2008－HPC－118 (6)
2008／12／16

WAN 上の Cluster-of-Clusters による 有限要素解析における反復法ソルバの性能評価

村岡 雅江[1)] 奥田洋司[2)]
[1)]東京大学大学院工学系研究科，[2)]東京大学人工物工学研究センター

WAN 環境における通信速度の向上と低レベルから高レベルにいたるまでの様々なグリッドミドルウェアが充実してきたことによって，計算コストの高いアプリケーションの実行環境としてグリッドを利用することが現実的な手段となってきた．その一方で，有限要素解析など工学分野で広く活用されている数値解析手法の多くは，並列計算の際，頻繁に同期・通信を行う演算が発生するために通信コストの高いグリッド環境を活用した事例が少ない．本研究では，有限要素解析のためのグリッド利用の実用化を図るため，計算および通信コストの高い部分でもある反復法ソルバに注目し，CG 法と GMRES（k）を取り上げる．複数のクラスタマシンからなるグリッド環境において，二つの反復解法の性能を評価し通信頻度やメモリ使用量などの観点からその実用性と利用技術について検討する．

Feasibility study on iterative solvers in FEM using cluster-of-clusters on WAN

MASAE MURAOKA[1)] HIROSHI OKUDA[2)]
[1)]School of Engineering, Univ. of Tokyo, [2)]Research into Artifacts, Center of Engineering, Univ. of Tokyo

With the rapid growth of WAN infrastructure and development of Grid middleware, it's become a realistic methodology to utilize cluster-of-clusters to run computation-demanding applications. However, tightly connected programs requiring frequent synchronization and communication, such as FEM, has rarely been attempted on the Grid environment. We focus on two iterative solvers widely used in FEM: CG method and GMRES(k) method. We report the numerical experiment using cluster-of-clusters on WAN and evaluate the feasibility for iterative solvers.

1. はじめに

近年のネットワークの性能向上とグリッドミドルウェアの充実により，組織を超え，さまざまな情報・計算資源の共有が安全かつ効率的に実現されるようになった．ローカルな環境では実現できなかった大規模な計算もグリッド技術の利用によって地理的，組織的に分散する計算資源を，ネットワークを介して統合的に利用することで実現可能となる [1]．なかでも，インターネットを介して結ばれる廉価で高性能な PC クラスタ同士がつながった計算グリッドはコストパフォーマンスの高い現実的な形といえる．

また，MPICH-G2 [2] や Grid MPI [3] などグリッド環境に拡張された MPI (Message Passing Interface) の実装系が提供され，既存の MPI プログラムが修正なくグリッド上で実行できる．これはアプリケーション利用者にとって大変魅力的な点である．しかしながら，アーキテクチャ，ハードウェア性能，管理主体が異なる計算機で構成され，未知な演算性能や動的に変化しうるネットワーク性能の下では，プロセス間の依存度の低いパラメータ探索やモンテカルロ計算などは応用例が多く高性能化・大規模化で成果を挙げている [4,5]．その方で，並列有限要素法のように頻繁に同期および通信を行う並列計算は通信コストの高いグリッド環境が活用される事例は少ない [6]．本研究は，有限要素解析のためのグリッド利用の実用化を目的とし，解析中のもっとも計算および通信コストの高い部分でもある反復法ソルバのためのグリッド利用について検討する．本研究では特に GMRES(k)法 [7] について，メモリ使用量に影響するリスタート周期の設定と収束性に関する性質に注目し，メモリ資源の拡張性に優れるグリッド環境の効果的な利用について検討する．GMRES(k)法は，直交ベクトルの算出部分で高頻度な集団通信が発生するため，グリッド環境では著しい性能劣化が懸念される．それに対して有限要素解析でもっともよく使われる CG 法は反復計算を通してメモリ使用量と通信頻度は一定であり，グリッド環境における CG 法の性能と比較しながら GMRES(k)法のグリッド実用性を評価する．さらに，集団通信の頻度が高い Gram-Schmidt 法の部分について利用した GMRES(k)法のプログラムでは Modified Gram-Schmidt 法のアルゴリズムが利用されているが Classical Gram-Schmidt 法への変更を行うことで集団通信の頻度を下げ，グリッド環境における性能向上について検討する．

