2G-03

時系列シミュレーションのためのSMW公式を用いた並列ソフトウェアパイプラインの高速化

柴田直樹（福井大学工学研究科）・柿本隆介（福井大学工学部）
・福間　慎治（福井大学工学研究科）・森眞一郎（福井大学工学研究科）

1　はじめに

本研究では，多くのシミュレーション中に現れる連立一次方程式 $\boldsymbol{Ax} = \boldsymbol{b}$ の求解問題に着目し，シミュレーション中のユーザからのインタラクションによって係数行列 $\boldsymbol{A}$ の一部の要素のみが変化をするインタラクティブシミュレーションを想定し，係数行列が変化するタイムステップと係数行列が変化しない複数のタイムステップが交互に発生するモデルを仮定する．係数行列が変化するステップから次に係数行列が変化するステップまでの間をステージと呼び，i ステージ目の係数行列 $\boldsymbol{A_i}$ が $\boldsymbol{A_i} \approx \boldsymbol{A_{i+1}}$ となる性質を持つものとする．また，対象となる問題サイズは直接法では実時間性が保証できず，反復法でも並列化効果が期待できない程度とする．本研究が想定するインタラクティブシミュレーションにおいては，動作の違和感をユーザーに感じさせないために 33〜100msec 程度の応答速度が必要とされる．

数学公式の 1 つに，SMW 公式 (Sherman-Morrison-Woodbury formula)[3] がある．いま,$\boldsymbol{A}$ を $n \times n$, $\boldsymbol{\Delta A_c}$ を $n \times s$, $\boldsymbol{E_c}$ を $s \times n$ の大きさの行列で, $\boldsymbol{A}$ は正則であるとする.$\boldsymbol{A}$ の近似行列 $\boldsymbol{A'}$ を $\boldsymbol{A'} = \boldsymbol{A} + \boldsymbol{\Delta A}$ と表現し, さらに $\boldsymbol{\Delta A} = \boldsymbol{\Delta A_c} \times \boldsymbol{E_c}$ とすると SMW 公式より式 1 が求まる.

$$(\boldsymbol{A} + \boldsymbol{\Delta A})^{-1} = \boldsymbol{A^{-1}} - \boldsymbol{A^{-1}\Delta A_c}(\boldsymbol{I} + \boldsymbol{E_c A^{-1} \Delta A_c})^{-1}\boldsymbol{E_c A^{-1}} \quad (1)$$

この式は,$\boldsymbol{A^{-1}}$ を用いて補正により $(\boldsymbol{A^{-1}})'$ が求まる可能性を示唆している．$(\boldsymbol{I} + \boldsymbol{A_c A^{-1} \Delta A_c})^{-1}$ の計算に $O(s^3)$ の時間がかかるが,$n >> s$ かつ s を定数とみなせる場合, 全体としては $O(n^2)$ となる．SMW 公式での計算行程では行列積計算が多くの割合を占め, 並列処理による高速化が得られる．

我々はこの公式を用いた並列実行による時系列シミュレーションの高速化を実現した．この公式を用いることにより，係数行列 $\boldsymbol{A}$ の近似行列である $\boldsymbol{A'}$ の逆行列 $(\boldsymbol{A'})^{-1}$ が既知である前提のもと，変化後の係数行列は $(\boldsymbol{A} + \boldsymbol{\Delta A})^{-1} = \boldsymbol{A^{-1}} -$（補正項）で求めることができる．このとき，$\boldsymbol{A}$ と $\boldsymbol{A'}$ の差分行列で非零要素数を持つ行の数を $\boldsymbol{s}(\boldsymbol{s} \ll \boldsymbol{n})$ とすると，SMW 公式の計算量は $\boldsymbol{O(sn^2)}$ となる．この公式はほとんどが行列積演算であり大半を並列化することができる．反復法においても部分的に並列化可能ではあるが，本研究が対象とする行列サイズにおいては，並列化効果に疑問が残る．そこで本研究では，係数行列 $\boldsymbol{A}$ の近似行列である $\boldsymbol{A'}$ の逆行列 $(\boldsymbol{A'})^{-1}$ が既知である前提のもと，SMW 公式を用いた計算を行う．

