．一般に，SpMxV の行方向の最大ループ長は行毎に異なるため，アンローリング段数を固定にすることができない．したがって，複数のアンローリング段数の実装を持ち，かつ実行時に適切なアンローリングを選択する機構(実行時自動チューニング機構)を有すソルバが性能面で有効である．ただし，実行前にループ長毎の最適なループアンローリング方式が判断できることから，事前に個別のループ長における最適な実装方式を調査して，実行時にその情報をもとに実装方式を選択することが可能である[2]．また，ユーザが事前に疎行列形状の情報をユーザ知識として与える方法[3]も考えられる．

2.4 自動チューニング機能の概略

GMRES(m)法の処理概略，および SpMxV の実装方式を考慮すると，実行時に行うべき自動チューニング項目は以下のようになる．

【数値アルゴリズムに起因するもの】

i) 前処理方式 (ブロック ILU, 行列多項式)

ii)リスタート周期

iii)ベクトル直交化方式 (CGS, MGS, IRCGS)

【実装方式に起因するもの】

・単体性能に起因するもの

iv)SpMxV に対するループアンローリング方式

・並列処理に起因するもの

v) SpMxV に対する通信方式 (MPI_Allgather 関数, MPI_Bcast 関数, 1 対 1 ブロッキング通信, 1 対 1 ノンブロッキング通信 (Isend 関数先行発行版), 1 対 1 ノンブロッキング通信(Irecv 関数先行発行版)など)

3. 性能評価

3.1 実験環境

マイクロソフト(株)新宿本社にある Windows HPC Server 2008(β2)クラスタを利用した．計算ノードは，SGI Altix XE310，CPU は Intel Xeon 5365 (4CPU コア，3.00GHz，FSB 1,333MHz，共有 L2 キャッシュ 8MB)を 2 個搭載，メモリは 16GB である．開発環境は Visual Studio 2008 を利用した．ネットワークは，各計算ノードに Intel PRO/1000EB が 2 つあり，そのうちの 1 つを MS-MPI 専用として利用した．計算ノードは 4 台あり，このシステム全体での CPU コア数は 32 である．

疎行列反復解法ソルバは，東京大学情報基盤センターで開発された ILIB_GMRES[2]を利用した．

3.2 MS-MPI の性能評価

ここでは GMRES(m)法で頻繁に使われる MPI 関数についての性能を調べた．1 対 1 通信については，MS-MPI の性能を客観的に評価するため，自前で作成した Winsock(TCP/IP)による通信性能との比較を行った．

図 2 にノード間の 1 対 1 通信性能を示す．メッセージサイズが 512KB から 4MB の間で，MS-MPI では約 90MB/秒という高い通信性能を出しており，単純な TCP/IP 実装よりも良い結果となっている．一方，メッセージサイズが 8MB を超えてからは MS-MPI は通信性能が大きく落ちている．Windows CCS 2003 の MS-MPI ではこのような現象は起きていないので，今回使用した MS-MPI のバージョン固有の問題であると考えられる．

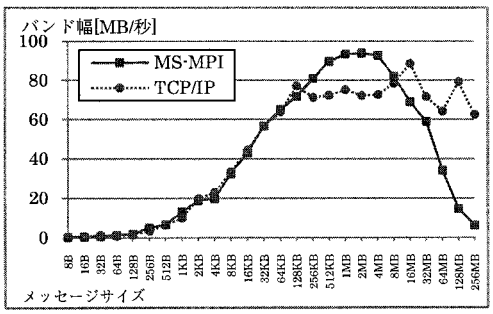

図 2 ノード間 1 対 1 通信の通信性能

図 3 は同一ノード内の 2 つの MPI プロセス間を MPI で通信したときの性能である．ここでは，正確に測定するために，SetProcessAffinityMask 関数を用いて，各 MPI プロセスの実行する CPU を固定化した．L2 キャッシュ 4MB を共有する 2 つの CPU コア間の通信性能はメッセージサイズが 1 MB に近付くにつれとても高くなることがわかる．ただし，これは 1000 回の繰り返しで測定しているため，繰り返し実行しない場合には，このような現象は生じない．また，同じソケット内であっても L2 キャッシュを共有していない 2 つの CPU コア間の通信性能はノード内の 2 つのソケット間の通信性能とほぼ同じ性能であることがわかる．

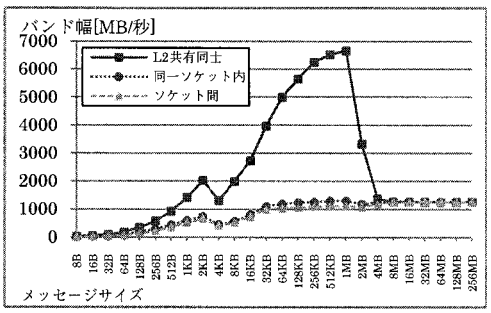

図 3 ノード内 1 対 1 通信の性能(MS-MPI)