2. グリッド環境における並列有限要素解析

われわれの実験環境である Cluster-of-Clusters 環境（以下 C-of-C と略す）はアーキテクチャ，ネットワーク性能，CPU 性能，メモリ容量，セキュリティポリシーなどさまざまな面で異質なクラスタから構成される．各々のクラスタは高性能な CPU がギガビットネットワークで接続されており有限要素解析のような密結合並列プログラムはクラスタ単体で実行するのが当然とされ，グリッドアプリケーションとしてはパラメータ検索のようなプロセス間の独立性が高いものやマルチフィジクス，マルチ スケールといった連成問題のような疎結合アプリケーションが主流である．有限要素解析も，例えば解析と解析結果の可視化といった異なる処理が組み合わさった疎結合アプリケーションという枠組みの中での実用化が図られる一方で，メタコンピューティング的な使われ方は少ない．しかしながら，より現実的で精密なモデルを扱うことが求められる計算工学の分野において，ローカルの計算環境では計算資源が不足するといったような大規模問題に対し，グリッドの利用は一つの魅力的な解決策である．本研究では，ある程度の並列化効率を損なったとしても，問題の大規模化における計算資源の枯渇を解消する手段としてグリッドの利用価値を第一に置く．利用する有限要素解析プログラムは，大規模並列分散環境における高性能な構造解析ソフトウェア FrontSTR [8] である．本研究ではその中で解かれる行列方程式を例題として用いるため，並列化にともなう通信パターンなどは有限要素法における並列化手法である領域分割法に基づく．

3. 有限要素解析における反復法ソルバ

3.1 通信パターン

一般的に，有限要素解析の並列化は領域分割に基づき，各プロセスは割り当てられた部分領域における節点と要素に関する剛性行列および右辺ベクトルを計算し，保持する．したがって，反復法のアルゴリズムで繰り返し行われる内積や行列ベクトル積では通信が必要となる．本研究で用いる反復法ソルバのプログラムにおいて，まず行列ベクトル積では，割り当てられた部分領域が隣接するプロセス間において共有節点に関するベクトルの成分が送受信された後に演算が行われる．この通信は，MPI_Isend MPI_Irecv MPI_Waitall （以下 Isend/Irecv と呼ぶ）で実装され，隣接するプロセスすべてと行われ，データサイズは倍精度演算であれば共有節点数×8 [B]となる．解析するモデルの規模や形状にもよるが，大体 KB から MB 規模の通信データとなる．次に内積計算については全プロセスで同期する集団通信によって各プロセスで計算された演算結果の和をとるが，これは MPI_Allreudce （以下 Allreduce 略す）で実装され，データサイズはスカラーのデータ 1 つなので 8 [B]と固定される．

さらに，CG 法と GMRES(k)法における通信頻度をより詳しく見ていくと，まず CG 法については，一反復あたり内積計算が 3 箇所，行列ベクトル積が 1 箇所ある．N を全反復回数とすると Allreduce の全回数は，

$$3N \quad [回]$$

となり，Isend/Irecv は，

$$N \times NEIBPE \quad [回]$$

となる (ここで，$NEIBPE$ は隣接プロセス数である)．

次に，GMRES(k)法では，正規直交基底を毎ステップで求めるが，その演算アルゴリズムである Gram-Schmidt の直交化 (図 1) について注目すれば，m ステップ（$m < k$）において $m+1$ [回]の内積計算と 1 回の行列ベクトル積が行われる．従って，内積計算は一リスタート周期当たりだけでも

$$\sum_{m=1}^{k}(m+1) = \frac{1}{2}(k^2+3k) \quad [回]$$

となり，M 回リスタートを回繰り返すとすると，

$$\frac{M}{2}(k^2+3k) \quad [回]$$

となる．一方，Isend/Irecv は一反復当たりの頻度が CG 法と同程度で全反復回数にわたっては

$$Mk \times NEIBPE \quad [回]$$

となる．したがって両解法を比べると Allreduce の頻度が大きく異なり，GMRES(k)法では非常に高頻度な同期が発生する．

$r = b - Ax_0$, *for a given initial guess* x_0
$v_1 = r/\|r\|$
for $m = 1,\ldots,k$
  $\tilde{v}_{m+1} = Av_m$
  for $i = 1,\ldots,m$
    $h_{i,m} = v_i^T \tilde{v}_{m+1}$
    $\tilde{v}_{m+1} = \tilde{v}_{m+1} - h_{i,m} v_i$
  end
  $v_{m+1} = \tilde{v}_{m+1}/\|\tilde{v}_{m+1}\|$
  $h_{m+1,m} = \|\tilde{v}_{m+1}\|$
end

Figure 1 Construction of basis vectors by Gram-Schmidt Process.

