楽観的な並列離散事象シミュレーションの効率化
―自動生成可逆プログラムの比較―

2018SE034 児玉春司
指導教員：横山哲郎

1　はじめに

離散的なモデルの並列シミュレーションは，大規模な通信システム・計算システムの設計・予備評価などに使用されている．シミュレーションモデルが大規模化するにつれて，実装の効率化やシミュレーション時のオーバヘッド削減へ向けた研究がなされている．

[1]．では，モデルの大規模化に対して手書きでプログラムを作成することは持続不可能であることを指摘している．この問題に対して，C++ 言語で書かれたプログラムを自動で変換するライブラリが提案されている．しかし，提案されたライブラリは C++ 言語全体を使用できるように実装されたため，オーバヘッドの改善が可能であることがすでに述べられている．

本研究では，PHOLD シミュレーションモデルのコードと， Backstroke ライブラリによって，自動生成されたプログラムでどのような違いがあるのかを調べることで，Backstroke ライブラリの有用性を評価する．また，ROSS シミュレータを用いて比較を行うことを目標とする．

2　関連研究

2.1　可逆計算

可逆計算とは，計算過程において任意の計算ステップで前の状態が高々一意に定まる計算のことである．可逆計算は逆計算をすることができ，並列離散事象シミュレーションでのロールバックに応用することができる．可逆計算を実行するためには，プログラムを可逆になるよう実装するか，プログラムを可逆化する必要がある．プログラムを可逆化するための変換器として Reverse C Compiler が知られている．

2.2　可逆プログラミング言語 Janus

可逆プログラミング言語とは，これらの言語を用いて実装した場合に可逆性が保証されるプログラミング言語である．可逆プログラミング言語として，命令型プログラミング言語の Janus が存在する．非可逆なプログラミング言語では，条件分岐や繰り返し文で可逆性が保証されないが，Janus では，アサーションという式を追加することによって，可逆性を保証している．

2.3　並列離散事象シミュレーション

ある状態がいくつかあり，その状態の間の状態変化によってモデル化できる離散的なモデルを考える．状態変化をイベントと呼び，イベントによって制御しながら，複数のコンピュータを用いて並列にシミュレーションを行うシミュレーションを並列離散事象シミュレーションという．

モデルを分割して各モデルごとに仮想時間を持たせ，それぞれの仮想時間を局所仮想時間（LVT），すべての仮想時間の最小値である仮想時間を大域的な仮想時間（GVT）と呼ぶ．この仮想時間を用いてイベントの実行順序を同期する．

並列にシミュレーションを行うための同期手法として，楽観的手法と保守的手法，楽観的手法と保守的手法を動的に切り替えながら同期する手法が提案されている．

2.3.1　楽観的同期手法

イベントの実行順序に誤りが出るまで同期をとらず実行する方法を楽観的手法という．各モデル間で実行順序に誤りがあった場合にはロールバックをして仮想時間を同期させる．

2.3.2　保守的同期手法

毎回スナップショットをとっておき，イベントの実行順序に誤りがないか確認しながら実行する方法を保守的手法という．NullMessage 法や問い合わせ NullMessage 法が提案されている [2].

2.3.3　楽観的と保守的手法を動的に切り替える手法

システムのパフォーマンスが高い限りは楽観的手法を用いて実行する手法である．[3] によると，HPDES と Lua-Based Simian PDES Engine という並列離散事象シミュレーションを行うソフトウェアを Python で開発して 3 倍パフォーマンスを改善したことが報告されている．

2.4　Reverse C Compiler (RCC)

RCC は C 言語を対象にしたプログラム変換機である．C 言語で書かれたプログラムを入力とし，前進実行用のプログラムと逆実行用のプログラムを生成する．生成した二つのプログラムを C コンパイラに入力することで可逆プログラムを実行する．RCC で生成されたプログラムは C 言語のプログラムであるから，ROSS を利用することができる．

2.5　Rensselaer's Optimistic Simulation System (ROSS)

ROSS はマルチプロセッサシステムやスーパーコンピュータ上で実行することができる並列離散事象シミュレータである．大規模なシミュレーションモデルを実行することが可能であり，C++ 言語全体の機能を使用できる．シミュレーションのプログラムでは，各モデルを logical

processes (LP) の集合として実装することによってモデル化する．楽観的，保守的どちらの手法を使用することもできる．

3　楽観的な並列離散事象シミュレーション

この研究では，主に楽観的な並列離散事象シミュレーションについて述べる．楽観的手法を用いた並列離散事象シミュレーションの中でも，TimeWarp 法と逆計算を用いた手法が提案されている．