図 4 は同一ノード内で TCP/IP を利用した場合である．MS-MPI に比べて格段に性能が落ちることから，MS-MPI はノード内の MPI 通信に関しては十分な最適化がなされていると考えられる．

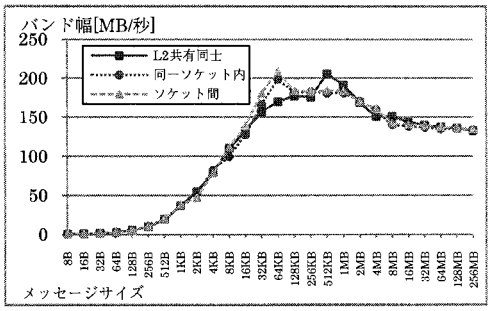

図 4 ノード内 1 対 1 通信の性能 (TCP/IP)

一方，ノンブロッキング通信を利用して計算と通信をオーバラップする方法もよく使われる．この効果を得るためには，通信と計算が重複できるような通信機構が実現されている必要がある．この通信機構を評価するため，ノードに計算負荷をかけた場合とかけない場合でノンブロッキング通信性能を比較した．それを図 5 に示す．

ここでは2ノードを用いて，各ノードは1MPIプロセス(スレッド並列無しで1CPUコアにのみ負荷をかける)とした．

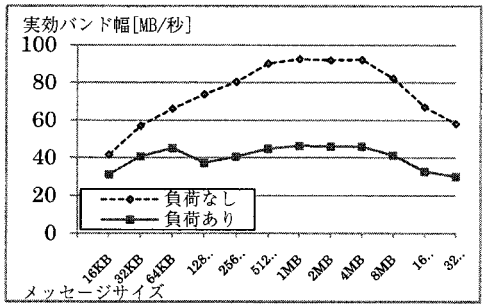

図 5　ノンブロッキング通信性能

負荷をかけると実効バンド幅が1/2になる場合，通信と計算のオーバラップの効果がないことを意味している．したがって，この環境では通信と計算のオーバラップの効果がないと判断できる．

マルチコアに関する通信性能も重要である．図6に，MPI_Allgatherの性能を，4ノード32MPIプロセスで実行した場合と，4ノード4MPIプロセスで実行した場合を示す．通常，1ノード8MPIプロセスで実行する場合には，1ノード1MPIプロセスで実行した場合と比べて扱う配列サイズは8分の1となるため，ここでは，横軸は1MPIプロセスあたりではなく，1ノードあたりのメッセージサイズとした．

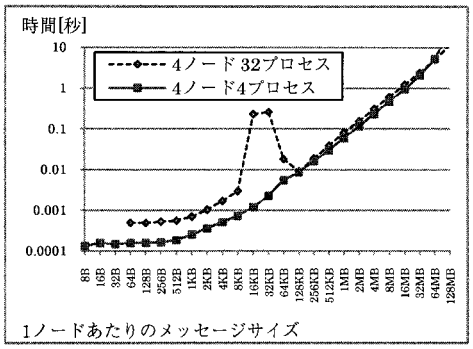

図 6　MPI_Allgatherの性能

図7はメッセージサイズを8 MBとし，計算ノードを2から4まで変えた場合の実行時間である．1ノードを8MPIプロセスで実行したときのオーバーヘッドは，ノード数には関係なく，約0.14秒であった．

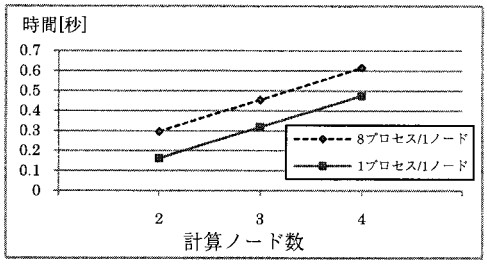

図 7 MPI_Allgatherの性能(各ノード8MB)

図8は，MPI_Allreduceの性能である．データ型はdouble型であるためメッセージサイズを8で割った値が要素数となる．図8から1要素から64要素(512B)まではほぼ同じ時間であることがわかる．このことから処理がまとめられる場合には，まとめてMPI_Allreduceを実行した方が高速になる．実際，CGSによる直交化ではこの利点を利用している．図9は，要素数を32とし，計算ノードを2から4まで変えた場合である．MPI_Allreduceはノード数が2の冪乗でないときに性能が悪くなることが一般的であるが，3ノード24 MPIプロセスの場合にはその影響が比較的小さいことがわかる．