3.2 GMRES 法のリスタート周期と収束性

並列化効率を損ねるとしてもメモリ資源の拡充が容易にできることをグリッド環境の最大の利点として活用する上で GMRES(k)法のメモリ使用量に直接影響す

るリスタート周期と収束性の関係について整理する必要がある．GMRES 法は一反復当たりの計算量とメモリ使用量が増加してゆくという特徴をもつために適当なステップ毎にリスタートする GMRES(k)法が一般的だが，リスタート周期が短すぎると収束するのに膨大な反復を要したり，残差が停滞したりといった恐れがある[1]．GMRES 法は必要とする計算時間と記憶容量が反復回数にともなって増加してしまうため，通常はリスタートを行う GMRES(k)法が使われるが，リスタート周期 k を短くしすぎると反復回数が膨大になったり，残差ノルムが停滞してしまったりする．一方，リスタート周期を長くとれば，条件数の大きな係数行列の場合にも収束する可能性は高くなるが，アルゴリズム中にある正規直交基底の算出部分において周期内で生成したすべての基底ベクトルとの直交化をおこなうため必要とする記憶容量および演算量が増加する．適当なリスタート周期をいかに設定するかについては，係数行列の固有値の分布や条件数，リッツ値などを用いた数理的なアプローチや自動チューニングの対象として活発な議論がなされている[9,10,11]．しかしながら，そうした研究の多くで取り上げられる例題は，リスタート周期が大きくても数十～数百のものがほとんどであり，大きくても単体のクラスタマシンや並列計算機による実験環境が中心となっている．ここで，非対称行列を係数行列にもつ行列方程式を例題として，リスタート周期を500から4000までに変化させたときの収束履歴，最大メモリ使用率，計算時間を調査した結果を示す．例題は図2に示すような二輪車のホイールのモデル（約94000節点，約48000要素，四面体二次要素）を用いた線形弾性静解析で得られた行列方程式を，図3で示すような行操作を行うことで係数行列に数値的な非対称性を与えた行列方程式である．

Figure 2　Model of motor cycle wheel partitioned into 16 domains

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \Rightarrow \begin{bmatrix} 2a_{11} & 2a_{12} & 2a_{13} & 2a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

Figure 3　Row operation to make linear system numerically unsymmetric.

ここでの実行環境は後述する C-of-C 環境でもつかわれるクラスタのうちの一台で東京大学柏キャンパスにある SC と呼ばれる PC クラスタである．Intel Pentium4 CPU 3.00GHz, Memory 2GB の 16 ノードを利用した．収束判定は残差ノルムについて10^{-8}を条件とし，本例題を CG 法および GMRES(k)法の両方でといた場合の収束履歴を図4にメモリ使用率と計算時間については図5に示す．図4の結果よりリスタート周期を増やすほど短い反復回数で収束することがわかる．その一方でメモリ使用量については線形に増加，そして計算時間については k=1000 で最小となるという結果が得られた．一周期当たりの計算量はリスタート周期 k の二乗の関数として増加するが，リスタート周期が増加すると全反復回数が減るために本問題では k=1000 のときに収束までにかかる全計算量が最小となった．従って，グリッドを利用するからといって闇雲にリスタート周期を長くすればよいということはなく，今回のように最適なリスタート周期においてもなおローカルなリソースや単体の計算機では不足するという場合でのグリッド利用が有効と考えられる．