2　研究背景

2.1　並列ソフトウェアパイプライン

SMW 公式は本来逆行列を求める際に使われる公式であるが、最終目的が $\boldsymbol{Ax} = \boldsymbol{b}$ の解を求めることであれば先に SMW 公式の両辺に右からベクトルをかけることで逆行列そのものの導出を省略し直接 $\boldsymbol{x}$ を求めることが可能である。この場合行列の計算量は逆行列を求めてからベクトルを乗算する場合と比べて激減し、高速な解計算が可能となる。本論においてこれを直接計算法と呼ぶ.

直接計算法では非常に高速な解の導出が可能である。ところが、直接計算法に限らず SMW 公式では, 計算量が差分 S に依存しており、係数行列 $\boldsymbol{A}$ とその変化後の $\boldsymbol{A'}$ との差分行列のサイズ増加に伴い計算時間が増加してしまう. そのため, 時間経過とともに計算時間が激増する問題が発生する. そこで, 複数の計算ノードを用いて, 直接計算による解計算と並行して, 通常の SMW 公式で逆行列を求め, 直接計算法で用いる基準となる逆行列に補正をかけていく並列ソフトウェアパイプラインの実装を行ってきた.[2] このシステムでは SMW 公式による逆行列計算と SMW 公式にベクトルをかけた直接法による解計算が繰り返し行われる。図 1 は逆行列計算に 2 ステージ相当の時間を要する状況だが、並列パイプライン処理により実時間処理を実現している。この図で $\boldsymbol{xi}$ と記している部分ではステージが変わるまでステップごとに解計算が複数回行われている。また、逆行列計算に必要な時間が変動した際にソフトウェアパイプラインの構造を動的に変動させるフレームワークも既に研究されてきた。

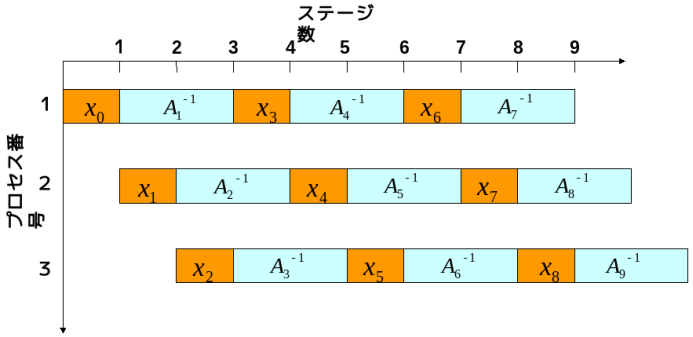

図 1: 並列ソフトウェアパイプライン

2.2　逆行列計算のハイブリッド並列処理

SMW 公式を用いた逆行列計算は並列化が容易であり、近年のマルチコアプロセッサによる恩恵を受けやすい (図??)．しかしながら、問題サイズがある程度大きくなるとノード内メモリ帯域の制約から来る性能向上限界に到達してしまう。この限界を越えるには異なるノードのメモリ帯域を集約しデータ通信能力を向上させるハイブリッド並列処理が必要となる。[1] 一方でハイブリッド並列処理では新たに発生するノード間通信によるオーバーヘッドの問題を解決しなければならない。その解決法のひとつが通信と計算のオーバーラップ実行による通信隠蔽である (図??)。並列化により必要な通信が存在するが、通信が完了する前にも実行可能な計算は存在する。そこで通信をサブスレッド上で行うことで通信をしながら計算を行い、通信を隠蔽する。ただし，アルゴリズムの変更が必要となるため，一部計算順序の入れかえを行う．

3　並列ソフトウェアパイプラインのハイブリッド並列処理

これまで我々は SMW 公式を用いた並列ソフトウェアパイプラインや逆行列の並列演算の実装を行ってきた．本研究では更なる高精度インタラクティブシミュレーションの実現を目指してこれらの技術を統合し更なる実行時間の短縮を図る．図 2 は逆行列計算並びに直接計算法を用いた解計算を 2 台のノードを用いてハイブリッド並列処理した場合のパイプライン構成である。逆行列演算におけるアルゴリズムは従来研究