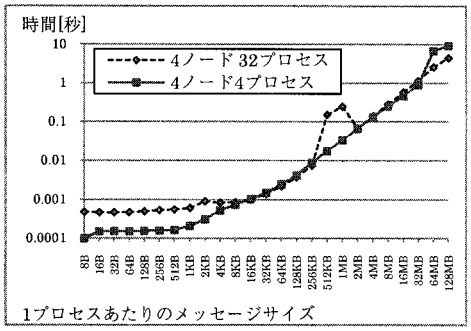

図 8　MPI_Allreduceの性能

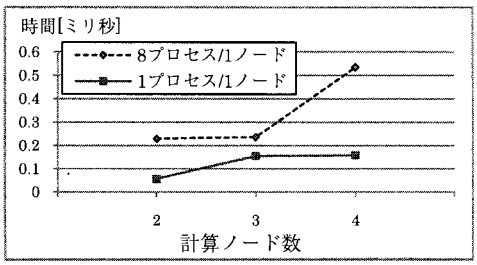

図 9　MPI_Allreduceの性能(32要素)

3.3 実行時自動チューニングの有効性評価

ここでは，実行時自動チューニングの有効性を検証する．ここで，自動チューニングを利用しない場合の標準設定として「ループアンローリングなし，通信方式は1対1ブロッキング通信による実装，前処理なし，直交化はCGS法，リスタート周期は30で固定」とした．そして，それぞれの項目だけを自動チューニングしたもの，全ての項目を自動チューニングしたものを比較した．また，実行時間には自動チューニングに要した時間も含まれている．

図10は5点差分行列の問題である．標準設定のままでは解けなかったため，全ての場合で前処理については自動チューニングの対象とした．1行あたりの非零要素数が少ないためアンローリングの効果は小さかった．また，計算ノード数も少なかったため通信方式の効果はほとんどなかった．リスタート周期の自動化の効果が一番高くなっていることがわかる．全てを自動チューニングの対象としたとき，ノード数が3から4に増えたときに逆に実行時間が増加している．この原因は，通信方式の自動チューニングにかかる時間だけで3秒近くを費やしていたことが原因である．測定項目の一部に極端に性能の悪い部分が含まれていると，全体として低い性能にしかならないこともあり，それに当てはまったようである． MPI の基本通信関数の性能を予め調べておき，ライブラリ実行時にその情報をフィードバックするような仕組みが必要であることがわかる．

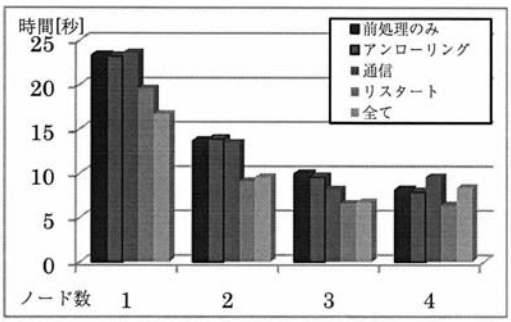

図 10　行列サイズ：160,000
(2次元楕円型偏微分方程式の境界値問題の5点差分式)

図11に，7点差分行列での実行時自動チューニングの結果を示す．この問題も，アンローリングの効果は小さく，前処理の効果が高かった．またノード数が増えるに従い，通信の自動チューニング効果も見えてくる．ここでも，ノード数が3から4に増えたときの実行時間が増大してしまっている．これは，3ノードまでは直交化方式としてMGSが選択されていたものが，4ノードからCGSが選択されたことが主要因となっている．このため，許容残差を満たすために多くの反復回数を要することとなり実行時間が増大した．一方，4ノードで前処理のみを適用した場合には，リスタート周期が30と固定であったことから，結果的に全ての項目を自動チューニング対象とした場合よりも短い時間となった．

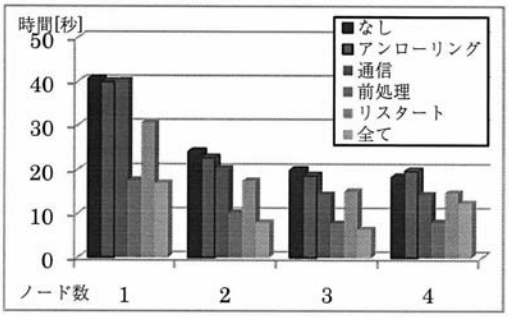

図 11　行列サイズ：512,000
(3次元楕円型偏微分方程式の境界値問題の7点差分式)

GMRES(m)法は残差を最小化していくため反復毎に真の解に近付くことが期待されるが，実際は丸め誤差の影響などで収束しない問題も数多くある．そういう問題の典型的な例を図12に示す．1, 2, ・・は標準設定で実行した場合，1_AT, 2_AT, ・・・は全ての項目に対して自動チューニングを適用した場合で数字はノード数を示す．どの処理でどれくらいの時間がかかっているのかも示した．また，許容残差は0.06のようにかなり大きく設定した．それでも，標準設定では収束しなかったため反復50000回で打ち切った．標準設定で収束しなかったのは，リスタート周期(30に固定)が小さかったことが原因である．自動チューニングを適用することで，リスタート周期として最終的には72～110が選択されている（ノード数によって値が異なる）．