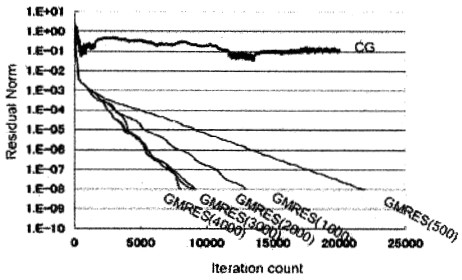

Figure 4　Convergence behavior for different restart cycle

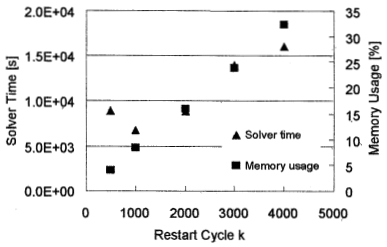

Figure 5　Memory usage and solver time vs. restart cycle

4. クラスタ間での反復法ソルバの実行

4.1 実験環境

実験環境はインターネットで結ばれる2台のクラスタで構成されるC-of-C環境である．一台は3章の数値実験でも利用したSCと呼ばれる東大柏キャンパスにあるPCクラスタで，もう一台はつくば市の産業技術総合研究所にあるF32とよばれるクラスタである．2台のクラスタの構成について詳細は表1に示す．地理的な距離は約30Kmでネットワーク上の距離としては約10箇所のルータを介し，2-3ms程度の遅延時間が観測される．SCクラスタについてフロントエンドの一台のみがGatewayマシンとして外部ネットワークとのインターフェースをもち，ローカルネットーワークにある全計算ノードのパケットはこのGatewayマシンを介する設定となっている．実行する並列有限要素解析プログラムはMPIプログラムであり，MPIプロセスはプログラム実行中互いにコネクションを確立し保持しつづける．したがって，プライベートアドレスの計算ノードに対してはGatewayマシンにおいて1対1NATのサービスが行われている．通信性能の点からするとこのような設定は非効率的といえるが，ローカルで使われていた既存の運用形態になるべく手を加えずにグリッド環境で利用したいという運用上のポリシによるものであり，あくまでもグリッド環境に傾倒した形態にはなっていない．また，MPIのプログラムの実行にはGridMPIを利用した．GridMPIは地理的に分散する複数のクラスタマシンを統合した複合クラスタ環境をはじめとするグリッド環境におけるHPCを目的として設計されたMPIの実装系である．

Table 1 Specification of two clusters

	SC	F32
CPU	Intel® Pentium® 4 3.00GHz	Intel® Xeon™ 3.06GHz
Memory	2GB/node	4GB/node
Network	Gigabit Ether	Gigabit Ether
OS	DebianGNU/Linux3.1	RedHatLinux8.0
No. of nodes	16	64
PE / node	1	2

4.2 実験条件

線形弾性静解析において得られる行列方程式をCG法とGMRES法の2種類の解法で解くとする．このとき，領域分割法に基づく並列化によりプロセス数を2, 4, 8, 16, 32として2台のクラスタでプロセスを分配して実行し計算時間と通信時間，および(1)式で算出されるワークレシオについて計測する．プロセスの分配方法としては，各クラスタに同数のプロセスを割り当てるとする．たとえば16プロセスの並列計算ではSC，F32ともに8プロセスずつ割り当てて実行する．

$$work\ ratio = \frac{solver\ time - comm.\ time}{solver\ time}\ [\%] \qquad (1)$$

4.3 テストモデル

線形弾性静解析の問題として次の3つのモデルを用い，そこで得られた行列方程式を例題とした．

(1) **ホイール**

3章で利用した二輪車のホイールモデル（約94000節点，約48000要素，四面体二次要素）．

(2) **エンジン**

二輪車のエンジンモデル（約580000節点，約330000要素，四面体二次）

(3) **フレーム**

二輪車のフレームモデル（約520000節点，約270000要素，四面体二次）

ただし，ホイールモデルについては規模が小さく，最大のプロセス数を１６とする．

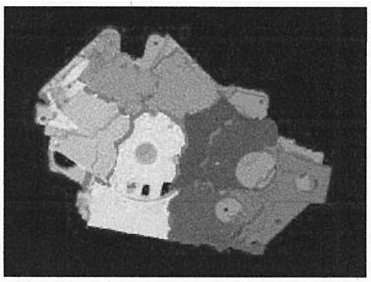

Figure 6 Model of engine partitioned into 16 PE

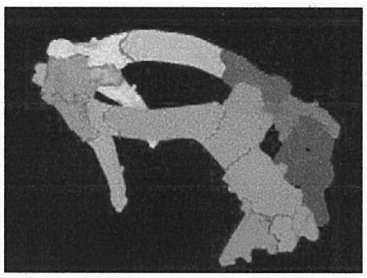

Figure 7 Model of frame partitioned into 16 PE

4.4 CG法の実行

CG法で実行した場合の結果をモデルごとに図8,9,10で示す．収束に要した反復回数はホイールが約6600回，エンジンが約16000回，フレームが約22000回であった．サイトをまたいた並列計算となるため，通信時間が非常に大きくなると予想したが，ホイール

ではプロセス数が 2，4程度までであればワークレシオが90%近く保たれ，規模が大きいエンジンではプロセス数が8，フレームでは16まで約90%を保っている．エンジンにおいてはプロセス数が 32 になったところでワークレシオの低下が顕著に見られたが，集団通信の頻度がCG 法程度（3 回 / 反復）であれば，全計算時間への割合で見ればサイト間であっても大きな影響が出ないことがわかった．