3.1　楽観的な並列離散事象シミュレーションの同期手法

楽観的な並列離散事象シミュレーションの中では, TimeWarp 法が提案されている．TimeWarp 法は，同期に問題が発生した場合に，以前のメッセージを送信解除するアンチメッセージを送り，大域的に同期する手法である．TimeWarp 法はロールバックが連続して発生しなければ保守的手法よりも効率的に実行できることが知られている．プログラムに逆計算をするコードを追加し，同期に問題が発生した場合に，逆計算を実行することで大域的に同期する手法である．

4　Backstroke ライブラリ

Backstroke ライブラリとは，C++ を自動で可逆化するためのライブラリである．このライブラリの利点としては C++ 言語全体を使用可能であり，ROSS と並列にシミュレーションをすることが可能である．C++ の前進実行用のプログラムに，すべての形式の割り当てとメモリ割り当ておよびメモリ割り当て解除演算子のみを追加するだけで十分であり，他のすべての言語構造は元のプログラムのままにすることができる．様々なハードウェアアーキテクチャで使用可能であり，並列離散事象シミュレーションのモデルを開発が可能である．

複雑な離散事象モデルの場合，手作業で記述する必要のあるコードが少なくなるため，プログラムの実装効率が大幅に向上し，手書きのリバースコードと比較して，エラーが発生しにくく，コードのメンテナンスの負担が軽減されることも挙げられる．

欠点としては，C++ 言語全体を使用可能であるようにするために，ランタイムオーバヘッドが大きいことがあげられる．

5　行列ベンチマークモデル

[1] で提案された行列を用いたベンチマークモデルである．可逆計算に適した自動生成コードの正確性を確認するために開発された手法であり，ROSS シミュレータを用いてパフォーマンスの評価を行われた．[1] で評価が行われ以下のような結果が得られた．

行列のサイズが小さいとセットアップに時間がかかるため，Backstroke を適用したコードがオリジナルのコードと比べて，シミュレーションの実行時間が大きくなる．行列のサイズが大きくなるとセットアップによるオーバヘッドが無視できるようになるため，シミュレーションの実行時間がオリジナルのコードと同等になる．

6　手書きと Backstroke の比較

本研究では，オリジナルのコードと Backstroke のコードを既存のベンチマークモデルである PHOLD のモデルで用意した．二つのコードの比較を示し，比較した結果，オリジナルのコードと Backstroke のコードでどのような特徴があるかを考察する．

Backstroke を phold.c に適用した結果，プリプロセスが評価され，ヘッダファイルがすべて展開され，可逆性を損なう文が変換された．元のプログラム phold.c では mean = mean - lookahead; という代入文があるが，Backstroke を適用すると，(xpdes::avpushT(mean)) = mean - lookahead; となり，可逆性を保つために Backstroke が正しく適用されていた．

7　考察

PHOLD ベンチマークモデルでも，行列のベンチマークモデルと同様に，Backstroke を適用したコードでは，物理プロセスの量が小さいとシミュレーションの実行時間が大きくなり，物理プロセスの量が大きいとシミュレーションの実行時間が同等になると予想できる．

8　おわりに

本研究では，並列離散事象シミュレーションでの様々な手法の中で主に楽観的な手法について確認をした．PHOLD のベンチマークモデルを Backstroke を用いて可逆化した．行列のベンチマークモデルと同様に PHOLD のベンチマークモデルと推測した．ROSS シミュレータで Backstroke を適用したプログラムを動作させることができなかった．ROSS シミュレータを用いて比較を行えば，推測が正しいか判断でき，Backstroke の評価につながると考える．

参考文献

[1] Schordan, M., Oppelstrup, T. and Jefferson, D.: Generation of Reversible C++ Code for Optimistic Parallel Discrete Event Simulation, *New Gener. Comput.*, Vol.36, pp.257–280 (2018).

[2] 高井峰生，山城登久二，成田誠之助：離散事象並列シミュレーションにおける保守的同期手法の評価，情報処理学会論文誌, Vol.37, No.11, pp.2010–2019 (1996).

[3] Eker, A., Arafa, Y., Badawy, A.H.A., Santhi, N., Eidenbenz, S. and Ponomarev, D.: Load-Aware Dynamic Time Synchronization in Parallel Discrete Event Simulation, *Proc. the 2021 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS '21, Association for Computing Machinery, pp.95—105 (2021).