ノード数1のところでは，SpMxVの時間が大半を占めているが，ノード数が2を超えると占める割合が小さくなることがわかる．これは，2ノードで計算することに

よりL2キャッシュメモリにデータが入りきったことを意味する．このようにある一定のノード数からSpMxVにかかる時間が極端に減少することはよくある．また，収束の難しい問題では，リスタート周期が大きくなり，直交化に要する時間がほとんどを占めるようになるといったことがわかる．

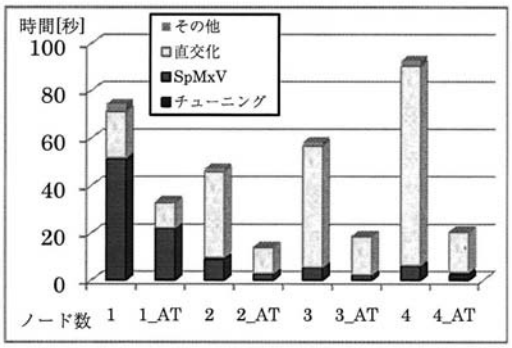

図 12 行列サイズ：16,146 非零要素数：1,015,156 (University of Florida Sparse Matrix Collection： Simon / olafu)

4. 関連研究

数値アルゴリズムレベルではなく，数値計算ライブラリレベルでの実現を考慮した実行時自動チューニング機能の研究は多くの研究が行われているとは言い難い．しかし，疎行列反復解法ソルバにおいてはいくつかの先行研究がある．まず，疎行列反復解法ソルバ用カーネルの自動チューニングを行う米国カリフォルニア大学バークレー校のOSKI[3]が代表的な先行研究例である．OSKIには，ユーザ知識の導入による実行時選択機構が実装されている．また，東京大学のIILBでは，インストール時に代表的なループ長に対するSpMxVアンローリング方式のループ長を推定してチューニングする方法[4]，および，実行時にいくつかのアンローリング段数を変化させ実行時間をチェックしながら解への反復をしていく方法[2]が提案されている．また，[5]では実行時にSpMxVでのブロック幅を変更する実行時自動チューニング方式が提案されている．GRID環境のように冗長実行が許される並列環境では，数値計算ポリシを導入することで，複数実行した直交化方式のうちでポリシに合う方式を実行時に選択する機構を有す数値計算ライブラリの提案[6]がなされている．

5. おわりに

本稿では，Windowsクラスタにおける MS-MPI の基本性能，および実行時自動チューニング機能について，実用的な疎行列反復解法ソルバの観点から評価を行った．性能評価結果として，MS-MPI の性能はマルチコア環境を十分考慮した実装になっているが，一部のメッセージサイズにおいて著しい性能低下があることがわかった．また実行時自動チューニング機能は，PC クラスタにおいても有効性が高いことが示された．今後の課題として，計算ノード数が増えた場合の性能評価が挙げられる．

謝辞 本研究の一部は，マイクロソフト産学連携研究機構(IJARC)「Windows CCS 2003 上の数値計算ライブラリのための MS-MPI の実装方式の自動チューニング」，および，特定領域研究（情報爆発）「情報爆発時代のロバストな自動チューニングシステムに向けた数理的基盤技術の研究」(19024018) の支援による．

参考文献

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

[2] H.Kuroda, T.Katagiri, Y.Kanada: Performance of Automatically Tuned Parallel GMRES(m) Method on Distributed Memory Machines, Proceedings of VECPAR2000, pp.251 - 264 (2000).

[3] R.Vuduc, J.Demmel, and K.Yelick: OSKI: A library of automatically tuned sparse matrix kernels, Proc. of SciDAC 2005, Journal of Physics: Conference Series (2005)

[4] M.Kudoh, H.Kuroda, and Y.Kanada: Parallel Blocked Sparse Matrix-Vector Multiplication with Dynamic Parameter Selection Method, Lecture Notes in Computer Science 2659, pp.581-591 (2003).

[5] 田中輝雄，片桐孝洋，弓場敏嗣：ソフトウェア自動チューニングにおける標本点逐次追加型性能パラメタ推定法の疎行列計算への適用，情報処理学会論文誌：コンピューティングシステム，Vol.48, No. SIG13 (ACS19)，pp. 223-234 (2007).

[6] 直野 健，猪貝 光祥，木立 啓之：数値計算ポリシーを入力とするベクトル群の直交化ライブラリ，情報処理学会論文誌：コンピューティングシステム，Vol.46, No.SIG7(ACS10)，pp. 35-43 (2005).