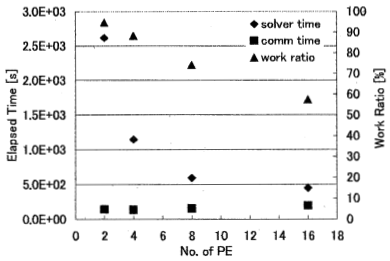

Figure 8　CG method on C-of-C using wheel model

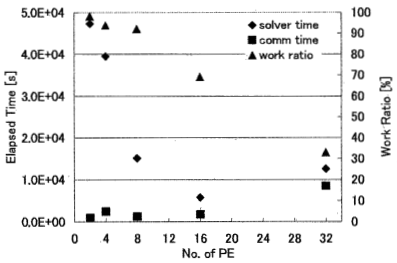

Figure 9　CG method on C-of-C using engine model

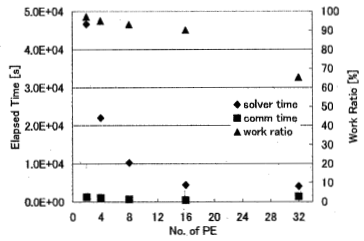

Figure 10　CG method on C-of-C using frame model

4.5　GMRES(k)法の実行

つぎに，GMRES(k)法による結果を図 11 に示す．ここではホイールのモデルを用いた結果のみを示す．また，3 章のリスタート周期に関する実験より，リスタート周期は k=1000 に固定した．反復回数は約 12000 回であった．3.1 節で述べたように，GMRES(k)法ではCG 法に比べ，集団通信の頻度が非常に高くなるため，プロセス数を増やすに従い計算時間の中で通信時間が支配的になっていく様子が顕著に現れる．グリッドがメモリの拡張性に優れるとはいえ，このような通信コストでは実用性が乏しいといえる．

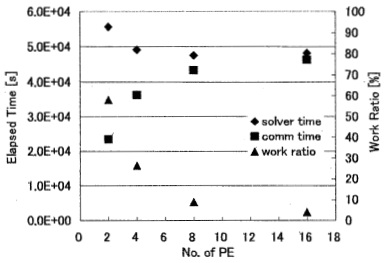

Figure 11　GMRES on C-of-C using wheel model

5.　Gram-Schmidtの直交化における通信コスト削減と計算精度

GMRES の正規直交基底を算出する箇所において，例えばCG 法程度まで集団通信の通信頻度を下げることができれば，CG 法に近い実行性能が期待される．現状の実装では，図 12 で示すModified Gram-Schmidt 法（以下MGS）のアルゴリズムが用いられている．これを図 13 に示すように実装しなおすことで，

$(k^2+3k)/2$ [回]のAllreduceの回数を$M \times k$ [回]に減らすことができる．図13 はClassical Gram-Schmidt 法（CGS）のアルゴリズムと解釈され，両者は解析的には同値である．しかしながら有限精度演算においてCGS では直交性の計算精度が低いため一般的には並列効率よりも精度を優先し MGS での実装が行われる[12]．図 14 に CGS への変更を施した GMRES(k)法をC-of-C 環境で実行した場合の計測結果を示す．反復回数は約42000 回でCGS に変更したことにより約3.5 倍となった．しかしながら反復回数が増えて計算量が増加したにもかかわらず通信コストが圧縮されることにより，図 11 の計算時間より短縮されていることがわかる．

At m^{th} step $(0 < m \le k)$

calc　$w = Av_m$

do　$i = 1, \ldots, m$

　　$val\_0 = (w, v_i)$

　　call MPI_Allreduce(*val_0, val*, 1,
　　MPI_SUM, COMM_WORLD)

　　$w = w - val \times v_i$

enddo

$v_{m+1} = w / \|w\|$

Figure 12　Current implementation
of Modified Gram-Schmidt in FrontSTR

```
At m^th step (0<m≤k)
allocate val_0(1:m), val(1:m)

calc w = Av_m

do i = 1, ..., m
      val_0(i) = (v_i, w)
enddo

call MPI_Allreduce(val_0, val, m,
                   MPI_SUM, COMM_WORLD)

do i = 1, ..., m
      w = w − val(i) × v_i
enddo
v_{m+1} = w / ||w||
deallocate val_0, val
```