とほぼ同じであるが、MPI コミュニケータを用いてグループ化を行うことで MPI による局所的同期や関数の使用を可能にしている。解計算部分においても基本的には逆行列計算部分と同様のハイブリッド並列処理を行うが、計算量と通信量のバランスが逆行列計算とは異なるためその点を考慮した実装が必要である。また、シミュレーションシステムとして全体を考えた場合、計算部分の高速化に伴いこれまで相対的に無視できていた管理ノードと計算ノードの通信オーバーヘッドを軽減する必要が発生した。

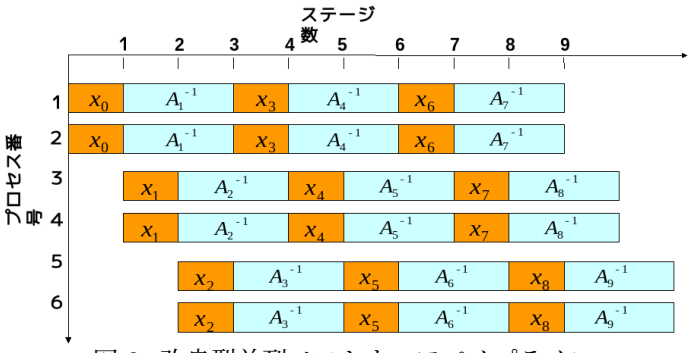

図 2: 改良型並列ソフトウェアパイプライン

3.1 解計算の高速化

直接計算法による解計算だが、更なる高精度シミュレーションのためにも解計算の高速化を求めた際、直接法による解計算の高速化は重要である。逆行列計算での実装と同様の実装を解計算に行うことは可能である。ただし、解計算においてはベクトルをかけていることにより計算量が減少しており、並列化を実装することは通信量を増やすことにつながるため効果が期待できない。そこで逆行列計算で用いられた技術を応用し一部のみを並列化、オーバーラップ実行できない計算を増やす代わりに通信量を削減する手法をとる。$\boldsymbol{Ec}$ との積を求めていた箇所ではベクトル $\boldsymbol{B}$ が追加されているため同じ手法でもベクトルを用いて実装することで通信量の削減が実現した。今後は通信後行う計算をさらに細かく分割することで通信量と計算量を減らし、オーバーラップ実行が可能な部分を増やすことも考えられる。

3.2 管理ノードとの通信削減

図 1,2 では省略されているが、本システムでは管理ノードが存在しそのノードがインターフェースに接続される仮定の下で実装されている。そのため、管理ノードからパイプライン処理を行う特定ノードに対し入力であるベクトル $\boldsymbol{B}$ や差分行列 $\boldsymbol{Ac}$ および $\boldsymbol{Ec}$ を分配する通信が必須である。ただし、ベクトル $\boldsymbol{B}$ については $\boldsymbol{n}$ のサイズのベクトルであり、$\boldsymbol{Ac}$ および $\boldsymbol{Ec}$ は $\boldsymbol{n} * \boldsymbol{n}$ のサイズの行列であることから通信の大半はこの $\boldsymbol{Ac}$ と $\boldsymbol{Ec}$ であることがわかる。そこでこの $\boldsymbol{Ac}$ と $\boldsymbol{Ec}$ を情報圧縮することで通信を削減する。情報圧縮については既にいくつも技術が存在するが、その一つに CRS 形式への圧縮がある。しかしながら今回 CRS 形式を適用した場合 1 行に N 要素が残り、$\boldsymbol{O(n)}$ が残ることは可能な限り避けたい。ここで $\boldsymbol{Ac}$ と $\boldsymbol{Ec}$ の要素の大半は零要素であり、$\boldsymbol{Ec}$ の非零要素はすべて 1 である。また、非零要素の数は $\boldsymbol{s}$ に依存する。これを利用し、$\boldsymbol{Ec}$ については非零要素の座標のみの情報、$\boldsymbol{Ac}$ については座標と値のみの情報に変換することで行列を $\boldsymbol{O(s)}$ で大幅に圧縮することを実現した (図 3)

