離散事象型並列シミュレータの実性能評価*

4P－6

高井 峰生　根本 貴由　成田 誠之助†

早稲田大学 理工学部‡

1 はじめに

離散事象型シミュレーションの並列処理 [2] は、近年の大規模システムに対応する為には必須の技術である。この離散事象型並列シミュレーションの処理効率は、主にシミュレート対象システムの持つ並列性と並列シミュレータの性能に依存する。しかし、従来は並列シミュレーションの一問題であるデッドロック解決のためのオーバーヘッドが重要視され、他の要因についてはあまり考慮されていなかった。本稿では、並列シミュレータの性能を左右する要因について考察し、実際の待ち行列シミュレータでそれぞれが与える影響を実測した。

2 離散事象型並列シミュレーション

2.1 並列シミュレーション手法

離散事象型シミュレーションの並列処理にはさまざまな方法が考えられるが、モデルの空間的な並列性に注目し、それぞれのプロセッシングエレメント (PE) でサブモデルを並列にシミュレートする方法が一般的である [2]。このような空間並列性を利用した並列シミュレーションでは、下記の処理を繰り返してシミュレートを進める。

1. 他の PE からのメッセージを受け取る.
2. 保証時刻を計算する.
3. 将来事象リストの先頭事象時刻 < 保証時刻の場合
   (a) 将来事象リストの先頭事象を取り出す.
   (b) 仮想時刻を事象の生起時刻に合わせる.
   (c) 事象が他の PE で処理されるものであった場合
   当該 PE にメッセージを送出する.
   そうでない場合事象を処理し，システムの状態変数を更新する.
   (d) 新たな発生事象を将来事象リストに登録する.

*Performance Evaluation of a Parallel Discrete Event Simulator

†Mineo Takai, Takayoshi Nemoto, Seinosuke Narita
‡School of Science and Engineering, WASEDA University

2.2 並列シミュレーションの処理

2.2.1 事象処理

将来事象リストから取り出した事象の処理は、離散事象型シミュレーションを行なう上で最も基本的な処理である。事象処理は基本的に “状態変数の更新”, “統計変数の更新”, “新たな事象の生成” から成り立ち、新たな事象の将来事象リストへの登録を除けば、必要とされるコストは静的に予測できる。

他の並列アプリケーションと同様、並列シミュレーションも負荷分散が非常に重要であるが、事象処理数を均等にすることによって静的な負荷分散を計るのが一般的である [3]。

2.2.2 将来事象リスト登録

生起事象の将来事象リストへの登録は、将来事象リストが長くなるに従って大きなコストを要求される。この為、大規模な (仮想時刻付近に多くの事象が存在する) システムを逐次シミュレートする上で将来事象登録は最も問題となる。並列シミュレーションにおいては、将来事象リストが少なくとも PE 数だけ分割されるため、逐次シミュレーションに比べてリスト長は短くなる。しかし、各 PE の仮想時刻に大きな差が生じるとリストの長さが極端に長くなる場合があり、将来事象リストへの登録は逐次処理よりも問題となる。

2.2.3 メッセージ送受信

並列シミュレーションでは、他の PE で処理されるべき事象をメッセージとして送信する。このメッセージ送受信にかかるコストは並列処理特有のオーバーヘッドであるので、メッセージ送受信数がなるべく少なくなるようなマッピングを行なう必要がある。

2.2.4 メッセージ待ち

並列シミュレーションでデッドロック解決手法に保守的手法を用いると、保証時刻が更新されない為に処理可

能な事象が生じず、他の PE からのメッセージを待つことがある。

保守的手法でこのようにメッセージ待ちが多く生じる場合、デッドロック解決手法に楽観的手法を用いても、ロールバックが頻繁に発生し効率が悪いことが予想される。

2.2.5 保証時刻更新

他の PE から処理の正当性を保証されている時刻の更新はメッセージ待ち時間 (履歴保存コスト) を減少させる上で非常に重要である。大規模なシステムをシミュレートする場合、一つの PE にマッピングされたモデル内でも並列性が存在する。この並列性を利用し、事象のスケジューリングに自由度を持たせれば不必要なメッセージ待ち時間を減少させることが出来る。しかし、論理プロセスの依存関係を調べ、事象が処理できるかどうかを判別すること自体オーバーヘッドとなるため、必ずしも保証時刻計算を細かく行なえば良いとは限らない。

2.2.6 デッドロック解決

2.1の繰り返しでシミュレートを進めると、シミュレート対象モデルの性質によってはデッドロックに陥ることがある。デッドロック解決手法はさまざまなものが提案されているが、そのオーバーヘッドはそれぞれの策によって異なる。例えば保守的手法の中で代表的なヌルメッセージ法 [1] であれば、ヌルメッセージ送出、楽観的手法の中で代表的な Time-Warp 法 [2] であれば、履歴保存・ロールバックがこれに相当する。

3 実シミュレータ評価

前節で述べたシミュレーションの処理内容は

- シミュレートモデルの並列計算機へのマッピング手法
- 事象スケジューリング手法
- デッドロック解決手法

に大きく影響される。本稿では、シミュレータ実現手段としてマッピング手法に iterative improvement method[3]、事象スケジューリング手法に論理プロセスクラスタ単位の事象スケジューリング、デッドロック解決手法にヌルメッセージ法 [1] を適用し、高並列計算機 AP1000 を用いて実際に待ち行列シミュレーションを行なった。

入力となる待ち行列ネットワークは、400 個の待ち行列が平均 2 入力 2 出力で近隣の待ち行列とランダムに結合したものを 10 個用意し、それぞれについて 6 個の異なるマッピングを用いて統計を取った。

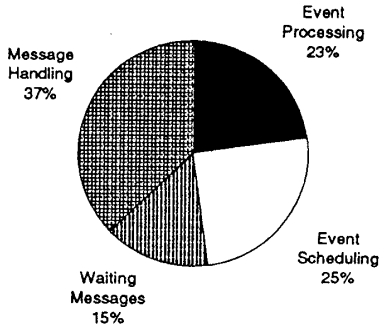

図 1: シミュレート実行時間占有比率

統計結果を図 1に示す。ただし、図中の事象処理コストは逐次シミュレーションの実行時間を基準に算出した。

従来指摘されてきた、デッドロック解決のオーバーヘッド (ヌルメッセージ法はメッセージ送受信に含まれる) に加え、将来事象リストへの事象の登録が大きな割合を占めている事が分かった。

4 まとめ

本稿では、AP1000 上の待ち行列シミュレータを用いて、並列シミュレーションの処理内容比率を実測した。従来から指摘されてきたデッドロック解決のためのオーバーヘッドに加え、実際には基本的な処理である将来事象リストへの登録コストも大きいことが分かった。今後は、将来事象登録のコストに大きく影響する事象スケジューリング法に改良を加えていきたい。

最後に本研究を行なうにあたり AP1000 を利用させて頂いた富士通研究所の方々に大変感謝致します。

参考文献

[1] K. M. Chandy and J. Misra. Distributed simulation: A case study in design and verification of distributed programs. *IEEE Transaction on Software Engineering*, Vol. 5, No. 5, pp. 440–452, September 1979.

[2] R. M. Fujimoto. Parallel discrete event simulation. *Communication of the ACM*, Vol. 33, No. 10, pp. 30–53, October 1990.

[3] B. Nandy and W. M. Loucks. An algorithm for partitioning and mapping conservative parallel simulation onto multicomputers. In *6th Workshop on Parallel and Distributed Simulation*, pp. 139–146. SCS, 1992.