Figure 13 Implementation of Classical Gram-Schmidt

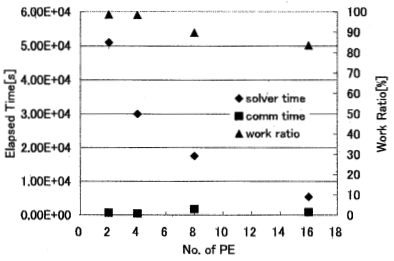

Figure 14 GMRES with CGS on C-of-C using wheel model

6. 結論と今後の課題

グリッド環境における有限要素解析のための反復法ソルバの性能評価を行った．利用した反復法ソルバはグリッドの様なヘテロ性を意識せずに開発されたものだが，CG 法程度の通信コストであれば適当なプロセス数の下で良好なワークレシオが得られることが確認された．その一方で，プロセス数を大きくすると通信コストの影響大きくなり，特にエンジンの例題においてそれが顕著に見られた．一方，GMRES(k)法については，メモリ資源の拡張性に優れるグリッド環境を効果的に利用できると期待される．ただし，リスタートと収束性および計算コストに関する実験より，闇雲にリスタート周期を長くしても計算時間の増加につながるため，最適なリスタート周期においてもなお，ローカルなリソースや単体の計算機では不足するという場合でのグリッド利用が有効と考えられる．例題に対し適当なリスタート周期下でのサイト間の性能評価を行ったが，非常に高い集団通信を行ため実用的な性能は得られなかった．正規直交基底の算出部分のアルゴリズムを MGS から CGS に変更することで通信コストの削減が図られたが，反復回数の増加など直交性の計算精度における課題が残りさらなるアルゴリズムの改善が必要である．また，本研究の結果を踏まえ，さらなる大規模モデルによる数十，数百プロセスといった大規模並列有限要素解析におけるグリッド環境の実用性評価が必要不可欠である．

謝辞 本研究は ApGrid の研究活動の一環として行われた．計算資源を利用させていただいた(独)産業技術総合研究所グリッド研究センターをはじめ ApGrid の研究活動に参加される皆様に謝意を記す．

参考文献

1) I. Foster, C. Kesselman, Computational Grids, The Grid: Blueprint for a New Computing Infrastructure, Morgan-Kaufman (1999).
2) N. Karonis, B. Toonen, and I. Foster, MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface, Journal of Parallel and Distributed Computing, Vol. 63, No. 5, pp. 551-563 (2003).
3) 松田,元彦; 石川,裕; 鐘尾,宜隆; 枝元,真彦; 岡崎,史裕; 鯉江,英隆; 高野,了成, GridMPITM Version 1.0 の概要, 情報処理学会研究報告, 2005-HPC-103-(29), pp. 169-174 (2005).
4) 三村 泰成, 吉村 忍, 河合 浩志, 廣安 知之, 下坂 久司, グローバルコンピューティング環境における実用構造機器の並列最適設計, 最適化シンポジウム講演論文集, Vol.2002, No.5, pp. 363-368 (2002).
5) 池上,努; 武宮,博; 長嶋,雲兵; 田中,良夫; 関口,智嗣, Grid：広域分散並列処理環境での高精度分子シミュレーション, 情報処理学会論文誌 コンピューティングシステム, Vol.44, No.SIG11(ACS), pp.14-22, (2003).
6) 村岡雅江; 奥田洋司, CLUSTER-OF-CLUSTERS 環境における並列有限要素解析の実用性評価, 日本機械学會論文集. A 編, Vol. 73, No. 733, pp981-988 (2007).
7) Y. Saad and M.H.Schultz, GMRES:A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comp.,Vol.7, No.3, pp.856-869 (1986).
8) 寺坂 晴夫, 陳 錦洋, 栗原 正道, 革新ソフトウェアに対する統合解析システムの開発, 計算力学講演会講演論文集, Vol.2006, No.19, pp. 447-448 (2006).
9) 羽部 充, 野寺 隆, リスタート周期を動的に変えるGMRES(m)法, 情報処理学会論文誌, Vol.43, No.6, pp.1795-1803 (2002).
10) 黒田 久泰, 金田 康正, 自動チューニング機能付き並列疎行列連立一次方程式ソルバの性能, 情報処理学会研究報告, 99-HPC-76-3, pp. 13-18 (1999).
11) 張 臨傑、野寺 隆, Ritz 値を考慮した GMRES(m)法の適応的なリスタート, 情報処理学会論文誌コンピューティングシステム, Vol.45, No.SIG11(ACS), P.1-10 (2004).
12) 鈴木 洋夫, 森屋 健太郎, 野寺 隆, 並列ブロックグラムシュミット法を用いた Deflated-GMRES(m)法の一考察, 数理解析研究所講究録, Vol.1198,pp. 101-107.