3.3 実験：ハイブリッド並列化による影響の評価

並列パイプラインをハイブリッド並列化したことによる評価を行うため実験を行った。同環境でそれぞれの解計算、逆行列計算にかかる時間を計測し、比較する。実験結果を表 1 に示す．実験には肝臓を有限要素モデル化したものを係数行列として用いた．行列のサイズは $\boldsymbol{3759 \times 3759}$ で，非零要素数の割合は約 1.0[%] であり，係数行列の変化サイズ $\boldsymbol{s}$ は 15 から 45 の間を変動する場合とする．実験環境は CPU:Core2Quad

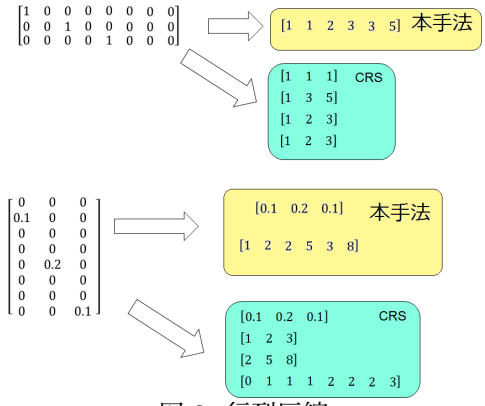

図 3: 行列圧縮

2.83[GHz]，L2Cache:12[MB]，メモリ:2[GB]，メモリ帯域幅:10.6[GB/s]，OS:Linux2.6.191.2895.fc6，コンパイラ:Intel icc12.0.0，MPI:IntelMPI，Network:Gigabit Ethernet を用いる。従来手法では管理ノードとの通信時間を分けて計測した。本手法ではオーバーラップにより分離しての計測は不可能であることから一つにまとめて計測を行った。並列度 1 の状態では情報圧縮による通信時間は大きく軽減しているものの計算時間に変動は無い。解計算は並列度が 2 で少し早くなり、4 で遅くなる。これは解計算が逆行列経産に比べて通信オーバーヘッドの影響を強く受けるためであると考えられる。それに対し逆行列計算については並列度をあげたことにより計算が高速化していることがわかる。逆行列計算は解計算に比べ計算量が多く、通信オーバーヘッドによる影響よりも並列化によって計算量が軽減したことによる影響が大きかったためであると考えられる．実時間性を保つためには解計算は 30msec 以内に収める必要があるが、4 並列でも要求時間内での動作に余裕があり、逆行列計算時間を向上させることができた。

表 1: 実験 1 結果 [msec]

並列度	従来法 解計算	従来手法 逆行列計算	従来手法 計算情報通信時間	本手法 解計算+通信	本手法 逆行列計算+通信
1	9.25	58.80	16.73	12.4	60.4
2	-	-	-	10.3	45.3
4	-	-	-	12.1	41.9

4 まとめと今後の課題

並列ソフトウェアパイプラインとハイブリット並列処理を統合することにより逆行列計算及び解計算にかかる時間を短縮した。また、情報圧縮により通信オーバーヘッドを大幅に軽減した。今後は行列積を細かく分解することにより計算量と通信量を削減するとともに差分 $\boldsymbol{s}$ のサイズが変動することによるシステムへの影響を検証する．

謝辞

本研究の一部は科研費 (25280042) の助成を受けて実施した。

参考文献

[1] 松井祐太，福間慎治，森眞一郎：実時間シミュレーションへの応用を前提とした SMW 公式を用いた逆行列計算のハイブリッド並列処理の評価：情報処理学会論文誌 Vol.53 No.10 1—8 (Oct. 2012)

[2] 柿本隆介：時系列シミュレーションのための SMW 公式を用いた線形計算の並列ソフトウェアパイプラインの柔構造化，福井大学工学部卒業論文 (2012)

[3] G.H. Golub and C.F.Van Loan: Matrix Computations: 3rd ed., Johns Hopkins Univ. Press, 1996.

[4] Avoiding communication in sparse matrix computations：James Demmel, Mark Hoemmen, Mar ghoob Mohiyuddin, and Katherine Y elick：Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on